Security and Privacy in the Smartphones Ecosystem

Alexios Mylonas

ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
DEPT. OF INFORMATICS

Information Security & Critical Infrastructure Protection Research Laboratory
Director: Professor Dimitris A. Gritzalis

Security and Privacy in Ubiquitous Computing: The Smart Mobile Equipment Case

Alexios **Mylonas**

Technical Report Series
No. 3 (2013)
Report code: AUEB-CIS/REV-0313/v.1.1/20.07.2013

July 2013

Athens University of Economics and Business, Dept. of Informatics
Information Security & Critical Infrastructure Protection
Research Laboratory
76 Patission Ave., Athens GR-10434, Greece
Tel.: (+30) 210.8203157, Website: www.infosec.aueb.gr

Table of Contents

List of Figures

List of Tables

List of Acronyms

ETP	Evidence Transmission Protocol
MDM	Mobile Device Management

(this page is intentionally left blank)

Chapter 1: Introduction

1.1 Motivation and approach

Merge:

Smartphones are some of the devices that enhance Weiser's vision of ubiquitous computing (Weiser, 1991). Their small size, mobility, connectivity capabilities, and multi-purpose use are some of the reasons for their vast pervasiveness (Gartner, 2010).

Malicious software or malware (Andleman, 1990; Cohen, 1989; Kephart & White, 1991) has also appeared in smartphone platforms (Hypponen, 2006), but initially their occurrences and severity were limited. Nonetheless, recent reports show that the risk of malicious application execution on smartphones is severe and contingent (McAfee, 2010; CISCO, 2011). Moreover, the use of smartphones extends the infrastructure perimeter of an organization, thus amplifying the impact and the risk of potential execution of malicious applications (Sindhu et al., 2010).

Apart from the increasing smartphone sales (Gartner, 2010), the annual downloads for applications developed for smartphones and distributed from official application repositories were also bound to increase by 117% in 2011 (Gartner, 2011). In addition, popular web applications (Gmail, YouTube, etc.) and social networks (Facebook, Twitter, etc.), are being accessed on mobile devices through native applications, instead of their usual web browser interface. In this context, smartphones contain a vast amount of the user's data, thus posing a serious privacy threat vector (PAMPAS, 2011; ENISA, 2011; GSMA, 2011). These data are augmented with smartphone sensor data (i.e. GPS) and data created by daily use (personal or business) making the device a great source of data related with the smartphone owner. This data source, if maliciously collected, can be used by attackers trying to increase their revenues (e.g., with blackmail, phishing, surveillance attacks). Hence, attackers may try to infect smartphones with malware applications, harvesting smartphone data without the user's knowledge and consent. It should be noted that the growing smartphone use by non-technical and non-security savvy people increases the likelihood of using smartphones as a security and privacy attack vector.

The security model of smartphone platforms has, under these circumstances, two contradicting goals. On the one hand, it must provide mechanisms to protect users from attacks and, on the other hand, it must attract third party developers, since the popularity of a platform depends on the attractiveness of its applications. The former goal is approached by each smartphone platform under a non unified and standardised approach that its effectiveness is controversial (Sophos, 2011). For the latter smartphone platforms provide developers with

development friendly environments that include extensive documentation, programming libraries, and emulators. This development friendliness, nonetheless, may also be used to implement applications that can compromise the security and privacy of smartphone users more effectively.

Smartphones, as ubiquitous devices, merge with a person's everyday life. As a recent report[1] points out, smartphone sales outnumbered these of feature phones, thus, acquiring a significant user base. Smartphones are characterized by mobility, context-awareness, and diversity on the data sources that they integrate.

Smartphones create, store, process and transmit user data, which vary in meaning or type according to the device's use (e.g. personal, business, governmental. Go to android

Previous literature has revealed that smartphone users use the same device both for personal and business purposes. Therefore, personal information can be found in most smartphones. These include data which can identify a user, his preferences and habits or more sensitive information, such as his political or religious beliefs, his physical or mental health, his sexual life and others. Disclosure of such data to third parties may cause significant distress to an individual and may even be subject to sanctions, depending on the context. In addition to concerns about personal data, there are confidentiality issues for other data categories, such as intellectual property or trade secrets, which fall into the corporate domain. These are also confidential information and require analogous protection, but the regulatory framework for their protection varies significantly.

The ubiquity of Android applications, Android's openness regarding the sources of available applications, and the variety of available personal data are some of the reasons that privacy risk is increased in Android (Marinos and Sfakianakis, 2012). Previous research reveals the likelihood of applications violating privacy in Android's official app repository or app market (i.e. Google Play) (Enck et al., 2010; 2011). Android apps from Google Play were found to leak privacy sensitive information, such as the user's location, device identifiers (e.g. device unique code (IMEI), subscriber unique code (IMSI)) (Enck et al., 2010; 2011). A recent study on Android malware (Zhou and Jiang, 2012) confirms that malware has been found to actively harvest various information on infected phones, including SMS messages, phone numbers, as well as user accounts. SMS messages do not only contain personal information of the user, but often contain activation codes, i.e. user credentials. In response to the increasing malware submissions in its repository, Google introduced Bouncer in February 2012, a service that performs malware analysis in the repository's apps. Moreover, only the latest version of Android (i.e. v.4.2) includes a thin client that performs app verification for all the apps that re-

[1] International Data Corporation (IDC). Smartphones Outstrip Feature Phones for First Time in Western Europe as Android Sees Strong Growth in 2Q11, September 2011.

side in the device -both from Google Play and alternative sources. Nonetheless, a recent evaluation proves the ineffectiveness of this mechanism (15% detection ratio) (Jiang, 2012).

Meanwhile, privacy violations can occur even when a user grants access to protected private data (e.g. contact list, exact location, etc.) to a benign app, i.e. one not trying to violate user privacy. This holds true, since the app may either be used as a confused deputy (Chin et al., 2011; Felt2011a; Grace et al., 2011) -accidentally allowing other malicious apps to use its functionality to access the resources- or be bundled with a malicious advertisement library (Grace et al, 2012, Pearce et al., 2012; Stevens et al., 2012), which shares the app's permissions and misuses them to violate user privacy (e.g. tracking, surveillance, etc.). Moreover, in Android the effort for the implementation and deployment of a privacy violating app is achievable even by an attacker with average skills (Mylonas et al., 2011a). In addition, benign Android apps tend to request more permissions than needed for their intended functionality, due to the vagueness of the API documentation (Felt et al, 2011).

Often the discussion focuses on Android. This platform was selected for the following reasons:

- It is currently the smartphone platform with the largest user space (79%, in the 2nd quarter of 2013), as well as the only platform with an increasing rate in its user base (~15% increase from the 2nd quarter of 2012) (Gupta et al., 2012).
- It is considerably portable, i.e. compatible with the hardware of several smartphones (e.g. Samsung, LG, etc.).
- Android is open source, hence the details of its security model are readily accessible and have been studied in the smartphone literature.
- It is extensible, i.e. a custom ROM can be created either from device manufacturers (e.g. for unique driver support) or by researchers (e.g. to add enhanced security or functionality to its core components).
- It allows the installation of applications from sources other than the official app repository (i.e. Google Play). Thus, there is no need to root or ‘jailbreak’ the device in order to install the agent to a device.
- Development and testing in Android is considerably aided by the existence of programming libraries and device emulators that are freely and readily accessible.

(this page is intentionally left blank)

Chapter 2: Background

"Imagination is more important than knowledge." – A. Eistein

2.1 Security and privacy in Ubicomp

TBC

2.2 Smartphone: Definition and data taxonomy

2.2.1 Definition

The term *smartphone* is frequently used by the industry and research community to refer to state-of-the-art cell phone devices. These devices are considered 'smart', and are distinguished from ordinary and technologically constrained cell phones. The latter, which are referred to as *feature phones*, are often restrained by small screen size, limited processing and network capabilities, and execute, in general, a proprietary and not adequately documented operating system. Thus, their security is mainly based on secrecy or as the IT security community refers to, as "security by obscurity".

In contrast with the term 'feature phone', a widely accepted definition for 'smartphone' can hardly be found in the literature. Becker et al. (2011) define smartphones as devices which: (a) "contain a mobile network operator smartcard with a connection to a mobile network", i.e. a SIM or USIM card in GSM and UMTS systems, respectively, and, b) "have an operating system that can be extended with third-party software". However, this definition appears to be rather broad. Also, its properties are valid for feature phones. For instance, the Motorola V3i[2] feature phone would be incorrectly classified as a smartphone, as it contains a mobile carrier SIM card and has a proprietary OS that can be extended by third party applications (specifically with MIDP 2.0 Java applications).

The alternative definition of a smartphone, which is adopted in this thesis, is the following: *smartphone is a cell phone[3] with advanced capabilities, which executes an identifiable operating system allowing users to extend its functionality with third party applications that are available from an application repository*. According to this definition, smartphones must include sophisticated hardware with: a) advanced processing capabilities (e.g. modern CPUs,

[2] `http://www.motorola.com/mdirect/manuals/V3i_9504A48O.pdf`

[3] A *cell phone* is a device which: a) *is used primarily by its holder to access mobile network carrier services*, e.g. phone calls, Short Message Services (SMS), etc., and b) *contains a smartcard*, which is controlled by the network carrier (i.e. SIM or USIM card) and *incorporates a billing mechanism* for the used network carrier services.

sensors), b) multiple and fast connectivity capabilities (e.g. Wi-Fi, HSDPA), and (optionally) c) adequately limited screen sizes. Furthermore, their OS must be clearly identifiable, e.g. Android, Blackberry, Windows Phone, Apple's iOS, etc. Finally, the OS must allow third party application installation from application repositories ('app markets'), e.g. Android Market, BlackBerry App World, App Hub, App Store, etc.

2.2.2 Data Source Taxonomy

During regular (e.g., daily) use, smartphone data are created, processed, and consumed or stored on various sources. The following taxonomy is based on *data source* by extending a prior taxonomy in (Mylonas, 2008):

- *Messaging Data* derive from: (a). mobile carrier messaging services i.e. Short Message Service (SMS), Enhanced Messaging Service (EMS), Multimedia Messaging Service (MMS), or (b). Instant and e-mail messages. They also include messaging logs, e.g. receiver, sender, delivery time and date, attachments, etc.
- *Device Data* are data that (a) are not related to any third party application, or (b) contain device and OS specific information. They may reside in internal (e.g. flash drive, flash memory) and removable (e.g. microSD cards) storage media. Some examples include images, contact list, Wi-Fi MAC address, device serial number, etc.
- *(U)SIM Card Data* reside either in a Universal Subscriber Identity Module (USIM) or Subscriber Identity Module (SIM) card. Typical examples are the International Mobile Subscriber Identity (IMSI)[4] and the Mobile Subscriber Identification Number (MSIN)[5]. This source often contains SMS and contact list entries.
- *Application Data* include permanent or temporal data that are necessary for application execution. They may be stored as individual files, or constitute a local database, e.g. SQLite. A typical example is a flat dictionary text file.
- *Usage History Data* are used for logging purposes, such as: (a) call history, which contains incoming or outgoing phone call logs, (b) browsing history, i.e. temporary data created while the user browses local or remote files, (c) network history logs for wireless connections, e.g. Wi-Fi SSIDS, Bluetooth pairing, and (d) event logs, which are created by the OS for system monitoring and debugging.
- *Sensor Data* are created by dedicated hardware. Camera(s) and microphone(s) are two popular sensors. Other sensor hardware include: a) GPS sensor, b) accelerometer, c) gyroscope, d) magnetometer (i.e. digital compass), and e) proximity sensor. These are used to infer the exact device location, its orientation, the way the device is being moved, its head-

[4] A unique number that identifies the subscriber to the network.
[5] The 10-digit phone subscriber number.

ing direction, and the device distance from a surface, respectively. Sensor hardware, such as the light sensor and the temperature or pressure sensor, are present in some smartphones, measuring the device environment surroundings (context). Sensor data are mostly consumed on the fly and are not typically stored for later retrieval. Finally, they may be used as metadata (e.g. in geotagging where GPS data are embedded in photographs and videos).

User Input Data include user gestures, hardware button presses, and keystrokes from a virtual or smartphone keyboard. All involve user interaction with the device. User input data are often consumed on the fly, or stored in a keyboard cache for performance reasons (e.g. improvement of spelling software).

2.2.3 Information Type Taxonomy

Data are classified on the basis of *information type*.

Smartphone data hold various meanings. Their classification, according to the *information type* they may infer, led us to the following taxonomy:

- *Personal data* are directly related to an identified individual. They are considered private and should not be made public. Examples include the content of a user's communication, images, videos, etc. Disclosure or unauthorized modification may result in embarrassment, reduction in self-esteem, or legal action.
- *Business data* (or corporate intellectual property) refer to data with commercial and economic corporate significance. These include marketing information, products under design, etc. Unintended disclosure of this data to the public or competitors may lead to strategic advantage loss, copyright breach, loss of goodwill, etc. Such data are usually likely to exist in a 'personal' smartphone, if it is (even occasionally) used for business purposes.
- *Government data* affect: (a) public order, (b) international relations, or (c) performance of public service organization(s). They differ from business data, because they hold national or international significance, as opposed to business value.
- *Financial data* refer to records of financial transactions, current financial holdings or position. Unauthorized modification, disclosure, or unavailability may lead to financial loss or contract breach (e.g., due to delays).
- *Authentication data* refer to user credentials, e.g. passwords, PINs, biometrics, etc. Their unauthorized access may lead to impact, such as financial loss, personal information disclosure, legal consequences, etc.
- *Connection/ Service data* refer to data, which are required for network connections. They include connection identifiers, such as Wi-Fi MACs, IMSI, or IMEI, as well as data regarding the connection itself, such as the Wi-Fi joined networks history.

Table 1 associates the two dimensions. These associations are used in §XX as the basis for the data impact valuation.

Table 1. Smartphone data taxonomy

Information Type / ***Source***	*Personal*	*Business*	*Government*	*Financial*	*Authentication*	*Connection/Service*
Messaging	✓	✓	✓	✓	✓	
Device	✓	✓	✓	✓	✓	✓
USIM Card	some	some	some	some	✓	✓
Application	✓	✓	✓	✓	✓	✓
Use history & caching	✓	some	some	some		✓
Sensor	✓	✓	✓			
Input methods	✓	✓	✓	✓	✓	

2.3 Threat model

This section briefly introduces our threat model, which includes user and adversarial details. Unless otherwise stated, the discussion in the following chapters is in accordance with the assumptions of this threat model.

Users. In our work smartphone users are assumed to be *average users*, i.e. not security and technically savvy. Regarding their *practices* it is assumed that these users:

a) Install applications (or 'apps') into their devices, on a regular basis (e.g. daily),
b) Install them only from the official app repository or app market (i.e. Google Play),
c) Protect their smartphone only with the built-in security mechanisms of the platform,
d) Enable Internet connectivity on their devices, either from a WLAN (i.e. WiFi), or WAN connectivity from the mobile carrier (e.g. 3G, 4G, etc.)

This means that this work does not take into consideration smartphones with a modified operating system. Therefore, the devices are not “rooted” or “jailbroken”. Also, the device does not execute a more secure custom operating system, which adds additional security mechanisms (e.g. enabling taint tracking to thwart data leakage (Enck 2010)). With the former, it is assumed that every app is executed in a sandboxed environment (e.g. as the one used in Android (Google, 2013a)). The later, is in accordance with the expectations for a user without a security mindset and without advanced technical skills. Finally, this thesis regards that the device is not part of a corporate Information Security Management System (ISMS)

and managed from another individual. Thus, it is assumed that Mobile Device Management (MDM) solutions that isolate corporate from personal data, etc, are not present on the device.

Adversaries.

2.4 Smartphone ecosystem

The smartphone ecosystem is characterized by its *heterogeneity* and *dynamic* nature. Currently, the ecosystem cannot be regarded as *stable*, since the changes in its technology and the introduction of new smartphone OS providers often reshape it. This is most evident in the market share of Symbian. This smartphone OS held the majority of the smartphone market share in the beginning of this work (~47%, fall of 2009 (De La Vergne et al., 2010)). Nonetheless, today the OS' is inactive - its software maintainer (Nokia) has replaced it with Windows Phone[6] as the primary OS in device shipments – with a market share less than 1% (Gupta et al., 2013) and the majority of smartphone's market share is held by Android. Furthermore, the technological innovation that occurs in the smartphone ecosystem is often rapid; for instance Android has released 10 versions for smartphones and two for tablets,[7] since its introduction in the ecosystem in 2008.

Currently, the security models of smartphone platforms do not follow a standardized and homogenous approach. They range from more relaxed security models, which allow users to install apps from any source (e.g. Android), to more controlled ones ("walled garden"), which allow apps only from the official app repository (e.g. iOS) (Barrera and Van Oorschot, 2011; Mylonas et al, 2011a). However, the security models restrict app execution only in sandboxes; therefore apps can access the functionality that is allowed by the platform's sandbox profile. These restrictions, offer a line of defence against malicious apps, but on the same time restrict the security arsenal that is available to smartphone users. For instance, sandbox restrictions hinder the availability of antivirus software in iOS.

The popularity of computing platform is one of the reasons that draw the attention of malicious writers. In this context, a lot of smartphone malware families target Android (Zhou and Jiang, 2012), which is currently the most popular smartphone OS. On the contrary, nowadays, there are hardly any new malware families for Symbian, which in the past was the platform that was most targeted by malware, due to its popularity.

The following subsections include a description of the security models of the smartphone platforms that were popular throughout this work, namely: (a). Android, (b). BlackBerry, (c). Symbian, (d). iOS, (e). Windows Mobile and (g). Windows Phone. Before moving to the next

[6] http://www.microsoft.com/en-us/news/press/2011/feb11/02-11partnership.aspx
[7] http://developer.android.com/about/dashboards/index.html

subsection is worth noting that is unclear whether the state of ecosystem will change with the introduction of new smartphone platforms, such as Mozilla's Firefox OS and Ubuntu Phone.

2.4.1 Android

Android is a Linux based, open source operating system (OS) developed and maintained by Google. It provides a free and publicly available Software Development Kit (SDK) that consists of tools, documentation and emulators necessary for the development of new applications in Java. At the writing of this chapter, Android holds the largest market share 79% in Q2 of 2013 (Gupta et al., 2013).

A core element of the Android security model is the *manifest file* (Google, 2013a). The manifest provides the necessary information to Android for the execution of an application. Security-wise, the manifest file is crucial for the system, since a developer defines within the *application permissions*, namely: (a) the way the application interacts with the system via access to system API, and (b) the way the system and the other applications interact with the given application's components.

Every application runs in a sandboxed environment without any permission to perform an action that can impact the operating system itself (e.g. crashing), other applications (e.g. disabling an application's components) and the user (e.g. surveillance). Applications request authorization from the user for their permissions during installation. No further permission checks are performed during the applications' execution. Moreover, the user cannot grant a subset of the requested permissions. Hence, the user either accepts all permissions or postpones the installation.

Every Android application has to be digitally signed by its developer. Android's security model then maps the signature of the developer with a unique ID of the application package and enforces signature level permission authorization. Nonetheless, Android's security model does not mandate that a developer's certificate must be signed by a trusted certificate authority. As a result, applications are usually signed with self-signed digital certificates; hence, providing only poor source origin and integrity protection. This preserves the anonymity of a potential attacker, since the certificate is not verified by a Trusted Third Party (TTP).

A developer may distribute her application either in the official application repository maintained by Google, the Google Play (formerly known as Android Market), or via another source (e.g. Amazon AppStore, forums, etc.). Google does not enforce any restriction in the installation of applications outside its repository (e.g. forums, other markets, etc.). On the other hand, Google developed technologies to remove applications from devices and the Android Market in case they are proven malicious.

Until the February of 2012, applications could enter the Android Market without undergoing analysis for malicious behaviour. Thus, an attacker could only use a Google account and pay a small fee for malware distribution in the Android application repository. In response to the increasing number of different malicious apps that were found in its repository (Zhou and Jiang, 2012), Google introduced Bouncer in February 2012, a service that performs malware analysis in the repository's apps. Moreover, recently Google included a thin client, which performs app verification for all the apps that reside in the device -both from Google Play and alternative sources, in Android (i.e. v.4.2). Nonetheless, a recent evaluation proved the ineffectiveness of this mechanism (15% detection ratio) (Jiang, 2012).

According to (Google, 2013b), Gingerbread and Jelly Bean are the dominant Android version, currently deployed in the majority of the Android devices (67%). Older versions of the OS (e.g. Froyo, Eclair) are still in use, but with a very limited distribution among Android users. The adoption of the latest versions of the OS (version 4.2.X and 4.3) is still low. Moreover, Android does not support updates for all Android devices, not even for Google phones (e.g. Nexus S).[8]

Table 2: Distribution of Android versions (**Google, 2013b**)

Version	*Codename*	*API*	*Distribution*
1.6	Donut	4	0.1%
2.1	Eclair	7	1.2%
2.2	Froyo	8	2.5%
2.3-2.3.2	Gingerbread	9	0.1%
2.3.3 -2.3.7	Gingerbread	10	33.0%
3.2	Honeycomb	13	0.1%
4.0.3- 4.0.4	Ice Cream Sandwich	15	22.5%
4.1.x	Jelly Bean	16	34.0%
4.2.x	Jelly Bean	17	6.5%

2.4.2 BlackBerry

BlackBerry is an operating system maintained by Research In Motion Inc. (RIM). The OS is executed on BlackBerry smartphones and tablet devices created by RIM. According to Gartner (Gupta et al., 2013), RIM's worldwide market share is limited only to 2.7% in Q2 of 2013, losing for the first time the third spot in the smartphone market share.

BlackBerry is a proprietary OS and, thus, comprehensive documentation describing the OS 'nuts and bolts' is not available. RIM provides the BlackBerry SDK, which includes the documentation, tools, API and emulators that are necessary for application development.

The platform's security model (BlackBerry, 2013) enforces restrictions to third party applications trying to access protected APIs, by demanding their signing with a cryptographic key

[8] http://asia.cnet.com/google-no-android-4-2-for-nexus-s-and-xoom-owners-62219464.htm

that is only provided by RIM. A developer needs to pay a small fee in order to acquire a valid RIM signing key pair. BlackBerry maintains an application repository, the App World, without restricting application installation from other application repositories. Currently apps that are submitted to Blackberry's app market undergo security analysis by BlackBerry to filter out apps with malicious behaviour. Similarly to Android's official repository, application testing was not offered initially to BlackBerry users that downloaded apps from its official app market, exposing its user to malware (Mylonas et al., 2011b).

2.4.3 Symbian

Symbian OS is an open source OS, which used to be maintained by Nokia.[9] While in the beginning of this research Symbian had the majority of the smartphone market share, the OS' population had a vast decrease, falling to 0.3% in the Q2 of 2013 (Gupta et al., 2013).

Symbian provides multiple free and publicly available SDKs. The SDK includes the tools, documentation and emulators that are necessary for the development of new applications, written in C++. However, the platform provides a provides a development environment that is not attractive to developers (Mylonas et al, 2011a) and this is one of the reasons for its failure.

The cornerstone in Symbian's security model is the use of *capabilities* (Nokia, 2013) for defining restrictions to sensitive platform APIs. These capabilities are grouped in the following categories: (a). *basic*, (b). *extended*, (c). *manufacturer*. The first category includes basic functionality (e.g. access to the network, access to location data, etc.). The user is prompted for the authorization of such functionality during app's installation. The second capability category controls the use of sensitive API that is only granted through the *Symbian Signed* process. The last capability category controls application access to the most sensitive API in the platform (i.e. *All-Files*, *DRM*, *TCB*). These capabilities are only granted by a Device Manufacturer (e.g. Nokia, Sony Ericsson, etc). As indicated by Nokia (2013) the *basic* capability category contains sufficient functionality for application development.

Signing the application's package file *(.sis file*) is required by the Symbian security model during app installation. Signing ensures that the application is not using API apart from the one corresponding to the applications signing level. If the application uses only *basic* capabilities the developer can self sign it. Self-signing has the advantages to be performed in the developer's computer and avoiding to map the application's installation package file with a device IMEI. This allows an app to be installed in multiple devices, but the smartphone user will be prompted with security warnings at installation time, since the signing key is not trusted. To eliminate these warnings and access sensitive capabilities the developer submits her application to *Symbian Signed* along with the list of device IMEIs. However, guidelines for by-

[9] At the writing of this thesis development maintenance of Symbian was outsourced to Accenture.

passing Symbian's security model are available (Symbian Freak, 2013), allowing the execution of unsigned applications that access any functionality bypassing the platforms capabilities. Finally, Symbian permits the installation of applications that do not reside in the platform's official app repository, i.e. OVI store.

2.4.4 iOS

iOS is a proprietary operating system maintained by Apple. According to (Gupta, 2013), iOS is the second most popular smartphone platform, but with a popularity far less than Android's, specifically 14.2% in the Q2 of 2013.

Apple provides, after registration to the company's development program, documentation, tools and the necessary API for application development in Objective C programming language. Nevertheless, it should be noted here that this toolset is only compatible with Mac OS operating system.

The security model of iOS follows a 'walled gardened' approach regarding app distribution, More specifically, it permits, only, the installation of applications that have been signed by Apple and reside in the official application repository, the App Store. For application distribution in the repository the developers incur an annual cost. Furthermore, the security model of iOS is not permission based as in Android. Its cornerstone is an application testing mechanism - also referred as 'application vetting' mechanism, which is controlled by Apple. During application submission each application undergoes automatic and manual testing to ensure its functionality consistency, official API use and 'absence' of malicious activity. Nonetheless, the testing process and criteria, which are applied by Apple, are not publicly available.

Once the application is installed to the device the user neither controls nor is prompted when an application accesses most OS' sensitive resources. For few sensitive resources (e.g. location data) the user will be prompted the first time that an application uses the resource and has the ability to revoke application access to the resource in the OS settings. For the rest resources, for instance Internet access, control over the access to them is not provided to iOS users. Finally, regardless of Apple's app restrictions and app vetting, spyware has been found in App Store (Egele, 2011).

2.4.5 Windows Mobile

Windows Mobile is a smartphone OS developed and maintained by Microsoft. Even though, it is the predecessor of Windows Phone its security models follows a different rationale. In specific, its security model of Windows Mobile (MSDN, 2013a) depends on the

device policy that is enabled on the device. This policy specifies which applications are allowed to execute on the device, what functionality of the OS is accessible to them, how desktop applications interact with the smartphone, and how a user or an application accesses specific device settings. The enabled policy on a Windows Mobile smartphone is either *one-tier access* or *two-tier access*.

A device with one-tier access policy enabled, only controls if one application runs on the device or not, without inspecting if the application is using sensitive API. This decision depends on whether the application's installation package file (*.cab file*) is signed with a certificate that exists in the device's certificate store. If the application is signed with a known certificate, then the application executes in privileged mode. In this case the app executes with the ability to call any API, access and modify any resource (e.g. the device's file system and registry). Otherwise, if the application is unsigned or signed with a certificate that is not known, further policy checks occur in order to grant application execution. In this case, security policies define whether the user is prompted to give her consent for the application to run or block its execution. It must be clarified that if the user permits the execution, then the application will run in privileged mode. This means that an unknown and unsigned application is granted full access to the device resources.

On the other hand, a device with two-tier access policy enabled, apart from controlling application execution, it also checks runtime permissions by controlling the APIs that the application uses. Access to protected API is determined by the application's digital signature, similar to Symbian's security model. More specifically, if the application is signed with a known certificate (i.e. one that is present in the device's certificate store), then it is executed without further checks and it is authorized the permissions that correspond to the certificate's class. If the certificate belongs to the *Privileged Execution Trust Authorities* certificate store, the application is authorized privileged permissions. Otherwise, the application is executed in normal mode. When the application is unsigned or signed with an unknown certificate, then further checks occur to determine if the application executes in normal mode. It is worth noting that the functionality provided by normal privileges is enough for most third-party applications.

According to (MSDN, 2013) the default security configuration of Windows Mobile, provides a weak security protection as: (a). it allows the installation of unsigned applications or singed ones with an unknown certificate, hence the platform does not provide any assurance about the application author or its integrity, and, (b). in case (a). the user is prompted to authorize the execution of the application. Hence, in both access tiers of the default security configurations, unsigned and unknown code is executed with the user's approval either in normal mode (*two-tier access*) or privileged mode (*one-tier access*). Furthermore, although

one-tier access does not provide strong security, it is the default access tier in some devices (MSDN, 2013).

Similarly to Android, the security model of Windows Mobile provide an app 'kill-switch', i.e. it includes security mechanisms enabling a Mobile Operator to revoke (i.e. remotely remove or disable) applications running on smartphones (Microsoft, 2010c). The revocation may concern either (a). a class of applications signed with the same certificate, where the corresponding certificate is being revoked, or (b). a specific application binary, where the hash of the binary is being transferred to the device.

Microsoft provides freely the required development toolkit (i.e. SDK, emulator, documentation, etc.) for the implementation of third-party application in Windows Mobile. Its supported implementation languages (e.g. C#, Visual C++) are compatible with the *Compact .NET Framework*.

2.4.6 Windows Phone

Windows Phone is the latest smartphone OS that is maintained by Microsoft. Even though this OS is the descendant of Windows Mobile their differences, especially with regard to the security model, enables it to be regarded herein as a separate smartphone OS. Currently, Windows Phone is the third most popular smartphone platform, surpassing BlackBerry's market share for the first time in Q2 of 2013. However, the market share of the OS is limited, i.e. 3.3% (Gupta et al., 2013). In February 2011 a partnership[10] between Nokia and Microsoft was announced where Nokia adopts Windows Phone as its primary smartphone operating system.

The security model of Windows Phone is based on application sandboxing in conceptual chambers (MSDN, 2013b), where access to protected resources is granted via capabilities, which are similar to permissions in Android. Third party applications are executed in a least privileged chamber, where access to resources is controlled by capabilities that are *indirectly granted* by the user at installation time and cannot be elevated during execution time. The capabilities are included in an application's manifest file, are *automatically granted* at installation. Contrarily to Android, users are not explicitly prompted to grant access to the application's capabilities during its installation. Users are informed about the capabilities that an application uses via: (a). the application's detail page in the official repository, the Marketplace, (b). an explicit prompt only for capabilities that have legal requirements for explicit consent collection, and, (c). the application itself when the location capability is used for the first time.

[10] http://www.microsoft.com/presspass/press/2011/feb11/02-11partnership.mspx

This way, the platform's security model assumes that users *indirectly* accept the capabilities by installing the application. Moreover, this approach of granting capabilities follows – similarly to Android- an *all-or-nothing* approach. This means that users cannot grant only a subset of the capabilities that the application requests.

The security model of Windows Phone permits only the installation of third party applications that are signed by Microsoft and available in the Marketplace. During application submission each application is tested and the developer is authenticated during registration. Each submitted application is tested for compliance with *Windows Phone Application Certification Requirement*. These requirements apart from testing the application's functionality and performance involve security tests for malware detection. Moreover, Microsoft employs a remote application removal mechanism to remove malicious applications that manage to enter the application repository and Windows Phone devices.

Finally, the platform provides a free SDK that consists of the tools, documentation and emulators, which are necessary for the implementation, using the Silverlight technology, of application for the Windows Phone.

2.5 Smartphone security

TBC

2.5.1 Physical Threats

TBC

2.5.2 Web Threats

2.5.3 Smartphone Privacy Impact Assessment (PIA)

Privacy essentially refers to the protection of personal data or `Personally Identifiable Information' (PII), but it can have wider interpretations in different, non-IT contexts, i.e. bodily privacy, privacy of personal behaviour or privacy of personal communications (ICO, 2013). Privacy Impact Assessment (PIA) refers to ``*a systematic process for identifying and addressing privacy issues in an information system that considers the future consequences for privacy of a current or proposed action*" (Warren et al., 2008). This process is a risk assessment process, focused on privacy, and it is mainly associated with the collection, use or disclosure of personal information. It is a tool for decision support and it is often a regulatory requirement for public information systems, as it serves as a means to address citizens' concerns over privacy. PIAs are currently used in the UK, Canada, USA and Australia, for projects of the public sector, e.g. a new system, technology, pilot, rule, program, or other collec-

tion. If we consider this concept within the smartphone context, PIAs could potentially be performed by application developers or by the marketplace, as a means to increase the users awareness and to present more clear policies regarding the use of the customers' personal data.

related: This work refines the risk assessment method proposed in (Theoharidou, 2012). Its focus is on a subset of smartphone threats that were presented in (Theoharidou, 2012), namely privacy threats. The privacy threats that are applicable to user data are identified and mapped to the permission combinations that are required for the threat to occur. Threat likelihood is computed from the frequency of these permission combinations on Google Play. Our work relates to (Sarma et al., 2012), which studies permission combinations as risk signals. The analysis of risk signals, however, is based on an outdated app sample -- the sample was collected in 2011, before the introduction of Bouncer that changed the frequency of permission combinations by filtering out apps from Google Play. Also, the analysis focuses only to a subset of the available permission combinations that may violate user privacy. DroidRisk (Wang et al., 2013) is, to the best of our knowledge, the first attempt to quantitatively assess the risk levels of both Android permissions and apps. Its assessment is based on patterns of permission requests in malicious and benign apps. However, DroidRisk's analysis is limited only to statistics on individual permissions and not on their combinations.

Our method can benefit from a generic impact valuation such as (Felt et al., 2012b), which includes a ranking of risks according to user upsetness. This generic impact valuation can be facilitated to create static, generic risk profiles. Finally, previous works (e.g. (Barrera et al., 2010; Zhou2012}) often include statistics about the popularity of individual permission in Google Play. Our work, provides up to date popularity of permission combinations that can violate user privacy when they are misused by apps.

2.6 Evaluating the malware threat

This subsection examines the feasibility and easiness of malware development on smartphones by average programmers that have access to the official tools and programming libraries provided by smartphone platforms. This is achieved through a proof of concept study that aims on evaluating the ease of malware development against users of smartphone devices. Thus, issues like state of the art attacks that might be performed by sophisticated attackers (Seriot, 2010; Lineberry et al., 2010) and the relation between malware attacks on smartphones and desktop computing devices are out of the scope of this chapter.

Our work contributes towards this direction by: (a) proposing a set of evaluation criteria, assessing the security level of well-known smartphone platforms (i.e. Android, BlackBerry, iOS, Symbian, Windows Mobile, Windows Phone), in terms of the development of privacy-attack malware, and, (b) providing a comparative case study analysis where a proof of con-

cept implementation of a location tracking malicious application is attempted in the above-mentioned smartphone platforms.

2.6.1 Scope of analysis

In this section we discuss the security models and development environments of the surveyed smartphone platforms: (a). Android OS, (b). BlackBerry OS, (c). Symbian OS, (d). Apple iOS, and (e). Windows Mobile 6 OS. Our analysis focuses on application installation and execution. Security mechanisms that are used for the physical protection of the device (data encryption, anti-theft solutions, etc.) are out of the scope of this chapter.

2.6.2 Comparative evaluation of smartphone platforms

This section provides a comparative evaluation of the smartphone platforms in terms of malware development and distribution. Our analysis examines the feasibility of attacks implemented by average application developers. More specifically, the presented evaluation is based on: (a). the definition of qualitative evaluation criteria, and (b). a proof of concept malware implementation study, in which the development of a location tracking application is examined. At this point it should be stressed that any sophisticated attack conducted by experienced attackers, as well as, publicly available malicious code used by script kiddies are not examined in this chapter. Furthermore, a comparison with malware development in desktop computing is not examined in this chapter either.

2.6.2.1 Evaluation Criteria

A comparative evaluation of smartphone platforms is performed by defining and using a set of evaluation criteria, which are elaborated in this section. The proposed criteria concern the development platform and the developer. From the proposed evaluation criteria the former are objective, relying solely on characteristics of the smartphone platform. The latter are subjective, giving details about the attack development effort and as a result depend on the developer's skills and background. The latter, however, are given as an indication on the effort needed to implement a malicious smartphone application.

Our overall approach focuses primarily to the objective criteria (development platform), while at the same time takes into account the subjective criteria (regarding the developer side). It must be noted that this list of criteria is not exhaustive. Table 3 summarizes the proposed evaluation criteria.

2.6.2.2 Development Platform Criteria

In this section we describe and analyse the introduced development platform evaluation criteria in relation with their data type.

Development Tools Availability {*Yes, Partial, No***}**. Refers to the availability of the necessary tools for application development. The existence of public and free development tools makes the malware development easier and cost effective. The reason for this is that a free emulator reduces the development cost of an attacker, since a device purchase is not necessary. Moreover, the SDK contains all the tools (e.g. debuggers, compilers, etc.) that are necessary for the implementation of the malicious application.

Table 3: Proposed Evaluation Criteria

Evaluation Criteria	*Type*
Development tools availability	String {Yes, Partial, No}
Development friendliness	Boolean
Installation vectors	String {Multiple, Restricted}
Application portability	Boolean
Application testing	Boolean
Application removal	Boolean
Unofficial repositories	Boolean
Distribution cost	Boolean
API restrictions	Boolean
Application signing	Boolean
Developer's background	Education Level
Development time	Number

Development Friendliness {*Yes, No***}.** This Boolean criterion assesses the "developer" friendliness of the programming language that is supported by the smartphone platform. The adoption of a well-known and widely used programming language (e.g. Java) is preferred during any application deployment.

Installation vectors {*Multiple, Restricted***}.** This criterion assesses the available installation options for an application on the smartphone device. These installation options include the use of removable media, through the Web, email, etc.

Application Portability {*Yes, No***}.** Refers to the ability of a malicious application to execute in different smartphone OS versions. The more compatible an application with different OS versions is, the greater the attack target population becomes.

Application Testing {*Yes, No***}.** This criterion refers to the possible application testing procedures, which can be used from application repositories to determine if the application is malicious. These tests usually take place before the application is added in the application repository.

Application Removal {*Yes, No***}.** This Boolean criterion refers to the existence of a remote application removal mechanism, commonly referred as 'app kill switch'. An application re-

moval from an application repository and smartphone devices takes place, when evidence is discovered that the application acts in a maliciously way.

Unofficial Repositories {*Yes, No***}.** This Boolean criterion refers to whether the security model permits the installation of applications from sources other than the official repository. If the application repository adopts application testing, one option for an attacker is to place the application in alternative sources, e.g. forums.

Distribution Cost {*Yes, No***}.** This criterion assesses whether the submission of an application into the application repository incurs costs to a potential attacker.

API Restrictions {*Yes, No***}.** This criterion refers to the restrictions imposed by smartphones' OS, in terms of how they control the use of protected API.

Application Signing {*Yes, No***}.** This criterion refers to any restrictions imposed by the smartphone OS, w.r.t. the signing of the applications before their installation.

2.6.2.3 Developer Criteria

This set of criteria includes the Developer's Background and the Development time. More specifically:

Developer's Background {*Education Level***}**. It refers to the developer's knowledge in information security, as well as to her programming skills. We assume that the amount of knowledge a developer possesses in information security and her programming skills, determine the sophistication of the attacks she is able to implement.

Development time {*Number***}**. It is used for determining the development effort that is needed to create the malicious application.

Apparently, the abovementioned criteria are given as an indication for the time and skills needed for the development of an attack by an average-skilled programmer. The evaluation criteria are summarized Table 3.

2.6.3 Implementation of a malware attack

This section provides a comparative evaluation of smartphones platforms in terms of malware development and distribution. Our analysis examines the feasibility and easiness of attacks implemented by average developers. Specifically, we examine the implementation feasibility of the simple attack scenario shown in Figure 1: Case study attack scenario.

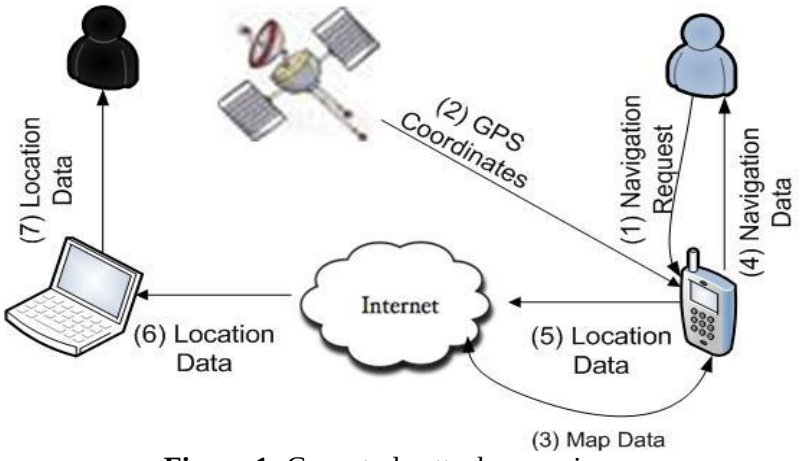


Figure 1: Case study attack scenario

The evaluation is based on: (a). the proposed qualitative evaluation criteria and (b). a proof of concept malware implementation study, in which the development of a location tracking application is examined. To evaluate the robustness and the security properties of the smartphones platforms, our overall approach focuses on the objective criteria (development platform), while at the same time takes into account the subjective criteria (regarding the developer). At this point it should be stressed that any sophisticated attack conducted by experienced attackers, as well as, publicly available malicious code used by script kiddies are not examined herein. Furthermore, a comparison with malware development in desktop computing is not examined in this chapter either. Also, the implementation study assumes that the security models are unmodified, i.e. the smartphones have not been 'jailbroken'.

The attack scenario refers to a malicious application that performs location tracking. The application collects a user's GPS coordinates (i.e. her exact position) and sends them to an attacker. It is assumed that the malicious functionality is included in a free GPS navigation application. The application apart from getting the user's location and presenting it using Google Maps, it also sends the location data to the attacker's server. The described malware is executed, in most cases, without creating any suspicion to a naive smartphone user. This is since the application's functionality requests (i.e. access to device location and Internet access to connect to the Internet) are consistent with the application's expected functionality.

We performed the development case study in our lab using two computers running a Windows XP and a Mac OS Leopard operating system. In the Windows machine we installed the emulators and the SDKs of all smartphone platforms, apart from Apple's iOS that was only compatible with Mac OS X.

For the malware implementation we selected two computer science students (one undergraduate and one postgraduate student, respectively) with basic information security background and moderate programming skills. Before the case study the students had successfully completed information security-related courses that are consistent with the Common Body of Knowledge described in [29]. The undergraduate student had completed a course on Infor-

mation Security Management and the postgraduate had completed the courses Information Security Management, Information System Auditing, Network Security, and Cryptography. Both students were more familiar with the Java programming language, since this was the language used in most implementations during undergraduate and postgraduate projects. The postgraduate student was only involved in the implementation of a smartphone platform only if the undergraduate was unable to implement it.

In the following paragraphs we analyse the results regarding the development and use of this malicious application to the smartphone platforms described in Section 2, namely[11]: Android, BlackBerry, iOS, Symbian, Windows Mobile and Windows Phone.

2.6.3.1 Android case study

The malware implementation was successfully developed on Android in one day. The official development toolkit (i.e. SDK and emulator) was used during development. The implementation was aided by: a). the adoption by the platform of a widely used programming language (i.e. Java), and b). the effective documentation of its API. In addition, the same source code successfully compiled and executed in Android versions 2.2 and 2.3; hence the application is considered portable within the Android platform at the time of the writing.

Regarding application distribution an attacker has many options, since Android does not impose any restriction neither on the application source (i.e. originating from an unofficial repository), nor on the installation vector (e.g. removable media, WWW, etc.). A small registration fee is required for application submission in the official repository, but it is inadequate to impede an attacker. Furthermore, application testing for malicious behaviour is not taking place if the official repository is selected as the distribution vector. Hence, it is likely that malware such as one described in this case study is currently present and downloaded by naive users from the repository[12].

As mentioned beforehand, the security model of the Android platform imposes some application restrictions concerning the application signing and API control. We argue that these restrictions provide only partial security protection. API requests are authorized by the –often not technical and security savvy- user during application installation. No further checks about application permissions take place after the installation. Hence, it is likely that the malicious application would be granted the requested permissions (i.e. access to location data and the Internet), especially in our case, where the permissions fully match the expected application's functionality. For the latter, the imposed signature can be self-signed by the developer and, as

[11] The conference paper [20] includes a case study implementation in Symbian. This paper replaces Symbian with Windows Phone, since Nokia (February 2011) decided to use Symbian only in feature phones and Windows Phone 7 in Nokia smartphones.

[12] The DroidDream incident [4] in the App Market was identified after the submission of our conference paper [20].

a result, the application's source origin is not verified and unknown. This situation, combined with the fact that an attacker may find valid credit cards numbers in the underground market, could be used to commence elite spoofing attacks. These attack scenarios are out of the scope of this chapter. Finally, Google's remote removal mechanism is the only efficient post installation protection mechanism against our case study scenario.

From the above analysis we infer that the likelihood of conducting such an attack on the Android platform is very high. In this context, Table 4 summarizes the results of our case study based on the criteria we have defined in section 4.1.

Table 4: Android case study results

Evaluation Criteria	*Android*
Development tools availability	✓
Development friendliness	✓
Installation vectors	multiple
Application portability	✓
Application testing	×
Application removal	✓
Unofficial repositories	✓
Distribution cost	×
API restrictions	✓
Application signing	✓
Developer's background	B.Sc.
Development time	0.5day

2.6.3.2 BlackBerry case study

Regarding our malware implementation on BlackBerry platform, the results were again successful. The malware implementation was carried out by the B.Sc. student. RIM's official development toolkit (i.e. SDK and emulator) was used for the implementation. The implementation was not demanding and its duration was one day, due to the adoption by the platform of a widely used programming language (i.e. Java), and the effective API documentation. Furthermore, the same source code successfully compiled and executed in versions 5 and 6 of BlackBerry, therefore the application is considered portable.

The security model of the BlackBerry does not impose any restrictions regarding the application origin. Nevertheless, the application file must be signed to access restricted and sensitive platform APIs. For the signing process the developer incurs a small key acquisition fee. As there is no strong authentication in the key acquisition process, application signing provides only integrity protection and poor source origin.

The cost for the cryptographic keys is considered inadequate to deter an attacker. On the other hand, the cost for the publication is expected to impede an attacker from publishing the application in the RIM's official repository, especially if she is in possession of limited eco-

nomic resources. The attacker, however, still has the option to submit her malicious application to an unofficial repository.

The security model of BlackBerry does not employ any application testing before adding the application in the official repository. Furthermore, there is no remote application removal mechanism for malicious applications. Hence, if the malware application is submitted in the official repository, then it is very likely to be downloaded and installed in BlackBerry devices by some users.

Conclusively, the development of the malware examined in this scenario is feasible and it demands little development effort. The only impediment is the cost of submitting the application to the official repository. Table 5 depicts the results of the case study regarding Blackberry OS.

Table 5: BlackBerry case study results

Evaluation Criteria	*BlackBerry*
Development tools availability	✓
Development friendliness	✓
Installation vectors	multiple
Application portability	✓
Application testing	×
Application removal	×
Unofficial repositories	✓
Distribution cost	✓
API restrictions	✓
Application signing	✓
Developer's background	B.Sc.
Development time	1 day

2.6.3.3 Symbian case study

Symbian OS provides basic functionality sufficient for application development providing the developer the option to self sign her application. Nevertheless, some compatibility issues exist, since the *location capability* -which controls access to API determining the location of the device - does not reside in the *basic capability* category in some Symbian OS versions (Nokia, 2011a).

For the deployment of the malware attack scenario only the *basic* capability category was used. Self-signed applications create a security warning at installation time that the user has to accept. Even so, the user would likely accept the installation of the application bypassing and ignoring the security warnings. Apart for signing the application, the security model of the platform does not restrict the application's distribution and as a result the attacker has many

distribution options (e.g. through an attacker-controlled application repository). Hence, the attacker has the option to avoid distribution cost and the Symbian Signed application testing. In addition, Symbian will only be able to revoke the self-signed certificate used in this case study, if Symbian becomes aware of the malware binary existence.

Table 6: Symbian case study results

Evaluation Criteria	*Symbian OS*
Development tools availability	✓
Development friendliness	×
Installation vectors	Multiple
Application portability	×
Application testing	×
Application removal	✓
Unofficial repositories	✓
Distribution cost	×
API restrictions	✓
Application signing	✓
Developer's background	M.Sc.
Development time	N/A

The attack implementation was not successfully completed in the Symbian platform. Even though the implementation was performed with the officially recommended development toolkits, the case study developers were unable to compile their code. In addition, even the sample applications provided by Symbian could not be compiled. The installation of the development toolkit was fully automated and the developers did not participate in its configuration. Hence, the possibility of toolkit misconfiguration is eliminated.

Furthermore, the developers faced other development obstacles during our case study, namely inadequate structure in the platform's API documentation (e.g. encountered "file not found" links), and the fact that they were not familiar with the platform's programming language. However, the obstacles reported in this subsection are of a minor importance to an experienced attacker; a case which is out of scope in this chapter. Hence, the above obstacles are likely to deter unmotivated attackers from developing malware attacks. The results of the case study on Symbian platform are presented in Table 6.

2.6.3.4 iOS case study

The malware implementation was successfully completed on iOS by the post graduate student. The implementation lasted 7 days and was tested on emulators running iOS 3 (version 3.1.2) and iOS 4 (version 4.1). The implementation duration was expected a priori to last more than in the other platforms due to no prior experience with Objective C. However, the toolset provided by Apple (i.e. SDK, documentation and emulator) minimized this lack of

experience. In Table 7 the relevant criterion was assigned the 'partial' value, since the toolkit is available only to Mac OS X users.

Application installation in devices running iOS is only possible via Apple's App Store. Hence, unofficial application repositories are not available for devices running an unmodified iOS version. As a result, a malware author must submit the application in the official repository. Application submission in the App Store requires a non free registration to Apple's development program. Prior to the application inclusion in the official repository it must be examined and signed by Apple. The details of application testing or 'application vetting' (e.g. automatic static malware analysis, manual inspection, etc.) are not known. The application testing criteria are, also, not publicly available, apart from the automatic rejection of any application that is using not official Apple API. Nonetheless, academic literature has identified the deficiencies of application testing as a proactive mechanism for application repositories [18].

The user has no control on the application's actions after the installation of an application in the device. In addition, the user is not informed when the application uploads data to a remote server. In our case study, the user would only be prompted to permit access to location data the first time the application executes. But, as the application is providing location based services the user is expected to confirm access to location data. Afterwards, the data would be transferred to the attacker's remote server, without the user noticing it.

Apple uses mechanisms allowing the remote deletion of malicious applications from iOS devices and the application repository. As in Android this is a post installation security mechanism, which can be used in case Apple or a user became suspicious of our malicious application. The results of the case study in iOS platform are depicted in Table 7.

Table 7: iOS case study results

Evaluation Criteria	*Apple's iOS*
Development tools availability	partial
Development friendliness	×
Installation vectors	restricted
Application portability	✓
Application testing	✓
Application removal	✓
Unofficial repositories	×
Distribution cost	✓
API restrictions	×
Application signing	✓
Developer's background	M.Sc.
Development time	7days

2.6.3.5 Windows Mobile 6 case study

The implementation of our location tracking application on a Windows Mobile device was

The malicious application implementation on a Windows Mobile device was also successfully completed by the student in 2 days. The programming language was C#. The reasons of the implementation effectiveness are the adoption by the platform of a programming language, namely C#, which resembles the programming rationale of Java and the effective API documentation.

The proof of concept malware was implemented for the versions 6.1 and 6.5 of Windows Mobile using the SDK provided by Microsoft. The installation package of the application was not signed. During the implementation the default configuration of the security model was preserved, in essence: (a). unsigned applications would be allowed to run, (b). the user would be prompted to authorize the application execution, and (c). if the application had been authorized by the user it would acquire full access to the OS system services.

The security model of Windows Mobile does not impose restrictions on the installation vector of applications. Hence, applications may be installed on devices even if they are downloaded from a source outside Microsoft's official repository. Hence, the attacker does not have application distribution costs. Furthermore, the application is not being tested for malicious behaviour, since it is not distributed by Microsoft distribution services. Nonetheless, the application removal mechanism, applied by Microsoft, may be used for the automated removal of the implemented malware.

To sum up, the feasibility of our malware attack in Windows Mobile depends on the device configurations regarding the security model, the user authorization at application installation time and the automated application removal security mechanism.

Table 8 summarizes the results, w.r.t. Windows Mobile platforms.

Table 8: Windows Mobile case study results

Evaluation Criteria	*Windows Mobile*
Development tools availability	✓
Development friendliness	✓
Installation vectors	multiple
Application portability	✓
Application testing	×
Application removal	✓
Unofficial repositories	✓
Distribution cost	×
API restrictions	×
Application signing	×
Developer's background	B.Sc.
Development time	2days

2.6.3.6 Windows Phone 7 Case study

The malware implementation was also successful in Windows Phone and was conducted by the undergraduate student. The implementation duration was one day, due to: a) the student's prior experience with the C# programming language, which was acquired during the implementation in Windows Phone and b) the effective API documentation provided by Microsoft. The official development toolkits (i.e. debuggers, SDK, etc.) were used and the implementation was tested in the version 7.1 of the official Windows Phone emulator. However, the malicious application is not backwards compatible with version 7.0 of Windows Phone [32].

As discussed beforehand the security model of Windows Phone employs different security mechanism decreasing the attacker choices regarding malware distribution. Each application must be signed by Microsoft in order to be installed on a device and it can only reside in the official application repository. As a result, applications cannot be installed from other locations and, hence, the attacker must pay a registration fee and be authenticated in the repository. In addition, the application before being indexed in the application repository, undergoes security testing. Thus - as in iOS - the attacker must circumvent the application testing mechanism in order to spread malware in the official repository, e.g. by encrypting parts of the application.

As in Android, the security model restricts application access to sensitive resources via capabilities. Contrarily to Android this list of capabilities are not prompted for authorization during application installation, but are only indexed in the application's web page in the application repository. As a result, a user has the option to reject application capabilities in an all-or-nothing approach by deciding not to install the application. Finally, the security model of Windows Phone includes a remote 'app kill switch' for a posteriori removal of malicious applications from Windows Phone devices. The aforementioned are summarized in Table 9.

Table 9: Windows Phone case study results

Evaluation Criteria	*Windows Phone*
Development tools availability	✓
Development friendliness	✓
Installation vectors	restricted
Application portability	×
Application testing	✓
Application removal	✓
Unofficial repositories	×
Distribution cost	✓
API restrictions	✓
Application signing	✓
Developer's background	B.Sc.
Development time	1day

2.7 Discussion

The security model of a smartphone operating system has two contradicting goals. On the one hand, it must provide users with security assurance concerning the execution of third-party applications on their devices. On the other hand, it must provide to the developers a secure ecosystem, where consumers are willing to install new applications, and it is easy and efficient to implement new applications.

The proof of concept study demonstrated that, under certain circumstances, the security model of all available smartphone platforms would not counter a location tracking attack. Moreover, it showed that it is possible to easily implement this malicious application on all the examined smartphone platforms. It should be stressed herein that until the writing of this chapter, the current antivirus software is inefficient, since: a) as every third party smartphone application, they execute in a sandboxed environment and, thus, are not able to acquire 'root' privileges, e.g. for placing API hooks and b) execute in a resource restrained device.

Our case study implementation was efficiently and effectively completed in all platforms except for Symbian, by using the official development tools, and, it was tested on the official emulators. The reasons of the implementation failure on Symbian were not security related. They were related with the developer's programming skills and Symbian's unstructured API documentation. The latter might have been the reason why Symbian's app repository slow development, which eventually lead to Symbian's discontinue. (I must add in the beginning how the apps have changed the smartphone ecosystem)

Moreover, the implementation of the malicious applications was conducted by average programmers with the use of the API documentation. Specifically, almost all malware was implemented by the undergraduate student. This fact is a serious indication of how malicious software may evolve in smartphones. In fact, the 'DroidDream incident' [4] in the Google Play (Android Market) was identified after the submission of our conference paper [20], whereas nowadays Android is one of the most targeted platforms by malware (references XXX).

Security testing of applications cannot be avoided only on Apple's iOS and Windows Phone. Furthermore, these were the only platforms having strict application installation requirements, i.e. only via the official application repository.

Table 10: Case study results overview

Evaluation Criteria	***Android***	***BlackBerry***	***Symbian***	***iOS***	***Windows Mobile***	***Windows Phone***
Development tools availability	✓	✓	✓	partial	✓	✓
Development friendliness	✓	✓	×	×	✓	✓

Installation vectors	multiple	multiple	multiple	restricted	multiple	restricted
Application portability	✓	✓	×	✓	✓	×
Application testing	×	×	×	✓	×	✓
Application removal	✓	×	✓	✓	✓	✓
Unofficial repositories	✓	✓	✓	×	✓	×
Distribution cost	×	✓	×	✓	×	✓
API restrictions	✓	✓	✓	×	×	✓
Application signing	✓	✓	✓	✓	×	✓
Developer's background	B.Sc.	B.Sc.	M.Sc.	M.Sc.	B.Sc.	B.Sc.
Development time	0.5day	1 day	N/A	7days	2days	1day

Among the examined platforms only Windows Mobile allowed, under some security model configurations, the execution of unsigned applications. Yet, the digital signing process in the rest platforms, which were examined, provides different security assurance to a user. The user was found, on the one hand, as not having any control on the API running in the device on some platforms. On the other hand, the user is fully responsible for authorizing application requests for access to sensitive resources in other smartphone platforms. The latter is a major weakness in the security model of these smartphones platforms, since the user's security knowledge and awareness is often insufficient.

Android and Windows Mobile were the only smartphone platforms where an attacker could avoid application distribution costs and only BlackBerry did not use a remote application removal mechanism. Table 10 summarizes our findings.

According to the case study findings, a non-sophisticated attacker would try to avoid iOS and Windows Phone as a privacy and security attack vector, since they are the platforms having the most defensive mechanism in place (i.e. application testing, controlled application installation vectors and remote application removal). Moreover, iOS was also found complex in (malicious) application development. An attacker is expected to prefer one of the rest platforms, especially Android and Windows Mobile, which were found to provide the least protection mechanisms. Nonetheless, it should be stressed that the decision for platform selection, is dependent on whether the attacker is preparing a targeted attack (i.e. against an individual), or not. In the former case the attacker will select the OS running on the individual's device, whereas in the latter case the attacker will chose the platform with the larger user base (i.e. Android during the writing of this dissertation).

2.8 Survey of Application Management Approaches

From the above-mentioned smartphone security schemes, it is obvious that a reliable security scheme must include an *Application Management System (AMS)* providing a managed application repository. The AMS must impede malicious applications from entering the re-

pository and be able to authenticate their developers. Therefore, the AMS must include secure and robust procedures for developer registration and application submission. In this context an AMS must at least include mechanisms that provide: (a). Application Integrity, (b). Application Testing, (c). Remote Application Removal, (d). Application Testing Documentation, and (e). Developer Strong Authentication. These mechanisms are described and analyzed in the following paragraphs.

Application Integrity ensures that an application's binary is not altered, e.g. by malicious code injection in pirated versions of the application. As mentioned previously, in all smartphone platforms - apart from the Android platform - the security scheme mandates application digital signing with a certificate controlled by the platform. On the contrary, Android allows users to sign applications with custom self-signed certificates not validated by a TTP. As a result, a malicious developer may download and repack an application with a new certificate and submit it to the Android Market or in an alternative application repository. Apart from monetary loss to the original application's developer, a rogue developer can infect the application with malware, compromising the security of Android devices (Sophos 2011).

Application Testing employs static and/or dynamic binary analysis to ensure application functionality reliability, official API usage and rational resource consumption. It typically contains tests for copyright infringements and, in some platforms, security testing (NOKIA 2011), (Microsoft 2010b). In a managed application repository, we argue that the security testing process should be mandatory. This will ensure that malware cannot easily be spread through the application repository. Only Symbian, iOS and Windows Phone platforms mandate application testing before submission in their repositories, whereas it is unclear if Apple's iOS employs security testing procedures.

Remote Application Removal also referred as "application remote kill switch" ensures that a malicious application will stop being executed in smartphone devices, if it has not been detected during application testing. The security scheme must ensure that (a) the mechanism will not be used for application censorship and (b) that it is conformant with legislation protecting access to a user's device. Among the surveyed smartphone platforms, only Symbian and BlackBerry do not use a documented remote application removal mechanism.

Application Testing Documentation on the one hand mandates developers into submitting applications that satisfy strict requirements and, on the other hand, informs smartphone users about these testing criteria before application acceptance. From the surveyed platforms only Symbian and Windows Phone document application tests.

Table 11: Current Application Management Approaches

Management Functionality	Android OS	BlackBerry OS	Symbian OS	iOS	Windows Phone
Application Integrity	×	✓	✓	✓	✓
Application Testing	×	×	✓	✓	✓
Remote Application Removal	✓	×	×	✓	✓
Application Testing Documentation	×	×	✓	×	✓

The adoption of the management functionalities by the surveyed security schemes is summarized in
Table 11 above.

2.8.1 Discussion

The proposed scheme includes mechanisms that satisfy the security requirements which were not provided by the surveyed smartphone security schemes, namely: (a). application integrity, (b). application testing, (c). remote application removal, (d). application testing documentation, and (e). developer strong authentication. The scheme is cross-platform, since its definition is not dependent on any smartphone platform implementation details. Moreover, its security mechanisms must be fully documented and extensible. The scheme deters malicious developers from submitting malware in the repository, as well as users from misusing the repository's resources, by giving penalties to miscreant activities. It may optionally deter malicious developers from entering the application repository, by using qualified certificates and imposing the digital signing of a CMA, or an equivalent, statement during developer registration.

The proposed scheme does not focus on the application testing criteria. It considers application testing as a black box containing the state of the art of application testing such as (NOKIA 2011), (Microsoft 2010b). Nonetheless, the application testing mechanisms and criteria in the scheme's implementation must be documented and carefully selected to avoid performance bottlenecks in this component. Furthermore, the proposed scheme is based on PKI certificates verified by TTP providing strong authentication. The scheme employs authentication only during developer registration, to avoid delays in user registration that could deter user access to the ARSS. The developer incurs a certificate creation cost, for the acquisition of a developer certificate, which is equal to the developer registration cost in current official application repositories. Hence, the enrolment cost must be carefully selected in the scheme implementation so as to not deter developers from enrolling to the repository.

2.9 Conclusions

Smartphone devices are multi-purpose portable devices enclosing multiple heterogeneous third party applications, which augment the device's functionality. The smartphone security models facilitate mechanisms and processes controlling the installation and execution of third party applications. Even so, the efficiency of the adopted security mechanisms seems to be controversial. Their ability to protect the devices from becoming a privacy attack vector from average developers, such as undergraduate and postgraduate computer science students, is proven to be unclear.

Our chapter (a). proposes evaluation criteria assessing the protection from simple smartphone malware, and (b). provides a comparative case study analysis, where a proof of concept implementation of a location tracking malicious application is attempted in well-known smartphone platforms (i.e. Android, BlackBerry, iOS, Symbian, Windows Mobile, Windows Phone).

Our proof of concept study has proven that, under circumstances, all examined smartphone platforms would not stop average developers from using smartphones as privacy attack vectors, harvesting data from the device without the user's knowledge and consent. It also showed the easiness of malware development by average programmers that have access to the official tools and programming libraries provided by smartphone platforms.

A silver bullet solution against similar attack scenarios is not available. Some of the solutions that can be used to avoid a potential malware outbreak in smartphones are: (a) user awareness, i.e. informing user about the security and privacy risks in smartphone platforms, and (b) providing secure application distribution in smartphone application repositories.

For further work we plan to extend the evaluation criteria and attribute weights to them. We also plan to repeat the case study with more developers to acquire more generalisable results.

Chapter 3: Mitigation of web threats

"*If you 're not paying for something, you 're not the consumer, you 're the product being sold*" – Andrew Lewis

3.1 Introduction

The proliferation of smartphones has introduced new challenges in secure web browsing. These devices often have limited resources, as well as mall size, which may limit the security 'arsenal' of their users. Such lack of protection controls, however, does not seem to hinder users from browsing the web via smartphones. On the contrary, according to a recent report (CISCOb), by 2017 smartphone mobile data traffic will increase 81%, comparing to 2012. The same report predicts that smartphones will be responsible for the 67.5% of mobile traffic growth in 2017.

Average users –i.e. not security and/or technical savvy ones – are not familiar with the details of security controls, which are used while browsing the web. For instance, a user may understand that SSL offers a level of protection to her online transactions. It is rather unlikely though, that she is aware of the relevant security details (e.g. cryptography, server authentication, etc.) and threats she is exposed to (e.g. eavesdropping, session hijacking, etc.).

Nowadays, users come across to different threats while browsing the web. These range from traditional client-side attacks (e.g. malicious files, Cross-Site-Scripting (XSS), etc.), up to recent zero-day exploits that target Java plugins[13]. Contrary to what one would expect, CISCO(a) reports that browser malware are not only present in 'bad' webpages (e.g. adult websites, ones hosting pirated software, gambling, etc.), but also in benign ones (e.g. social media websites, search engines). The latter may unwittingly serve malware embedded in their active content, typically after a server compromise or with the inclusion of malicious advertisements. Furthermore, progressively more attackers use in their client-side attacks browser exploitation frameworks (e.g. Blackhole exploit kit, Phoenix, etc.), which are available in the underground market (SERT).

Web browsers (hereinafter referred to as browsers) communicate security events to users through their Graphical User Interfaces (GUIs). For instance, the padlock icon appears every time a user visits a website with a valid digital certificate. Moreover, browsers include window gadgets (widgets), such as checkboxes, buttons, etc., for the configuration of their securi-

[13] http://www.reuters.com/article/2013/01/11/us-java-security-idUSBRE90A0S320130111

ty controls. Users are expected to configure the browser's security controls as they see fit (by interacting with its menu), so as to protect their security and privacy. To aid users in this task, every web browser contains a menu option focused on the configuration of security and/or privacy controls. Even though average users generally tend to ignore security events (Egelman et al., 2008; Motiee et al., 2010; Mylonas et al., 2013; Sunshine et al., 2009), some of them have been trained to interact with the above interfaces in desktop browsers towards a safe web browsing.

In this context, this work contributes by providing a systematic and comprehensive analysis of web browser security controls. In particular, we focus on popular browsers in smartphones and desktops, enumerate their security controls, and collect and compare their default settings as well as their manageability options. Then, we provide a comparative evaluation of the offered protection against web threats. Specifically, our goal is to examine the following research questions regarding the security controls that are provided by web browsers, namely:

- What protection against web threats is offered by the preconfigured security settings in browsers?
- What is the manageability options are provided by the security controls that protect from certain web threats?

The former provides indications of the offered protection to average users. The latter reveals the manageability of countermeasures for each threat, i.e. the flexibility to adjust the offered protection according to the users' "risk appetite" (e.g. a user may be willing to receive targeted advertising). Our work summarizes the differences in the availability and manageability of browsers' security controls. Overall, as expected, desktop browsers provide an increased manageability and availability. Regarding protection against web threats, our analysis revealed that browsers by default focus mostly on a subset of the examined threats (e.g. malware, privacy breach, phishing), while offering poor protection against the rest (e.g. third-party tracking, browser fingerprinting). Our key findings can be summarized as: (These may need to be revised accordingly)

- **Limited security and manageability in smartphone browsers**. Our analysis revealed that smartphone browsers provide a subset of security controls and limited manageability over them, compared to their desktop counterparts.
- **Smartphone users are unprotected from rogue websites**. Our evaluation revealed that smartphone browsers do not protect users from websites that host malware and/or phishing scams. Contrarily, desktop browsers include mechanisms, such as Google's *Safe Browsing* technology, *Smartscreen* technology, etc. (Google Developers; Microsoft, Opera), that filter out rogue websites.

- **Users are exposed to third-party tracking**. Users are exposed to third-party tracking/profiling in all examined browsers. This holds true, since by default browsers mechanisms that protect users from this tracking are disabled.

The remainder of this chapter is organized as follows. Section 2 presents related work and Section 3 a suggested threat model. Section 4 provides the reader with the methodology of our research. Section 5 includes our empirical and experimental observations. Finally, Section 6 includes a discussion of the results and our conclusions.

3.2 Related work

To the best of our knowledge, we are the first to perform a systematic and comprehensive evaluation of the security controls that web browsers provide to their users. Our work closely relates to (Botha et al., 2009), which provides a simple comparison of the availability of security options that exist between Internet Explorer 7 (for Windows XP) and Internet Explorer Mobile (for Windows Mobile 6 Professional Edition). Our work examines the availability of the security controls in current web browsers, as well as their manageability and the preconfigured protection against web threats. A part of our work closely relates with (Amrutkar et al., 2012), where the researchers have focused on the visibility of security indicators in smartphones. We have confirmed the findings of their work, regarding the way smartphone browsers handle digital certificates. We were able to find an additional vulnerability in the way smartphone and desktop browsers handle invalid digital certificates.

Recent literature on web security has focused on the protocols for secure connections (i.e. SSL/TLS). It has exposed vulnerabilities in the protocol itself (AlFardan and Paterson, 2013; Panday, 2011; Paterson et al., 2011; Paterson), as well as in the way the protocol is implemented in devices (Fahl et al., 2012; Georgiev et al., 2012)). Literature has also focused on the visibility of security indicators in desktop browsers, mostly focusing on invalid digital certificates, indicating the majority of users tend to ignore them (Amrutkar et al., 2012; Egelman et al., 2008; Fahl et al., 2012; Schechter et al., 2007; Shin and Lopes, 2011;Sunshine et al., 2009).

The extra functionality that is offered by the capability of browsers to execute code in the client side has been exploited by attackers via JavaScript malware (Johns, 2008). In response to the threat of malicious content, browser security literature has focused on the proposition of novel browser security architectures. The proposed architectures either extend a browser architecture with new components that offer enhanced security (Amrutkar and Traynor, 2012; Barth et al., 2010; Carlini et al, 2012; Chen et al, 2011; Meyerovich and Livshits; 2010), or provide new browser architectures (De Groef et al., 2012; Grier et al., 2008; Jang et al., 2012; Wang et al., 2009). Researchers have also proposed detection mechanisms tailored for JavaS-

cript malware, using either static (Canali et al, 2011; Cova et al, 2010; Curtsinger et al., 2011) or dynamic analysis (Jang et al, 2010; Kolbitsch et al., 2012; Saxena et al, 2010) techniques.

3.3 Threat model

This threat model extends the one that is described in §2. The chapter assumes *average users* (i.e. ones that are not security and technical savvy), who have been trained, via the browser's support pages, to adjust its settings through its GUI (i.e. menu options, buttons, etc.) for a safe web browsing. Thus, it is assumed that some smartphone users have been trained to change the default values of security and privacy controls as they see fit, so as to adjust the level of protection that is provided by the browser. Finally, in accordance with the initial threat model, it is assumed that: (a) the user has not altered the operating system of her smartphone (rooting, jailbreaking) and (b) the browser's security mechanisms are unmodified, i.e. the user has not installed any extension/add-on that adds a security mechanism to the browser (e.g. NoScript, AdBlockPlus). This type of user is referred to as *Alice*.

The extended threat model includes three types of attackers. Firstly, it includes an attacker who has unauthorized control over the network. Such an entity, uses active attacks (e.g. ARP spoofing) to conduct Man in the Middle (MiM) attacks and is referred to as *Eve*.

The second type of attacker has control of malicious web servers in the Internet. The servers are either used to distribute malware (e.g. by exploiting vulnerabilities in plugins, add-ons or the browser itself), or to conduct fraud attacks (i.e. phishing). This type of attacker is referred to as *Mallory*.

The last attacker type controls advertising services used for 'malicious' user tracking and is referred to as *Gorwel*. More specifically, *Gorwel* uses advanced user tracking mechanisms (e.g. non-persistent tracking data (Eckersley, 2010)), aiming at user profiling and/or user identification. The use of such malicious user tracking constitutes an intrusion of Alice's fundamental right to privacy (Commission of the European Communities, 2006). It is worth noting that using an advertising service on the web is not a malicious act per se.

3.4 Methodology

Our research goal is to evaluate whether the security controls that are available in a smartphone browsers provide Alice with similar manageability as in their desktop counterparts. In addition, we examine if smartphone browsers offer a similar level of protection against the aforementioned three attackers.

The scope of our evaluation includes the popular web browsers, i.e. Google Chrome, Mozilla Firefox, Internet Explorer, Opera, and Apple Safari (StatCounter), as well as their smartphone counterparts. More specifically, we installed the latest versions of desktop brows-

ers (as of June 2013, i.e., Chrome (v. 27), Firefox (v. 21), Internet Explorer 10, Opera (v. 12.15), and Safari (v. 5.1.7)) in two desktops. Windows 7 and Windows XP were selected for the installation of the aforementioned browsers due to their popularity in the desktop platform (78% of global market share (StatCounter)).

Contrary to desktops, the above browsers are not available in all smartphones. Table 12 summarizes the smartphones that were used in the evaluation, as well as the availability of different third-party browsers in them. The evaluation includes devices with Android, iOS and Windows Phone, which constitute the 96.5% of the smartphone market share (Gupta et al., 2013). Furthermore, it includes devices with the following Android versions: Gingerbread (v. 2.3), Ice Cream Sandwich (ICS, v. 4.0.*), and Jelly Bean (JB, v. 4.1.2); which constitute the 91% of the in use Android devices (Google). Therefore, our evaluation can be regarded as representative both in the desktop and smartphones platforms. Finally, for readability and space reasons, Table 12 refers to the stock browsers of Android, iOS and Windows Phone (i.e. Browser, Safari, and IE Mobile respectively) as 'stock browser'.

Table 12. Browser availability in the smartphones that were used in the evaluation.

Platform	Version	Device	Chrome Mobile (v. 26)	Firefox Mobile (v.21)	Opera Mobile (v. 14)	Opera Mini (v. 10)	Stock Browser†
Android	2.3.5	HTC Explorer			X		X
	2.3.6	LG-E400			X		X
	4.0.3	LG - P700	X	X	X		X
	4.0.4	Sony Xperia	X	X	X	X	X
	4.1.2	Samsung Galaxy S3	X	X	X		X
		Samsung Nexus S	X	X	X		X
iOS	5.1.1	iPhone 4	X			X	X
	6.1.2	iPhone 4S	X			X	X
Windows Phone	7.5	HTC Trophy7					X

† Browser for Android, Safari for iOS and IE Mobile for Windows Phone

Initially, all the available support pages in each browser that are dedicated for security and privacy were enumerated, since Alice is expected to use this material in order to be trained to configure the browser controls. Then, the graphical interfaces in desktop and smartphone browsers were enumerated and all the available configurable security controls, as well as their default values were collected. Any confusing text labels that may exist were marked, as well as any widgets that had obvious usability problems. The controls were grouped together into five categories according to their intended use from the support pages, namely: (a) content

controls, (b) privacy controls, (c) browser manageability, (d) third-party software controls, and (e) web browsing controls.

The next section presents our results regarding the manageability of web browsers' security controls.

3.5 Availability and manageability of security controls

Overall, thirty three (33) security controls appear in the browsers' interfaces, which are listed herein. The majority of the controls' labels are self-explanatory (e.g. block JavaScript). The rest of them are briefly described here, namely: (a) external plugin check refers to the existence of a web service that analyses the browser's plugins for vulnerabilities (e.g. Mozilla, Qualys) (b) local blacklist enables users to enforce controls on a per-site basis via local blacklist/whitelist (e.g. per-site cookie blocking), (c) under master password the browser requests the entry of a master password every time it restarts, before accessing any stored passwords, and (d) website checking enables a user to manually initiate analysis (for malware/phishing) on the web site she visits.

Tables 11-15 summarize the availability and manageability status of all the security controls that are available via the browsers' interfaces. Their availability and manageability differs between each browser as well as between the two versions, i.e. desktop and smartphone, of the same browser. The findings are grouped together into five categories, according to the controls intended use from the support pages, namely: (a) content controls, (b) privacy controls, (c) browser manageability, (d) third-party software controls, and (e) web browsing controls.

Tables 11-15 use the following notation: (i) ☒ is used when *the mechanism is not supported*, (ii) □ is used when the mechanism *is supported but not configurable*, (iii) ▣ is used when the mechanism *is supported but not easily configurable*, and (iv) ■ is used when *the mechanism is supported and easily configurable*. A security control is marked as 'not easily configurable' when it can only be configured from a hidden menu (e.g. about:config, see Appendix), or when there is a usability problem in the configuration of the control (e.g. confusing wording of the widget's label). In such cases, it is rather unlikely that users will be able to find and/or correctly configure it.

Regarding the default values of security controls, ● and ○ stand for *default enabled* and *default disabled* control, respectively. The following notation is also used: {GC=Chrome, MF=Firefox, IE=Internet Explorer, OP=Opera, AS=Safari; AB= Android's stock browser, CM=Chrome Mobile, FM= Firefox Mobile, IM=IE Mobile, OM =Opera Mobile, Om=Opera Mini, SM=Safari Mobile}. Finally, the stock browser of Android is referred to as 'ABrowser'.

3.5.1 Content controls

Table 13 summarizes the manageability of content controls, i.e. controls that enable Alice to block cookies, images and pop-ups. All browsers enable Alice to block cookies. Safari desktop and mobile were the only browsers that allowed, by default, only first-party cookies (c.f. **Table 14**). First-party cookies, i.e. those that are not created from a third-party domain, are normally used by web servers for user authentication and their blocking might cause disruptions in web applications' functionality.

By-default, all browsers present website images. Otherwise, a serious usability problem would arise in the webpages that they visit. Users may wish to block images for various reasons, such as to protect their privacy (Zeigler et al.), to speed up their browsing, etc. Contrary to desktop browsers, where Alice can block images in all browsers, this option is not available in most smartphone browsers (c.f. **Table 13**). This holds true, since only ABrowser and Opera Mini, provide a widget to enable this control and in Firefox Mobile this control is only available in a hidden menu (c.f. Appendix).

Similarly, as summarized in **Table 13** all browsers block pop-up windows by default. Firefox Mobile allows the configuration of the pop-up blocking mechanism from a hidden menu interface (see Appendix for hidden menus). IE Mobile and Opera Mini block pop-ups by default without allowing Alice to disable this control. This fact, however, may break the functionality of web applications that use benign pop-ups (e.g. a pop-up shown to upload a resume).

Table 13. Manageability of content controls.

Controls	GC	MF	IE	OP	AS	AB	CM	FM	IM	OM	Om	SM
Block cookies	■○	■○	■○	■○	■○	■○	■○	■○	■○	■○	■○	■○
Block images	■○	■○	■○	■○	■○	■○	☒	◙○	☒	☒	■○	☒
Block pop-ups	■●	■●	■●	■●	■●	■●	■●	◙●	□●	■●	□●	■●

3.5.2 Privacy controls

Blocking location data- either geolocation data in desktops, or via the smartphone's location provider – is configurable in most browsers (c.f. Table 14). Only Chrome and Safari block them by default, i.e. they prompt users before accessing location data. Safari Mobile follows a similar approach, but the control of location services is not configurable until such a request is made by Safari Mobile for the first time. In such a case, the user is prompted and access to location data is subsequently manageable from the settings of location data, not from those of Safari Mobile. Therefore this hinders Alice from finding the control, since it resides in a different configuration menu. Alice is also expected to have difficulties in configuring this mechanism in Firefox versions (i.e. desktop and mobile), since it is configured only from a hidden menu. Finally, our evaluation revealed that the availability of this control is

different in iOS and Android, being unavailable in the former and available but default disabled in the latter.

By default, browsers send the referrer value in HTTP headers (it is misspelled as 'referer' in the header), a value that can be collected from Gorwel for user tracking (Fielding et al., 1999). The analysis revealed that the security control that removes the referrer is unavailable in most smartphone browsers and in the desktop version of Internet Explorer and Safari (c.f. Table 14). Both versions of Firefox allow Alice to manage this control only via a hidden menu. Finally, enabling this control in Chrome is rather difficult, since it involves starting its executable with a parameter via the terminal (see Appendix).

Regarding third-party cookies, the majority of desktop browsers permit them (except for Safari). In smartphones the majority of browsers accept all cookies in an all-or-nothing approach, thus failing to protect Alice's privacy. This holds true, since they either block both first-party and third-party cookies, or allow them (c.f. Table 14). Only Firefox Mobile and Safari Mobile provide manageability over third-party cookies, while having the same default values as their desktop counterparts. One could argue that enabling tracking by default is acceptable, since in the majority of the examined browsers the user is allowed to block it. Nevertheless, it is unclear whether Alice can understand the impact of tracking (Madrigal), which, in its ultimate form (e.g. via user identification (Eckersley, 2010)) may constitute an intrusion of her fundamental right to privacy. Furthermore, during browser installation Alice is not explicitly asked whether she wishes to receive personalized advertisements.

As summarized in Table 14, by default, only Internet Explorer sends the do-not-track (DNT) preference, i.e. the value "DNT: 1" in the HTTP header (Zeigler et al.). In smartphones only a subset of browsers contain settings for DNT, namely: Chrome Mobile, Firefox Mobile and Safari Mobile. Moreover, DNT in Safari is available only in iOS 6 and the wording near the widget is confusing, i.e. "Limit ad tracking". Therefore, it is likely that Alice will accidentally enable web tracking by selecting the option "off", believing that she is disabling ad tracking in this way.

Most browsers provide a History Manager. ABrowser, Opera mini and Safari Mobile allow a user to delete history data, but the relevant widgets are scattered in the browsers' interfaces. Safari for desktop provides the widget under a widget with title "Reset Safari…", thus making it rather difficult for a normal user to find it. Moreover, Alice may initiate private browsing - i.e. a session where browsing data (cookies, browsing data, downloads) are not stored locally - in all desktop browsers. Contrary to desktops, smartphone browsers support private browsing only in ABrowser, Firefox Mobile, Safari Mobile and Chrome for Android. ABrowser does not offer private browsing in Android Gingerbread at all, while the newest Android versions do offer this mechanism only from a hidden menu (see Appendix). Finally,

this control is offered by Chrome Mobile for iOS, but its effectiveness is hampered by the platform's limitations[14].

Table 14. Manageability of privacy controls

Controls	GC	MF	IE	OP	AS	AB	CM	FM	IM	OM	Om	SM
Block location data	■●	▣○	■○	■○	■●	■○	☒|■○†	▣○	■○	☒	☒	▣●
Block referrer	▣○	▣○	☒	■○	☒	☒	☒	▣○	☒	☒	☒	☒
Block third-party cookies	■○	■○	■○	■○	■●	☒	☒	■○	☒	☒	☒	■●
Enable DNT	■○	■○	■●	■○	▣○	☒	■○	■○	☒	☒	☒	☒|▣○†
History Manager	■	■	■	■	▣	▣	■	■	□	■	▣	▣
Private browsing	■	■	■	■	■	☒|▣	■†	■	☒	☒	☒	■

† heterogeneity in different platforms.

3.5.3 **Browser manageability**

Browser updates protect Alice from security vulnerabilities and bugs. All desktop browsers support automatic installation of browser updates, except for Safari. Safari for Windows - as well as Mac OS Leopard - does not offer update support for the browser anymore, thus exposing its users to more than 100 vulnerabilities that were patched in Safari 6 (Apple). Alice is not aware that Apple does not provide updates for her browser, since she is not explicitly informed in the browser's download page or during/after its installation. Thus, she can only be informed that she is vulnerable by a third source (e.g. forum, blog, etc.), which may eventually make her decide to switch to an alternative desktop browser in order to stay secure.

Browser updates in most smartphones are, contrary to desktop browsers, semi-automatic (c.f. Table 15). Stock browsers (e.g. ABrowser, Safari Mobile) update with platform updates and third-party browsers update via the application repository (e.g. Google Play, App Store, etc). In both cases, the update requires the user's initiation. Only Firefox Mobile can be configured via a menu option to be updated automatically. Finally, it is worth noting that browser updates in smartphones often suffer from delays. The updates of third-party browsers may be delayed by the app analysis process of the app repository. Also, updates of Android may be either delayed or even be unavailable by the device vendor. Therefore, in some cases users of ABrowser may not get updates even if they are officially available from Google.

Among desktop browsers, only Safari did not offer a configurable certificate manager,[15] i.e. an interface where Alice can either inspect or edit the certificates that are trusted or blocked by the browser. The ability to manage trusted certificates is important, especially in the case that a Certification Authority (CA) becomes compromised (e.g. in Network Computing). In this case, Alice must be able to disable this CA. In smartphones, the evaluation showed that

[14] http://support.google.com/chrome/bin/answer.py?hl=en&answer=95464
[15] Safari uses Internet Explorer's certificate manager without providing a link to its interface.

most browsers (i.e., Safari Mobile, Chrome for iOS, ABrowser for Android Gingerbread, Firefox Mobile, IE Mobile, and Opera Mini) do not offer a configurable certificate manager. ABrowser, Opera Mobile and Chrome for Android use the certificate manager that is provided in the newest Android versions (i.e. the ones after Gingerbread version).

All browsers offer auto-fill functionality, i.e. the browser can remember passwords of certain websites. As shown in Table 15, only a subset of the desktop browsers offers password protection via a master password – i.e. the browser asks users to enter a master password every time it restarts, before accessing stored passwords. Furthermore, Chrome and Firefox were found to enable the unmasking of stored passwords. Therefore, it is very easy for an attacker who has temporary access to Alice's browser to access her passwords. Among smartphone browsers, only Firefox Mobile offers password protection with a master password. Even though smartphone browsers and most of the desktop browsers do not unmask passwords, an attacker with physical access to the browser can login to websites where Alice has enabled password auto-fill. The risk of this attack is greater in smartphones, due to their small size and mobility and the fact that in a recent smartphone awareness study only 64.4% of the respondents password-protected their device (Mylonas et al., 2013).

All desktop browsers provide an interface to configure a proxy server[16], which can provide Alice with anonymity (e.g. via a free proxy or onion network) and enhanced security, if the proxy implements a malware and/or phishing detection engine. Most smartphones can be configured to use a proxy server via the device's Wi-Fi settings, but it is rather difficult for Alice to find the configuration widget. This holds true, since the navigation to this configuration widget clearly violates the three-click rule (c.f. Appendix). Furthermore, the evaluation showed that Alice cannot enable a proxy server in any smartphone, when the device uses cellular connectivity (e.g. UMTS (3G), HSDPA, etc.). Thus, the aforementioned protection options that are offered by a proxy server are unavailable when Alice uses mobile Internet.

Alice may wish to configure her browser to use a search engine that does not track her (e.g. DuckDuckGo, Startpage).[17] As summarized in Table 15, this option is only available in four desktop browsers. The rest browsers either allow the selection of a search engine provider from a static list (e.g. Google, Bing, Yahoo), or do not offer such a selection, at all.

Among the examined browsers, only Internet Explorer and Opera enable Alice to inspect and select which protocol version of SLL and TLS is used in her secure connections. Various vulnerabilities have been discovered in SSL/TLS (e.g. BEAST, CRIME, Lucky 13) and recently new vulnerabilities have been discovered in TLS as well as in the SSL implementation that uses the RC4 algorithm (AlFardan and Paterson, 2013; Panday, 2011; Paterson et al., 2011; Paterson). Currently, the above protocol vulnerabilities in browsers and servers are

[16] Chrome and Safari use a link to the interface implemented by Internet Explorer.
[17] https://duckduckgo.com/, https://startpage.com/

fixed via workarounds that patch certain instances of the vulnerabilities. Furthermore, the adoption of the latest TLS protocol (i.e. TLS 1.2) may either hinder user experience, if the server does not support it, or it may be skipped during the protocol negotiation between the server and browser. For instance, during a MiM attack an attacker may convince the two communicating parties to switch to a vulnerable version of the protocol. Finally, the current interfaces in Internet Explorer and Opera browser allow Alice to disable or select an older version of SSL\TLS protocol, as any other non-security related setting in the browser's menu (e.g. enabling the automatic resizing of images).

Our results revealed that only Chrome offers a task manager. Even though the absence of this control does not imply that Alice is directly exposed to any threats, its presence can aid her to enhance control over web browsing by inspecting resource consumption (e.g. network, memory).

Table 15. Mechanisms for browser management.

Controls	GC	MF	IE	OP	AS	AB	CM	FM	IM	OM	Om	SM
Browser update	■●	■●	■●	■●	☒	□●	◙●	■●	□●	◙●	◙●	□●
Certificate manager	■	■	■	■	□	□/■[2]	■/□[2]	□	□	■	☒	□
Master Password	☒	■○	☒	■○	☒	☒	☒	■○	☒	☒	☒	☒
Proxy server	■	■	■	■	■	◙[1]	◙[1]	◙[1]	◙[1]	☒	◙[1]	◙[1]
Search engine manager	■	■	■	■	□[1]	□[1]	□[1]	☒	☒	☒	□[1]	□[1]
SSL/TLS version selection	☒	☒	■	■	☒	☒	☒	☒	☒	☒	☒	☒
Task manager	■	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒

[1] the control has a limitation, [2] heterogeneity in different platforms

3.5.4 Third-party software control

Desktop browsers and Firefox Mobile auto-update extensions, as soon as they become available in their application repository (e.g. Chrome Web Store). In some circumstances Alice may prefer to disable automatic updates (e.g. when she is roaming). Among the examined browsers that support extensions (c.f. Table 16), only Firefox and Safari provide Alice with this control over updates. Moreover, only Internet Explorer, Opera and Firefox Mobile do not enable Alice to manually update extensions. The rest of the browsers that support extensions provide such an interface to initiate an update, which will aid Alice to be timely protected from security vulnerabilities and bugs in extensions.

Contrary to extension updates, browsers do not update plugins automatically. Thus, browsers must provide an interface to inform Alice which plugins must be manually updated. Nonetheless, during the evaluation only Firefox and Chrome alerted users about vulnerable plugins. Firefox provides crystal clear indications when a plugin is vulnerable by highlighting it and providing an update link. Chrome alters the plugin’s version color to red and provides an update link. However, Alice may ignore this warning since it is not easily spotted among the various plugin details.

Table 16. Mechanisms for third-party software control.

Controls	GC	MF	IE	OP	AS	AB	CM	FM	IM	OM	Om	SM
Auto update extensions	□●	■●	□●	□●	■○	N/A	N/A	□●	N/A	N/A	N/A	N/A
Auto update plugins	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Disable extension	■○	■○	■○	■○	■○	N/A	N/A	■○	N/A	N/A	N/A	N/A
Disable Java	■○	■○	■○	■○	■○	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Disable JavaScript	■○	■○	■○	■○	■○	■○	□|■[1]○	▣○	☒	☒	☒	■○
Disable plugin	■○	■○	■○	■○	☒	■○|●[1,2]	☒	■●[2]	☒	☒	☒	☒
External plugin check	☒	▣	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒
Manually update extensions	■	■	☒	☒	■	N/A	N/A	☒	N/A	N/A	N/A	N/A
Manually update plugins	▣	■	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒

[1] heterogeneity in different platforms, [2] the control has a limitation

All desktop browsers, except for Safari[18], allow Alice to disable plugins as she feels fit. As summarized in Table 16, this option is not available in the majority of smartphone browsers. Smartphone browsers cannot be configured to individually block plugins (e.g. flash player video) from their settings. ABrower (version ICS and JB only, plugins in previous versions are enabled by default) and Firefox Mobile allow only limited plugin management. More specifically, the two browsers provide an 'all-or-nothing' control over the plugins and therefore Alice cannot disable individual plugins. By default, Alice will be explicitly asked to enable a plugin via a mechanism referred to as 'tap to play'. Furthermore, smartphones often invoke other applications to present content to the user (e.g. video players). Again, Alice cannot neither inspect which applications are invoked for specific content, nor disable them.

Browser's extensions (or "add-ons") can be enumerated and disabled in all desktop browsers. In smartphone browsers, only Firefox Mobile supports browser extensions and provides Alice the ability to inspect and individually disable extensions.

In desktop browsers, Java can be disabled in the plugin configuration interface - except from Safari, where Java can be disabled in the security menu tab. Java was enabled by default in all desktop browsers, even when the browser was installed in a desktop which included a vulnerable Java version. Finally, smartphone browsers do not support Java applets and returned an alternative text, namely "Your browser is completely ignoring the <APPLET> tag!" during the evaluation.

As summarized in Table 16, JavaScript is enabled by default in all browsers. While all desktop browsers allow Alice to disable JavaScript in case she feels that this will protect her from malicious content from a webpage, this option is not available in all smartphone browsers. More specifically, only Safari Mobile, ABrowser, and Chrome for Android enable Alice

[18] Plugins in Safari can only be enumerated. They can be manually removed from the plugins installation folder

with such a configuration. Chrome for iOS, Opera Mobile, Opera Mini, and IE Mobile do not offer the manageability of JavaScript, whereas Firefox Mobile offers it only from a hidden menu. Therefore, in most smartphone browsers Alice is exposed to any malicious JavaScript code.

Alice may initiate a web based plugin check only from Firefox's interface (referred as 'external plugin check'). The analysis revealed, nevertheless, that the use of this control may mislead Alice. This holds true, because when Alice enumerates her plugins she may accidentally interact with a widget in the upper right corner of the interface and not with the correct control that appears as a link. This widget checks for extension updates and not for plugin updates. The widget has a label "Check for Updates" and, as a result, Alice cannot distinguish its proper use. Moreover, if Alice had checked for extension updates before navigating to the plugins tab and no extension update was found, then the widget's label will remain "No updates found", potentially misleading Alice that there are no updates for the extensions, as well. Finally, our experimental analysis provides proof that this control is not always effective.

Table 17. Web browsing controls.

Controls	GC	MF	IE	OP	AS	AB	CM	FM	IM	OM	Om	SM
Certificate Warning	□●	□●	□●	□●	□●	■●	□●	□●	□●	□●	☒	□●
Local blacklist	■	■	■	■	☒	☒	☒	☒	☒	☒	☒	☒
Malware protection	■●	■●	■●	■●	■●	☒	☒	□●	☒	□●	☒	☒
Modify user-agent	▣	▣	▣	■	▣	■	■	■	■	■	☒	☒
Phishing protection	■●	■●	■●	■●	■●	☒	☒	□●	☒	□●	☒	■●
Report rogue Website	☒	■	■	■	☒	☒	☒	☒	☒	☒	☒	☒
Website checking	☒	☒	■	■	☒	☒	☒	☒	☒	☒	☒	☒

3.5.5 Web browsing controls

All desktop browsers provide a mechanism to protect Alice from Mallory. This mechanism includes a system's wide blacklist and/or page analysis (Google Developers; Microsoft, Opera). Chrome and Firefox use Google's *Safe Browsing* technology, Internet Explorer uses *Smartscreen* technology, and Opera and Safari do not document which engine they use. Such a mechanism is enabled by default in all desktop browsers. However, all of them enable Alice to disable malware protection without displaying any confirmation security warning. Contrarily, in smartphones, only Opera Mobile and Firefox Mobile inform Alice that this technology is supported in their support pages. Chrome Mobile's support page informs her that *Safe*

Browsing is not available, whereas the rest smartphone browsers neither provide any information to Alice about this control, nor provide any option in their menus.

Furthermore, the above mechanisms of desktop browsers also protect Alice from fraudulent websites (i.e. phishing attacks). Besides the aforementioned smartphone browsers that document protection from fraud, Safari Mobile also documents such a protection. In addition, Safari Mobile provides an interface allowing Alice to configure this control. Once more, this control can be easily disabled, without displaying any warning that this action would lower Alice's security protection.

Apart from the above two controls, which aim to block access to Mallory's websites, only Internet Explorer and Opera allow Alice to manually initiate analysis on the site she is currently visiting (referred as 'website checking'). Furthermore, only[19] Mozilla Firefox, Internet Explorer, and Opera allow Alice to manually report a rogue site.

Among the examined browsers, only Chrome, Firefox, Opera, and Internet Explorer provide local blacklists and whitelists of websites. Alice can edit these lists, as she sees fit, by adding or removing websites and setting restrictions or allowing access in the lists (e.g. blocking cookies, enable pop-ups, share location, etc.). Among the browsers that offer this mechanism, Internet Explorer provides a fine grained mapping of the rest browser mechanisms to the lists (referred as trusted/restricted zones). On the contrary, Chrome, Firefox, and Opera provide only a coarse grained mapping of the available mechanisms. This mapping is available through the page information interface (or the 'site's preferences') for a given domain or from a widget (e.g. button) near the configurable mechanism in the browser's menu[20].

Alice may wish to use a modified user-agent (UA) string in her HTTP request (i.e. a string which provides the browser's software details). For instance, Alice may prefer to navigate to the desktop version of a site with her smartphone, or to access a service that is available to a browser other than the one she is using (UA modification). Alice may change her UA in all examined browsers, except for Opera Mini and Safari Mobile, whereas in Firefox and Safari this configuration takes place only via a hidden menu. It is worth noting that UA modification may occur for privacy reasons (c.f. Eckersley, 2010).

Finally, our evaluation revealed that Opera Mini is the only browser that does not display a security warning for rogue digital certificates, i.e. either an invalid certificate (e.g., certificate with domain mismatch, expired), or an untrusted one, i.e. one that is not signed by a trusted CA (c.f. (Amrutkar, 2012)**)**. Thus, Alice is unprotected from Mallory's rogue web servers. Security warnings for rogue digital certificates may also unmask a MiM attack.

[19] The control must be initiated via the browser's interface. Other browsers may indirectly use other services such as https://support.google.com/websearch/contact/reporting_malware?rd=1

[20] Firefox also enables this mapping from a Permissions Manager that resides in a hidden menu

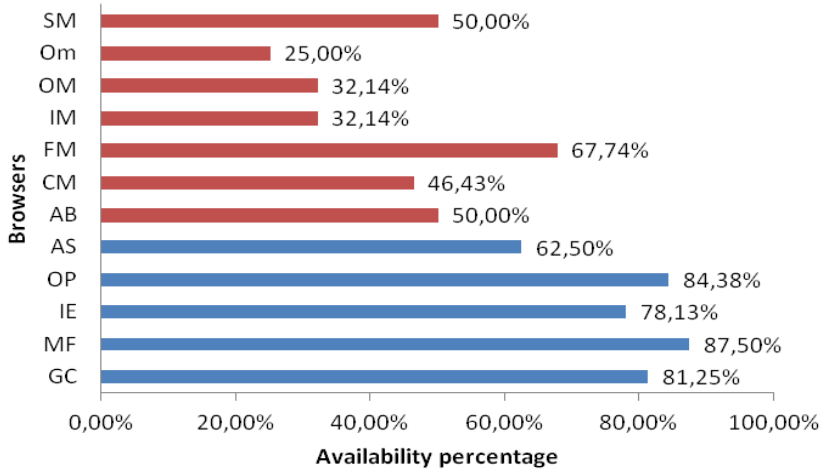


Figure 2: Availability of security controls in web browsers. The figure holds the percentage for AB (later than Gingerbread), CM (Android) and SM (iOS 6). The percentage for AB (Gingerbread), CM (iOS) and SM (iOS 5) is 46.43%, 42.86%, 46.43% respectively**.**

3.5.6 Overall availability of controls

Fig. 1 outlines the percentage of security controls provided by each browser. The descriptive statistics omit controls, where applicable, for instance "Disable Extensions" in Chrome Mobile. As the figure illustrates, smartphone browsers form three groups, regarding the availability of controls. The first group includes Firefox Mobile that offers the majority of security controls in smartphones (67.14%). The second includes browsers with control availability around 50%, i.e. ABrowser, Chrome Mobile and Safari Mobile. The browsers in the last group, i.e. Internet Explorer Mobile, Opera Mobile, and Opera Mini, provide around 30% control availability. Similarly, desktop browsers form three groups. Firefox and Opera form the group of browsers that offers the majority of security controls (87.5%, 84.38% respectively). Google Chrome and Internet Explorer provide control availability around 80%. Finally, Apple Safari browser provides only 62.5% of the security controls.

As expected, smartphone browsers implement a subset of security controls that are available in their desktop counterparts (c.f. Tables 11-15). One could argue that the unavailability of controls is due to the restrictions that are imposed by the smartphone sandbox profiles to all applications (Mylonas, 2011a). To test the validity of this argument we filtered the security controls that are implemented by at least one smartphone browser, while being not implemented by other smartphone browsers in the same smartphone OS. Such controls exist both in iOS and Android. More specifically, the controls in {Block images, Block location data, Block third-party cookies, Certificate manager, Certificate Warning, Disable JavaScript, Private browsing} are counterexamples of this argument both in Android and iOS, as well as {Block referrer, Master Password, Search engine manager} in Android (c.f. Table 18). Thus, any browser that does not support any of these controls (where applicable) is not restricted by the OS's sandbox.

Table 18: Counterexamples about sandbox restrictions

Controls	GC	MF	IE	OP	AS	AB	CM	FM	IM	OM	Om	SM
Block images	■○	■○	■○	■○	■○	■○	☒	◙○	☒	☒	■○	☒
Block location data	■●	◙○	■○	■○	■●	■○	☒|■○	◙○	■○	☒	☒	◙●
Block referrer	◙○	◙○	☒	■○	☒	☒	☒	◙○	☒	☒	☒	☒
Block third-party cookies	■○	■○	■○	■○	■●	☒	☒	■○	☒	☒	☒	■●
Certificate manager	■	■	■	■	□	□/■[2]	■/□[2]	□	□	■	☒	□
Certificate Warning	□●	□●	□●	□●	□●	■●	□●	□●	□●	□●	☒	□●
Disable JavaScript	■○	■○	■○	■○	■○	■○	□|■○	◙○	☒	☒	☒	■○
Disable plugin	■○	■○	■○	■○	☒	■○|●[1,2]	☒	■●[2]	☒	☒	☒	☒
Enable DNT	■○	■○	■●	■○	◙○	☒	■○	■○	☒	☒	☒	☒|◙○[2]
Malware protection	■●	■●	■●	■●	■●	☒	☒	□●	☒	□●	☒	☒
Master Password	☒	■○	☒	■○	☒	☒	☒	■○	☒	☒	☒	☒
Modify user-agent	◙	◙	◙	■	◙	■	■	■	■	■	☒	☒
Phishing protection	■●	■●	■●	■●	■●	☒	☒	□●	☒	□●	☒	■●
Private browsing	■	■	■	■	■	☒|◙	■[2]	■	☒	☒	☒	■
Search engine manager	■	■	■	■	□[1]	□[1]	□[1]	☒	☒	☒	□[1]	□[1]

When a browser implements the majority of security controls, it does not de facto mean that this is the most secure browser. This holds true, as browsers are preconfigured to disable security controls for the sake of functionality. Our analysis continues with the default protection from web threats.

3.5.7 Protection from web threats

This section presents a comparative evaluation of the offered protection against web threats. Specifically, our goal is to examine: (a) the protection of preconfigured security settings against web threats, and (b) the manageability of security controls that protect from certain web threats. The former provides indications of the offered protection to average users. The latter reveals the manageability of countermeasures for each threat, i.e. the flexibility to adjust the offered protection according to the users' "risk appetite" (e.g. a user may be willing to receive targeted advertising).

Table 19. Taxonomy of browser controls and web threats.

Threat (T_i)	Security Controls
Annoyance (T1)	Block images, Block pop-ups, Certificate Warning , Disable extension, Disable java, Disable JavaScript, Disable plugin, Local blacklist, Modify user agent
Browser fingerprinting (T2)	Disable JavaScript, Local blacklist, Modify user agent, Proxy server
Exploits/Malware (T3)	Auto update extensions, Auto update plugins, Block pop-ups, Browser update, Disable extension, Disable java, Disable JavaScript, Disable plugin, External plugin check, Local blacklist, Malware protection, Manually update extension, Manually update plugin, Modify user agent, Proxy server, Report rogue website, Website checking
Identity theft (T4)	Disable JavaScript, History manager, Local blacklist, Master password, Phishing protection, Private browsing, Report rogue website, Website checking
Data interception (T5)	Certificate manager, Certificate Warning, Local blacklist, SSL/TLS version selection
Phishing (T6)	Block pop-ups, Certificate manager, Certificate Warning , Disable JavaScript, Local blacklist, Phishing protection, Proxy server, Report rogue website, Website checking
Privacy breach (T7)	Block cookies, Block images, Block location data, Block referrer, Block third-party cookies, Certificate Warning, Disable extension, Disable java, Disable JavaScript, Disable plugin, Enable do-not-track, History manager, Local blacklist, Malware protection, Master password, Modify user agent, Phishing protection, Private browsing, Proxy server, Report rogue website, Search engine manager, Website checking
Resource abuse (T8)	Disable extension, Disable java, Disable JavaScript, Disable plugin, External plugin check, Local blacklist, Malware protection, Report rogue website, Task manager, Website checking
Rogue certificates (T9)	Certificate manager, Certificate Warning , Local blacklist
Spam advertisements (T10)	Block images, Block pop-ups, Local blacklist, Proxy server, Search engine manager
Technical failure (T11)	Auto update extensions, Auto update plugins, Browser update, Disable extension, Disable java, Disable JavaScript, Disable plugin, External plugin check, Malware protection, Manually update extension, Manually update plugin, Task manager

In doing so, we created a taxonomy of the security controls and web threats. The threats, which combine ICT threats (Marinos and Sfakianakis, 2012) and smartphone threats (Theoharidou et al., 2012), are listed in Table 19 along with their mapping to security controls the table uses the same notation for controls as in (MylonasSoups). The mapping was created in line with the controls' descriptions in the browser help pages, as well as the recommendations from (Veracode; CERT; CERT-US). Then, two heat maps were introduced summarizing the number of security controls that are enabled by-default in each browser, and the manageability of security controls that browsers provide, according to Tables 11-15.

Fig. 2 presents the heat map of the security controls that are enabled by-default in each browser. Our analysis revealed that desktop browsers (except for Safari) and Firefox Mobile

have the majority of pre-enabled controls (c.f. Fig. 1). Opera Mini provides no protection for the majority of the threats. Overall, the majority of pre-enabled security controls protect users from phishing, privacy and malware/exploits. Specifically: (a) desktop browsers (except from Safari) and Firefox Mobile enable by default the most controls against malware/exploits, (b) Chrome, Internet Explorer, Safari, Firefox Mobile, and Safari Mobile provide similar privacy protection, while ABrowser, Internet Explorer Mobile, and Chrome Mobile weak privacy protection, and (c) preconfigured settings in all browsers offer a comparable protection level against phishing (except for ABrowser, Chrome Mobile, IE Mobile, and Opera Mini).

Regarding the threat of technical failure (browser crashing), desktop browsers (except for Safari) and Firefox Mobile pre-enable the most relevant controls. The results indicate that all browsers provide similar protection from annoyance, interception of network data, rogue certificates, and spam/advertisements. The preconfigured browser settings provide similar protection against identity theft - except for ABrowser, Chrome Mobile, IE Mobile, and Opera Mini, which do not enable any relevant security control. Similarly, they provide comparable protection against resource abuse (except for Chrome Mobile, IE Mobile, and Opera Mini). Finally, the results suggest that none of the browsers is preconfigured to avoid browser fingerprinting.

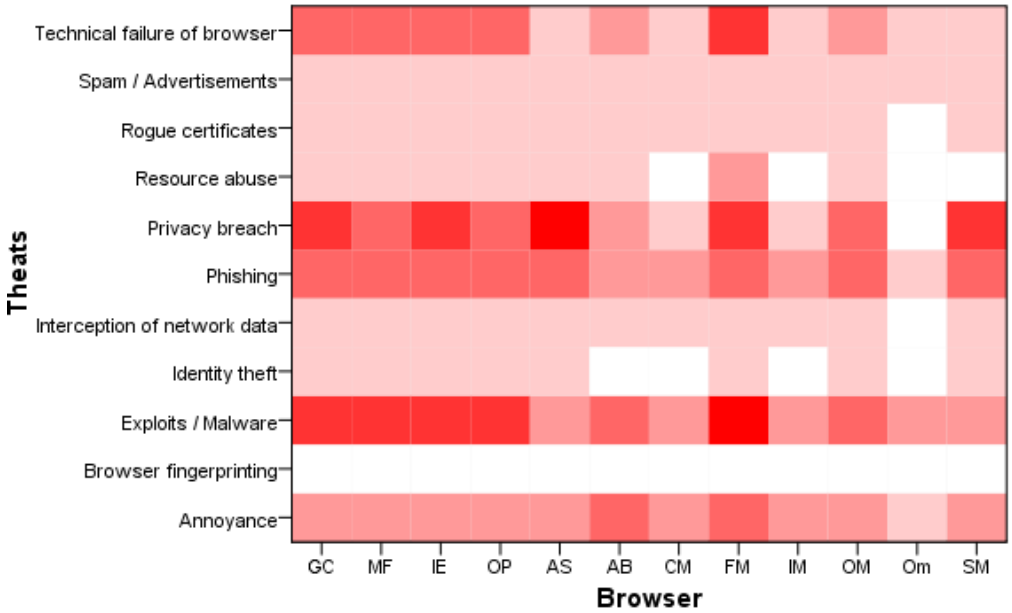


Figure 3: Preconfigured enabled security controls. For space and readability reasons the heatmaps include only CM for Android and SM for iOS 6.

Similarly, Fig. 3 summarizes the number of security controls that protect users from each web threat and are manageable. As expected, desktop browsers provide greater manageability of security controls than their smartphone counterparts. Overall, Opera and ABrowser provide the greatest manageability among desktop browsers and smartphone browsers, respectively. Chrome, Firefox and Internet Explorer offer comparable manageability and this also holds

true among Chrome Mobile and Firefox Mobile and Opera Mobile and Safari Mobile. In addition, Safari provides the least manageability of security controls in desktop browsers. Similarly in smartphones, this holds true for IE Mobile and Opera Mini. In both platforms, privacy controls are the most manageable, whereas data interception and rogue certificates are the least manageable ones. In desktops the threat of malware/exploits follow privacy controls w.r.t. control manageability, which in turn is followed by annoyance, identity theft, phishing, and resource abuse. Likewise, in smartphones the controls for annoyance and malware/exploits follow privacy controls w.r.t. control manageability.

On the other hand, browser fingerprinting, data interception, rogue certificates and technical failure are the threats in desktops having the less manageable controls. In smartphones the least configurable ones are: data interception, recourse abuse, rogue certificates, and spam/advertisements. Finally, browser crashing, browser fingerprinting, identity theft and phishing are threats that smartphone browsers do not offer manageable controls.

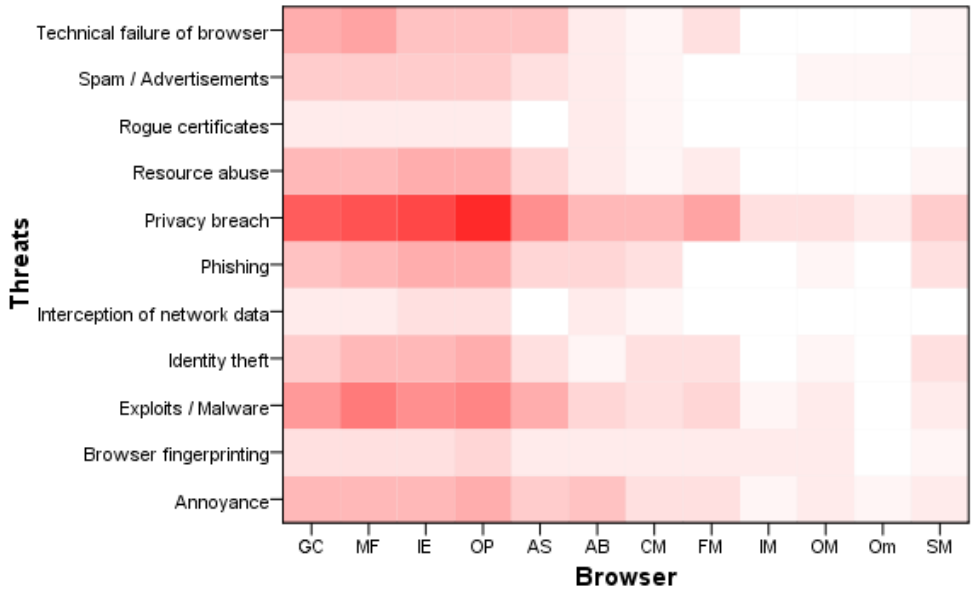


Figure 4: Manageability of security controls. For space and readability reasons the heatmaps include only CM for Android and SM for iOS 6**.**

3.6 Recommendations

This section includes suggestions for services and mechanisms that can offer Alice a better level of security and protect her from threats and vulnerabilities that were presented beforehand.

Security-oriented settings. Table 20, (columns 3 and 4) presents the preconfigured security settings in the examined desktop browsers (Table 20 uses the notation for controls that is described in Table A.1 and in Tables 11-15). These settings were collected by noting the default value and/or configurability option that appeared more often in them. Only the settings

of desktops were examined, since (a) they offer a superset of security controls comparing to smartphone browsers (c.f. Fig. 1) and (b) their longer presence in the field has made them more mature than their smartphone counterparts. As depicted in Table 20, this default configuration is functionality-oriented, i.e. provides reduced security for the sake of functionality (e.g. support of active content).

Average users are less likely to change the default values of security controls. For this reason, we propose a (default) configuration, which is security-oriented, i.e. aims to maximize the protection of the user's security and privacy (cf. Table 20 columns 2 and 4). This set extends the configuration that is proposed in (Veracode; CERT; CERT-US). We consider that all security controls should be, where applicable, implemented and enabled by default to maximize the offered protection. We also consider that a user should be able to configure them as she feels fit to adjust the level of her protection, except for (Certificate warning, Malware protection, Master password). Users should be discouraged from disabling malware and phishing protection (Malware protection, Master password) and warnings for invalid certificates (Certificate warning), or should be able to do so from an interface where only advanced users can reach, such as a hidden menu (e.g. about:config). Also, we propose that the controls Auto update extensions, Auto update plugins, and Browser update, should reside also in an advanced interface, e.g. one that asks for user confirmation before changes are applied. This will protect average users from accidentally disabling the controls.

We propose that the browser before/upon its installation should provide an interface which guides the user to configure the settings of Block cookies, Block location data, Block third-party cookies, Enable do-not-track, and Master password. This interface should be reasonably comprehensive by a normal user, e.g. "Would you like to receive targeted advertising" instead of "Enable third-party cookies" (the proposition of such an interface falls outside of the paper's scope). Finally, one should note that the configuration of the above controls by the user avoids any conflict with other stakeholders on the web (e.g. ad companies (Wired)).

Table 20 summarizes the differences between the security-oriented settings and the preconfigured settings of desktop browsers. Almost half of controls (14/32) have the same status (i.e. configurability, default enabled/disabled). We propose a 22% (7/32) to be added in browsers as configurable and/or enabled by default and 25% (8/32) of existing controls to be enabled by default. Comparing to vendor settings, 9% (i.e. Auto update extensions, Malware protection, Phishing protection) are proposed to change configurability status. Our security configuration is rather restrictive, i.e. functionality is disabled for the sake of security (e.g. cookies, location data, etc.). To ensure user experience, the browser should allow the user to enable such controls via local whitelists, similarly to the NoScript extension.

GUI issues. The browser's GUI allows users to disable security controls, which downgrades the offered protection level, as easy as modifying any other browser setting that is not security oriented (e.g. automatic resizing of images). Moreover, security controls often reside together with not security related options, such as zoom, character encoding, etc. In this case, we propose that all security oriented controls should be organized under the same interface. For instance, a tabbed menu interface may be used to organize security controls, as in Mozilla Firefox. Also, the security controls that can directly impair the security of Alice if they are disabled (e.g. disabling blacklists with malicious websites/phishing websites, disable SSL\TLS) must: (a) reside in an 'advanced' section of the configuration interface for security contorls, and (b) need a confirmation prompt (e.g. in the same manner the browser prompts the user before deleting all history entries) and/or master password before being disabled. Finally, the widgets that configure security controls must be easy for Alice to understand, avoiding confusing wording, such as in the DNT control in Safari Mobile.

Control manageability (local blacklist/whitelists). Smartphone browsers must enable Alice to map security controls to specific web domains, as in desktop browsers that use blacklists /whitelists. Otherwise, Alice has to either accept all websites or block all websites in a subset of controls (e.g. cookies, JavaScript, plugins, pop-ups, etc.).

Protection from third-party advertising. The browser must enable Alice to make an informed decision about third-party advertising. We propose that the browser should ask Alice during its installation or post installation (i.e. first time the browser executes), if she wishes to receive third-party advertising. On the background, the browser must configure the appropriate privacy controls (e.g. third-party cookies, DNT, etc.). At this point, it is rather important that the interface reasonably aids Alice to make an informed decision (e.g. with easily comprehensible text), giving her a clear understanding about the impact of third-party tracking (Madrigal). Furthermore, the interface of smartphone browsers must allow Alice to selectively delete cookies, as well as hinder some websites to install cookies in her device via a local blacklist/whitelist.

Rogue sites. Smartphones include sandboxes that place restrictions in functionality of third-party (security) apps (Mylonas et al., 2011a). As a result, while desktops' antivirus or other security software may be able to filter rogue webpages (i.e. those hosting malware and/or phishing scams), this is currently not feasible in smartphones. In this context, it appears that the browser itself must detect and/or block rogue sites. We regard that this can be achieved either with a frequent acquisition of a blacklist, or with ad-hoc queries in an online blacklist (e.g. by using a service similar to the one offered by Safe Browsing). Alternatively, the smartphone may connect to a secure proxy, i.e. one that filters rogue sites. Current smart-

phone browsers do not permit the connection to a proxy, when mobile Internet is used (e.g. 3G, HSDPA, etc).

Table 20. Comparison of security-oriented settings vs. preconfigured desktop security settings. The suggestions[1,2] do not apply to vendor settings (c.f. Tables 11-15)

Status	Suggested settings	Vendor settings	Common settings
■●	Auto update extensions[1], Auto update plugins[1], Block location data[2], Block referrer, Block third-party cookies[2], Disable extension, Disable java, Disable JavaScript, Disable plugin, Enable do-not-track[2], Master password[2]	Malware protection, Phishing protection	Block pop-ups, Browser update[1]
■○		Block location data, Block referrer, Block third-party cookies, Disable extension, Disable java, Disable JavaScript, Disable plugin, Enable do-not-track	Website checking, Block images
□●	Malware protection[1], Phishing protection[1]	Auto update extensions	Certificate Warning [1]
■	External plugin check, Manually update plugin, SSL/TLS version selection[1], Task manager, Website checking		Certificate manager[1], History manager, Local blacklist, Manually update extension, Modify user agent, Private browsing, Proxy server, Report rogue website, Search engine manager
☒		Auto update plugins, External plugin check, Manually update plugin, Master password, SSL/TLS version selection, Task manager, Website checking	

[1] configuration from an advanced interface, [2] user preference before/upon installation.

Third-party software management. Smartphone browsers must provide an interface where Alice can inspect the plugins and other applications that are being used by them. The browsers must also allow her to selectively disable this software, as she sees fit. Furthermore, both desktop and smartphone browsers must efficiently and timely inform Alice regarding third-party software vulnerabilities.

User awareness. As discussed earlier, browsers offer support pages dedicated to security and privacy. The support pages must be amended to provide enough background as well as links to material (such as National Cyber Security Alliance), focusing on the current threats on the web and the available countermeasures. This will enable Alice to understand the relevant threats and effectively adjust the browser's protection level, according to her security and privacy needs.

3.7 Discussion

This chapter provided a systematic and comprehensive analysis of the availability and manageability of security controls in popular smartphones and desktop browsers. It also provided a comparative evaluation of the: (a) protection of preconfigured security settings against web threats, and (b) manageability of security controls that protect from certain web threats. The former provides indications of the 'out of the box' offered protection to average users. The latter reveals the flexibility to adjust the offered protection according to the users' "risk appetite" (e.g. a user may be willing to receive targeted advertising). Our results can be used from browser users to adjust their protection level, as well as from browser vendors to cross-compare their security offerings.

We proved that the controls that are available in desktop browsers are a superset of the ones found in smartphones. Currently in smartphones, a user has to use multiple browsers in order to use certain security controls. For instance, in iOS she has to use Chrome Mobile for a robust control of security warnings, and Safari Mobile for both private browsing and phishing detection. This unavailability of browser controls can be partially explained by the restrictions of the smartphones' security models (e.g. private browsing in Chrome Mobile for iOS). However, our evaluation reveals occasions where a security control is available in a subset of browsers of a smartphone platform. This suggests that the restrictions from the sandbox were not the reason that the control was not implemented in the rest smartphone browsers in this platform.

Our analysis revealed that two browsers (Safari and Opera Mini) had a number of security issues, which are serious (i.e. unpatched vulnerabilities, no protection from invalid digital certificates). Furthermore, our analysis of the preconfigured security settings in browsers revealed that Firefox Mobile provides comparable protection against web threats to desktop browsers (c.f. Fig. 2). The evaluation also revealed that third-party advertising is enabled by default in most desktop browsers. In addition, in their smartphone counterparts it is not easily manageable, since they adopt all-or-nothing approach in cookie management. Therefore a user has to either accept all cookies including tracking cookies, or disable all of them, which will break the functionality of several benign websites. Also, DNT is disabled by default - or

unavailable as a control in smartphone browsers. We propose that privacy controls should be configured during browser installation or post installation (i.e. first time the browser executes), where the user should be reasonably aided to make an informed decision. Finally, private browsing is not supported in a subset of smartphone browsers.

Users are unprotected from rogue websites, which serve malware and/or perform phishing scams, while browsing with their smartphone. This holds true, as smartphone browsers fail to detect rogue websites. Users can be protected by disabling all dynamic content (i.e. the plugins, JavaScript) - a control that is not offered in all smartphone browsers. This, however, will not protect them from a Blackhole exploit framework that targets vulnerabilities in the browser's software (e.g. in plugins). It also hinders their browsing experience, since most web applications require JavaScript to function correctly. For this reason, we proposed that smartphone browsers either use a local blacklist or ad-hoc query an online blacklist with reported rogue websites, such as Google's Safe Browsing. This, however, will introduce delays, as well as additional bandwidth consumption, which in the case of mobile Internet may not be acceptable (due to cost and bandwidth quota limitations). Moreover, online queries introduce privacy issues. Alternatively, a proxy server may be used that provides detection of rogue websites. This proxy may also be used for UA modification, which protects users from system disclosure attacks. In this case, the proxy - apart from altering the device's UA - must filter out JavaScript code that may leak the UA string. Our analysis revealed that security controls can be disabled as easy as disabling controls that are not security oriented. Also, security controls often reside together with non security related options (such as zoom, character encoding, font size, etc). As discussed earlier, the interface of security controls must be reorganized in order to assist users correctly configure the security level offered by the browser.

Our evaluation focused on smartphone and Windows desktop and omits differences in the availability of controls in other platforms in the former (e.g. BlackBerry, Symbian), or latter (e.g. Mac-OS) device type. The evaluation includes the most popular devices in the two platform and as a result, we regard that our security findings are adequately representative in them. Another limitation of our work is that security controls may be added to browsers - especially to smartphone browsers - when they update. However, as discussed earlier, updates for smartphone browsers are less frequent, semi-automatic, suffer from delays (from the app market or device vendor), as well as an update may also be unavailable if the device is not supported anymore. Furthermore, the comparative evaluation of protection against web threats that was conducted is quantitative and not qualitative. In future work we plan to measure and compare the performance and efficiency of security controls.

3.8 Summary

The proliferation of smartphones has introduced new challenges in web browsing security. These devices often have limited resources and small size, which may limit the security 'arsenal' of their user. This, however, does not seem to deter smartphone users from accessing the Web via their devices. On the same time, the popularity of browser-based exploits among attackers is also on the rise, especially in the form of Blackhole exploit kit, i.e. frameworks that attack browsers using 0-day exploits (e.g., in plugins such as Java, Flash). In this context, this work contributed by comparing the availability and manageability of security controls that are offered by popular smartphones and desktop browsers. It also provided insights about their preconfigured protection against web threats.

(this page is intentionally left blank)

Chapter 4: User practices against physical and malware threats

"With great power comes great responsibility" - Vol. 1, #15 of the Marvel comic Amazing Fantasy

4.1 Introduction

Unlike PC software, smartphone applications (or 'apps') adopt centralized distribution architectures and are usually available to users from app repositories or app marketplaces. These app repositories may either be official (i.e. maintained by the smartphone platform, e.g. Apple's App Store, Microsoft's App Hub), or not (e.g. Amazon Appstore for Android). The security models of smartphone platforms provide different options with respect to the permitted sources of applications (Barrera et al., 2011; Mylonas et al., 2011a). In addition, the strictness of app vetting controls in an app repository[21] ranges from relaxed app submission in community-based app repositories, such as Google play, to strictly controlled repositories that follow the 'walled garden' model, such as Apple's App Store (Barrera et al., 2011; Mylonas et al., 2011a, 2011b). Regardless of how strict and centralized the security model of a platform may be, it always leaves some choice to the user. Again, this delegation can be simply authorizing access to some protected resources, or may give user the choice to infer if an application may impair her security and privacy.

Meanwhile, the rate of downloads for smartphone applications from app repositories is on the rise (Baghdassarian and Milanesi, 2010). This popularity of smartphone applications has drawn the attention of attackers, who try to use the app repository as a security and privacy attack vector. In this context, an increasing number of malicious applications have already been discovered in app repositories (Felt et al., 2011b; Zhou et al., 2012b).

This is one of the reasons that smartphones have also drawn the security literature's attention. The security literature that focuses on smartphone applications has elaborated on malicious application identification (Egele et al., 2011; Enck et al., 2009,2010, 2011; Nauman et al., 2010; Zhou et al., 2012a, 2012b; Zhou and Jiang, 2012). Automated scanners have been proposed to aid advanced users deduce whether an application requests permissions that can impair the security and\or privacy of users (Enck et al., 2010; Felt et al., 2011a; Hornyack et al., 2011). Nonetheless, it is unclear whether the burden of making security decisions is reasonable for average users. Studies have shown that average users are not able to make such deci-

[21] Unless stated otherwise, in the rest of the paper the term "app repository" refers to an official app repository.

sions, nor are able to use security controls adequately (Furnell, 2005, 2007; Furnell et al., 2006; Whitten and Tygar, 1999).

In this chapter, we examine the security awareness of smartphone users who install applications from official app repositories. Particularly, we conducted a survey with the aim to answer the following main research questions:

- Q_1: Do smartphone users enable security controls on their devices?
- Q_2: Do users consider security while choosing and downloading applications?
- Q_3: Do smartphone users trust applications from the official app repository?

The survey's scope includes only users who download applications from the official app repository of the current popular smartphone platforms, i.e. Android, BlackBerry, iOS, Symbian and Windows Phone. Our survey results indicate a clear lack of smartphone users' security awareness with regards to the adoption of the available security controls (pre-installed on the controls and/or third-party controls), as a well as their practices while installing third-party apps from the app repository, which erroneously regard to be risk free (c.f. research Q_3 above).

We continue our analysis by further examining whether this alerting mindset towards smartphone security is affected by the participants' security background (Theoharidou and Gritzalis, 2007). In specific: we split the sample population in two groups, comprising from security savvy and non-security savvy participants. Our goal is to answer the following research question, i.e.

- Q_4: Does the participant's security background affect their security awareness?

The participants' security background, i.e. at minimum comprehension of the notion of threat, risk, and safeguards, was derived from sources either academic (e.g., university information security courses) or industrial (e.g., information security certifications). Our results suggest that the participants' security background has a slight impact on their awareness about security and privacy in the smartphone ecosystem.

The rest of the chapter is organized as follows. The next section presents related work. Section 3 provides the reader with the methodology of the survey. In Sections 4 and 5 the findings from the summary of the sample responses and the essential statistical analysis are presented, respectively. In Section 6 the prediction model is described and its effectiveness is evaluated. Finally, Section 7 includes the survey's limitations, whereas Section 8 includes a discussion of the results and conclusions.

4.2 Related work

Even thought smartphones are well studied in security literature, the relevant research work on the security awareness of smartphone users is currently rather limited and mainly focuses on Android. Chia et al. (2012) studied risk signaling concerning the privacy intrusiveness of Android applications in two application repositories, i.e. Android market[22] and AppBrain.com. Their results suggest that the number of dangerous permissions that an application requests is positively correlated with its popularity. Even though users understand the notion of application popularity, the fact that an application is popular does not imply that it respects the users' privacy. Moreover, their results indicate that the current risk signals employed by an app repository (e.g. developer's website, application reputation) become ineffective over time, as users tend to click through them. Our findings also indicate that users tend to ignore the reputation and the reviews of an application, as well as the security and agreement messages received during application installation from app repositories.

Similarly to our user survey, smartphone users were found to ignore security messages during application installation in (Felt et al., 2012; Kelley et al., 2012). Moreover, they were unable to comprehend the permissions and the risks that are related with them (Felt et al., 2012; Kelley et al., 2012). As a result, in both studies the Android security messages did not assist most users to make appropriate security decisions. Our results suggest that the majority of respondents ignore every aspect of security and privacy during application selection, as well as the app's reputation, reviews and security and agreement messages. Nonetheless, when explicitly asked, a minority of users in the survey conducted by Felt et al. (2012) reported that they have cancelled the installation of an application due to its permission requests. In our survey a minority of users was found to delve into security and agreement messages, who tend to be security and technology savvy.

Also, in another user study that was conducted parallel to our user study, participants were found not to consider security and privacy issues during app selection, as they tend to ignore privacy policies and EULAs. Finally, in the user study conducted in (Kelley et al., 2012) users erroneously believed that applications undergo security analysis during their submission in the Android Market. In our study we also found such misconceptions about application testing in application markets. Moreover, most users were unaware of the existence of the application testing mechanism.

[22] Android Market is the smartphone app repository Google maintained before merging and rebranding it with other digital content services in Google Play (http://googleblog.blogspot.com/2012/03/introducing-google-play-all-your.html, March 2012).

4.3 Methodology

To assess the security awareness of smartphone users, a survey was conducted from September-December 2011, in Athens, Greece. This section presents the survey methodology, as well as some details about the tests mounted so as to ensure the validity and statistical significance of the results.

4.3.1 Data collection

The survey responses were collected from random people on the street and from public transportation means (train stations, underground, airports), via structured interviews (Flick, 1998). A questionnaire (see Appendix) was used for the structured interviews. The duration of the interview completion was 5-8 minutes, on average. The discussion with the user aimed to ensure the validity of her responses, the comprehension of the questions, and the comprehension of technical terms.

Questions 5 and 7 were used to filter out the non-smartphone users and the smartphone users who did not install third-party applications in their devices. We excluded these two user groups because the survey focused on smartphone users who do download applications from app repositories. Questions 2 and 8 were the only free text questions.

4.3.2 Sample demographics and background

Reports on Internet penetration in the EU member states (Communications Committee, 2011) show that Greece had one of the highest increases in fixed Internet connection penetration in 2011. On the other hand, mobile Internet penetration was only 27.1%, considerably lower than the EU average (34.6%). However, in January 2011[23] the subscriptions for access to mobile Internet via cell phones increased by 140%, comparing to the first half of 2010. This indicates that smartphone adoption in Greece is on a significant rise.

The sample's population includes 458 smartphone users, aged (min_{age}=15, max_{age}=52). Among them, 81% were aged [15-30] (Appendix contains the sample's age distribution) and 70.1% were male. The 56.3% reported that they were non-security savvy users (Q4), i.e. not having a generic security background (Theoharidou and Gritzalis, 2007) from a source either academic (e.g. university security courses) or industrial (e.g. IT security certifications). Regarding their IT skills, the sample reported that: (a). 10.3% had moderate IT skills, (b). 41.3% had good IT skills, and (c). 48.5% had excellent IT skills.

Amongst the smartphone users, the 61.6% of the sample were aware of the term *jailbreaking* and 81.4% were aware of smartphone malware existence. Also, 95.2% were concerned

[23] Observatory of the Greek Information Society (http://www.observatory.gr/).

about their privacy, 75.8% stored personal data in their devices, and 35.8% stored business data. Finally, 30.1% of them reported that they had misplaced their device, at least once in the past.

The popularity of smartphone platforms in the sample is depicted in Table 21. The second column presents the proportion of users in each platform, along with the popularity rank amongst the platforms. The platform popularity that was discovered in the Greek sample is comparable with a Gartner report[24] concerning global smartphone popularity at the time of the survey's analysis. Android, iOS and Symbian were the dominant smartphone platforms in both surveys, while in our sample Windows Phone had more popularity than Blackberry.

Table 21. Platform popularity (during survey analysis, Q4 of 2011)

Operating System	Sample popularity	Gartner popularity
Android	38.4% (1)	50.9% (1)
BlackBerry	9.2% (5)	8.8% (4)
iOS	23.8% (2)	23.8% (2)
Symbian	16.6% (3)	11.7% (3)
Windows	12.0% (4)	1.9% (5)
Other	0% (6)	2.9% (6)

4.3.3 Sample Considerations

To ensure the quality of the statistical analysis, the survey sample must satisfy the simple and randomness, reliability, and representativeness properties. This section summarizes the statistical tests conducted to ensure that these properties are fulfilled by the sample.

The collection methodology of the sample was *simple* and *random*, as the selection of a smartphone user u_i was not dependent on the selection of user u_{i-1}. We approached random users and asked whether they owned smartphones and whether they wished to participate in a smartphone survey about smartphone app usage. Then, we asked if they knew what an app repository (or app market) is, and if they had installed from it any third-party application. A printed questionnaire was given to the smartphone user only if she had installed at least one application in her device.

Sample *reliability* deals with the accuracy of measurement in a statistical analysis (Cronbach, 1951). Reliability analysis ensures that if the same measurement instrument (i.e. the questionnaire) is given to a different sample, then the results of the statistical analysis must be similar. In this context, the survey results may change only if the sample expresses a different

[24] Gartner 4Q11 Market Share (%) (http://www.gartner.com/it/page.jsp?id=1924314).

opinion and not because of any confusion or misinterpretation. Hence, to ensure the reliability of the sample, we successfully tested it against the Cronbach alpha algorithm (α=0.506).

We tested the *representativeness* of our sample by performing x^2 goodness-of-fit tests. We tested the null hypothesis, i.e. whether the observed frequencies in our sample differ from the expected ones by assuming equal probability for every question. We rejected the null hypothesis for every response that we collected from our sample, hence the sample is representative for populations with similar ages [15-52].

Finally, we also a priori estimated the required sample size (Cohen, 1998), so as to conduct our statistical analysis test with the x^2 distribution, with power level 95% and significance level 5%. As such, we estimated that the required sample size for the statistical analysis is 145 cases. Details regarding these tests are given in Appendix.

4.3.4 Data analysis

4.3.4.1 Analysis of the whole sample

The analysis of the *whole population* of the survey participants was twofold. Firstly, it includes the computation of descriptive statistics. The percentages, which are given in §8.4 refer to proportions of the sample population or to smartphone platforms users. In the latter, they refer to the proportions of a particular smartphone platform[25]. Contrarily to inferential statistics, descriptive statistics provide response summaries and, as a result, are bound to the Greek sample. Nonetheless, the survey summary indicates the smartphone users' security mindset.

To further investigate the security awareness of smartphone users, we checked the correlation between the responses in the questionnaire. For doing so, we drafted the contingency tables of every pair in the categorical variables in the questionnaire (Appendix lists the variables and their notations). Then, for each pair of categorical variables a and b, we examined whether an association between them exists, by computing the appropriate x^2 distribution test of independence. More specifically, for each variable pair a, b we tested the null hypothesis (significance level a=0.05), i.e.:

H_0*: variables a* and *b* are *independent*
H_1*: variables a* and *b* are *not independent*

In the cases where: (a) the expected cell frequency for each cell in the contingency table was ≥5, and (b) a statistically significant relationship ($p<0.05$) between the variables was dis-

[25] In this case, they are given to summarize the responses of smartphone users and not to compare the frequency of the answers among platforms, as the number of different platform users differs.

covered, we further investigated the association by computing the φ coefficient. The φ coefficient revealed the correlations' direction, i.e. positive the two examined questions (a, b) in the pair tend to have the same values, or negative two examined questions (a, b) in the pair tend to have the same values. We filtered out findings that did not offer anything to the security discussion, such as correlation between device misplace and efficiency as criterion for app selection.

4.3.4.2 Analysis of two groups

For the *analysis of the two groups* of security savvy and non-security savvy users we used the same analysis methods that was described previously, but in the two groups independently. More specifically, we splitted the sample into two groups (*SECSAV, NSECSAV*), according to the participants' responses in Q_4 of the questionnaire, in order to examine any differences in their responses. *SECSAV* included users with a *generic* security background (Theoharidou and Grtizalis, 2007) and *NSECSAV* included non-security savvy users, respectively. As noted beforehand, the generic security background included at minimum the comprehension of the notion of threat, risk, and safeguards. It was derived from sources either academic (e.g., university information security courses) or industrial (e.g., information security certifications).

Initially, we examined if the groups (i.e. *SECSAV, NSECSAV*) responded similarly in the same questions of the questionnaire. This task would give us whether there are (statistically significant) differences in the two groups' responses. For doing so, we computed the appropriate x^2 distribution test of independence (significance level a=0.05), between the responses regarding the security background question (*sec_background_question*) and the responses of the rest instrument's questions. In specific, for every pair {*sec_background_question*, *question*$_i$} in the instrument, we tested the null hypotheses:

H$_0$: *sec_background_question* and *question*$_i$ are *independent*.
H$_1$: *sec_background_question* and *question*$_i$ are *not independent*.

Then, we further analyzed the groups by identifying and comparing correlations between their responses. To this end, we computed for each group of smartphone users, the x^2 distribution test of independence (significance level a=0.05) for the responses of every pair of questions {a,b} in the instrument. More specifically, we tested the null hypothesis:

H$_0$: *question*$_a$ and *question*$_b$ are *independent*
H$_1$: *question*$_a$ and *question*$_b$ are *not independent*.

All correlations of question pairs that were statistically significant and their expected cell frequency for each cell in the contingency table was ≥5, were further analyzed by computing the φ coefficient, so as to find the correlations' direction (i.e. positive, negative). We filtered out findings that did not offer anything to the security discussion (e.g. correlation between device misplace and efficiency as criterion for app selection). Finally, we compared the different associations of the two groups.

4.4 Results of statistical analysis

This section presents the results from our statistical analysis of the user survey. Initially the results from the analysis in the whole population are presented in the first two subsections. The results from the analysis of the two subgroups of the participants are given in the last two subsections, respectively.

4.4.1 Results from Descriptive Statistics

This section presents the findings that were obtained by descriptive statistics. The percentages refer to proportions of the sample population or to smartphone platforms users. In the latter, they refer to the proportions of a particular smartphone platform[26]. Contrarily to inferential statistics, descriptive statistics provide response summaries and, as a result, are bound to the Greek sample. Nonetheless, the survey summaries indicate the smartphone users' lack of security awareness.

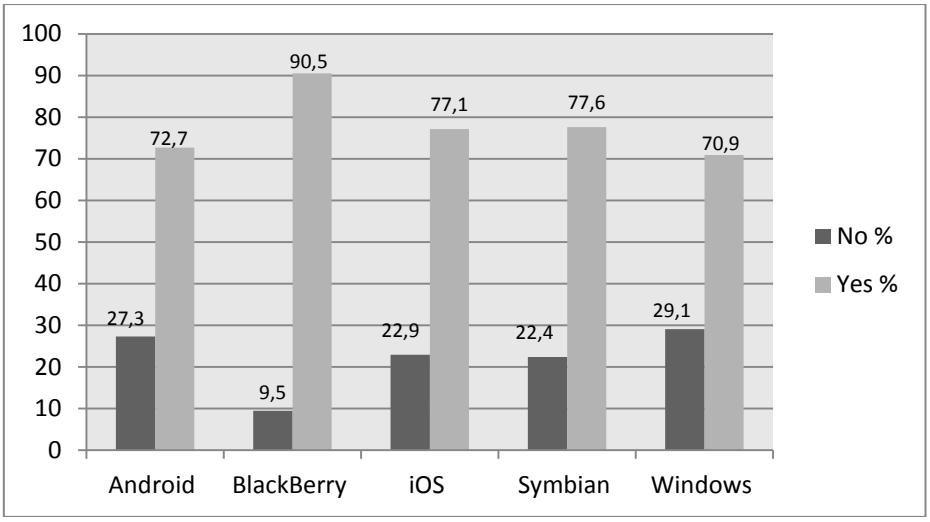


Figure 5: Trust in app repository among smartphone platforms

[26] In this case, they are given to summarize the responses of smartphone users and not to compare the frequency of the answers among platforms, as the number of different platform users differs.

4.4.1.1 Trust in the app repository

A commonly adopted definition for *trust* is hardly available in the literature. In this work we adopt the definition of McKnight and Chervany (1996), where trust is "*the extent to which one party is willing to depend on somebody or something in a given situation with a feeling of relative security, even though negative consequences are possible"*. In the context of smartphone application repositories, the negative consequences occur when the app repository is used as a privacy or security attack vector, i.e. malicious applications are submitted in the repository that impair the security and/or privacy of the users who download them. Some researchers consider that the concept of trust in McKnight's and Chervany's work refers to business trust, only, i.e. the social context of trust that a computer system performed the functions it was supposed to perform, and not on information security, which was not a major concern at that time. Some others believe that this definition is also applicable to information security.

Our results revealed that 76% of the survey participants believed that applications downloaded from the application repository are secure[27]. This is not in line with literature findings on smartphone security and malicious software; namely:

1. Security controls that application repositories often use, such as security analysis of apps (hereinafter 'application testing'), application 'kill switch' (i.e. remote application removal mechanism), etc., are not enabled in all official application repositories (i.e. Google Play, Blackberry App World, OVI store, Apple's App Store and Windows Marketplace) (Mylonas et al., 2011b).
2. Application testing has been proven inadequate to filter malicious applications by the smartphone security literature (Anderson et al., 2010; Egele et al., 2011; Enck et al., 2009, 2010, 2011; Hornyack et al., 2011; McDaniel and Enck, 2010; Nadji et al., 2011; Quirolgico et al., 2011), as well as by Cohen's analysis on malicious software detection (1989).
3. Application testing fails to satisfy the heterogeneity in each user's privacy requirements (McDaniel and Enck, 2010), suffers from time restrictions and the increasing rate of application submission in the repository (Anderson et al., 2010; McDaniel and Enck, 2010), and often requires manual analysis, which exposes the security testers to social engineering attacks (Felt et al., 2011b; Niemela, 2012).

Fig. 1 shows that approximately 3 out of 4 respondents in each smartphone platform do trust the app repository. BlackBerry users topped, in terms of trust in the app repository (90.5%), even though this app repository appears to have the least security controls in place (Mylonas et al., 2011b).

[27] For brevity reasons, in the rest of the paper we refer to this as 'trust in the app repository'.

Finally, we note that this feeling of trust can be one of the available enablers for users that do not download apps in their devices (e.g. the users in (Androulidakis and Kandus, 2012)) to start doing so.

4.4.1.2 Misconception about application testing

Another interesting finding is that users are unaware of the presence or absence of the mechanism that official app repositories use to test apps against malicious behavior. More specifically, 54.6% of the sample's population was not aware whether any application testing takes place in the app repository. This finding indicates that users regard the app repository as trusted, regardless of the fact that they are not sure whether app testing takes place.

Application testing has gained considerable attention in the smartphone security literature. Application testing may be performed by using the resources of the application repository, e.g. as in (Quirolgico et al., 2011; Gilbert et al., 2011; Zhou et al., 2012a, 2012b), or be crowdsourced as in (Amini et al., 2012). In addition, application testing can be amended with a submission policy that aims to deter submission of malicious applications, by imposing momentary penalties to mischievous users (Mylonas et al., 2011b).

In practice, application testing does not take place during application submission in the Google Play[28] and in the BlackBerry App World (Mylonas et al., 2011b). This means that these two app repositories may be used more easily as an attack vector, in order to deliver malware to smartphone users. In the analysis presented in Fig. 2, the misconception of smartphone users with regards to application testing in the repository is augmented, since (a) some users (7-10%) faulty reported that application testing does not take place in the iOS, Symbian and Windows Phone platform, and (b) relatively more users (34-38%) faulty reported that application testing does take place in the Android and BlackBerry platform. From these two misconceptions, the latter is the most security critical, as users appear to consider that applications have been security tested.

As the distribution model of smartphone applications is centralized, this may pose an opportunity for malware mitigation. If the application repository employs an app testing mechanism, this mechanism can prevent malware spreading via its distribution mechanism. Nonetheless, the existing app testing mechanisms suffer from false negatives and false positives. In any case, even though their effectiveness is debatable, these mechanisms do add a level of defense against the malware threat; especially from the threat of non-sophisticated attackers (Mylonas et al., 2011a).

[28] Our survey took place before Bouncer, i.e., Google Play's app testing mechanism, was enabled in Google Play.

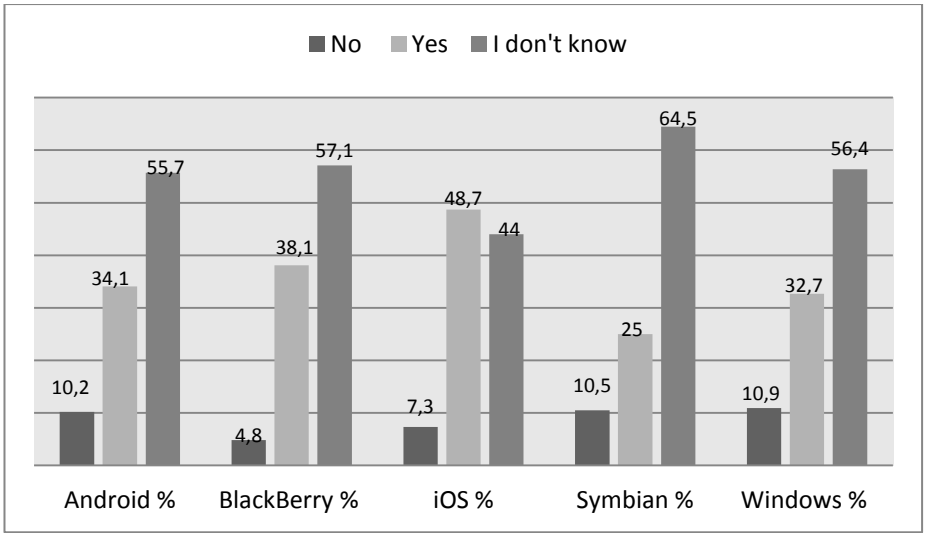


Figure 6: Application testing among smartphone platforms

4.4.1.3 Security and Agreement Messages

Literature has revealed that users tend to ignore repetitive warnings (Egelman et al., 2008; Motiee et al., 2010; Sunshine et al., 2009), especially when a warning appears while the user attempts to fulfill a task (Motiee et al., 2010). Moreover, users tend to pay less attention to consequent warnings, especially when these warnings resemble an End-User License Agreement (EULA) (Böhme and Köpsell, 2010).

The security models of smartphones prompt users with security messages when an application requests access to a protected resource during its installation, i.e. installation permission, or request for runtime access to a resource, etc. They also prompt users with agreement messages (e.g. licensing messages, term and conditions, etc.), concerning the privacy of their data. The security models are heterogeneous, ranging from open and community-based models to strictly controlled ones that follow the walled garden model (Barrera et al., 2011; Mylonas et al. 2011a, 2011b). The former delegate a user to make all authorization decisions for app access requests (e.g. as in Android). According to the latter, every application undergoes security analysis during submission and the user makes only a few authorization decisions (e.g. as in Apple's iOS). In both cases the security models assume that the user will scrutinize these messages, so as to make informed decisions. Otherwise, a user may be exposed to security and privacy attacks generated even by non-sophisticated attackers (Mylonas et al., 2011b).

Fig. 3 depicts the sample's responses with regard to attention paid by the respondents to security and agreement messages. Users were found to focus more on security messages rather than agreement messages. More specifically, the percentage of users who always scrutinize security messages is 38.6%, while only 10% thoroughly examine the agreement messages. Moreover, Fig. 3 suggests a clear asymmetry between the discovered attitude towards security messages and the security model expectations about user attitude towards them. This asym-

metry becomes a serious vulnerability when the examination of security messages is a cornerstone in the security model. For instance, in a community-based security model, e.g. as in Android, which authorizes users to grant application access to protected resources, ignorance of security messages leaves users exposed to malware.

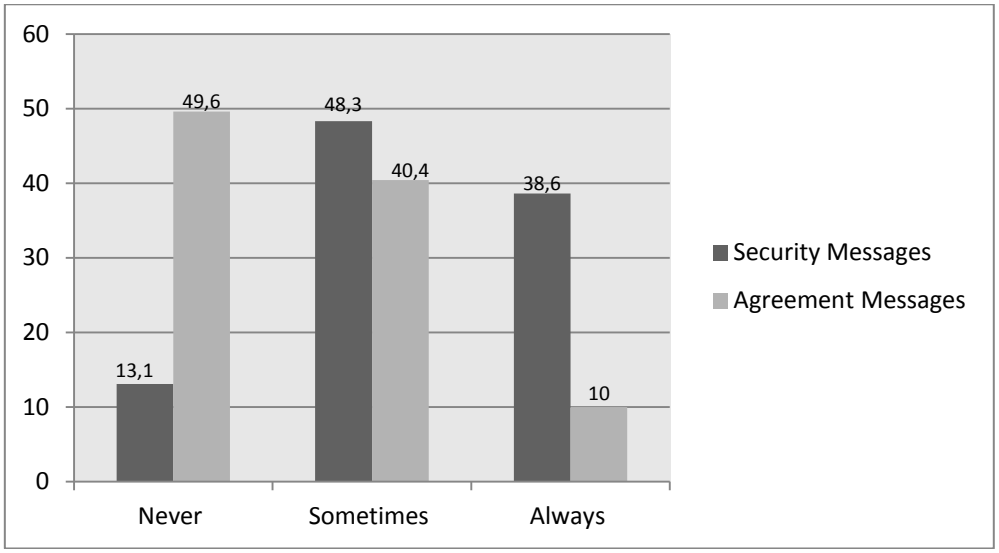


Figure 7: Security and agreement messages

In this context, Fig. 4 shows the user attitude towards security messages in each smartphone platform, where the majority of users only occasionally inspect security messages and the minority of them never inspect security messages. Also, Fig. 5 shows that the majority of users in each platform ignore agreement messages. This is orthogonal with the users view towards privacy, where the majority of users (95.2%) reported concern of their privacy.

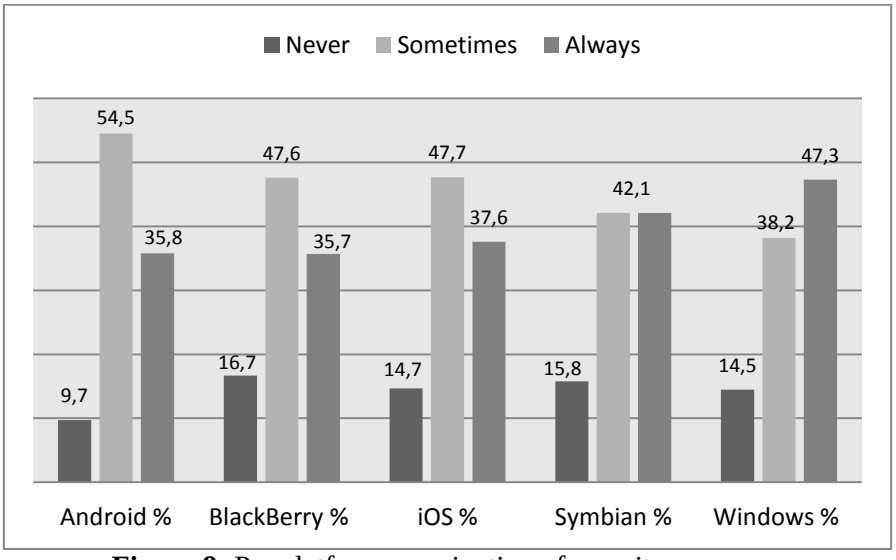


Figure 8: Per platform examination of security messages

4.4.1.4 Pirated applications

Pirated smartphone applications are a common malware spreading vector (Zhou et al., 2012a; Zhou and Jiang, 2012). Attackers often lure naive (smartphone) users into downloading malware that are masquerading as popular\attractive applications. These applications may enter unobstructed in application repositories that do not operate application testing. Even in application repositories adopting the *walled garden* model (Barrera and Van Oorschot, 2011), which adds a layer of protection against this threat, users often modify (crack) the security model of the device (e.g. in iPhone jailbreaking), so as to install free versions of application (cracked) that do not reside in the official app repository. This security model modification circumvents the security mechanisms and may expose the user to malware.

The 60.7% of the Greek sample reported preference to pirated applications from original ones. This percentage is in accordance with a survey about PC software piracy that was performed in (BSA, 2012), revealing a considerable piracy rate in Greece (61% while only 33% on average in the EU). As depicted in Fig. 6, preference to pirated applications in Android and Symbian were on average the greatest. Android and Symbian are the two of the three platforms where the security model allows apps to be installed from sources other than the official repository (Mylonas et al., 2011a). BlackBerry is the third platform allowing this flexibility, but as in iOS and Windows Phone that both follow the *walled garden* model, about half of its users reported preference in pirated applications. As a result, there is a significant likelihood that users get infected by a trojanized pirated application.

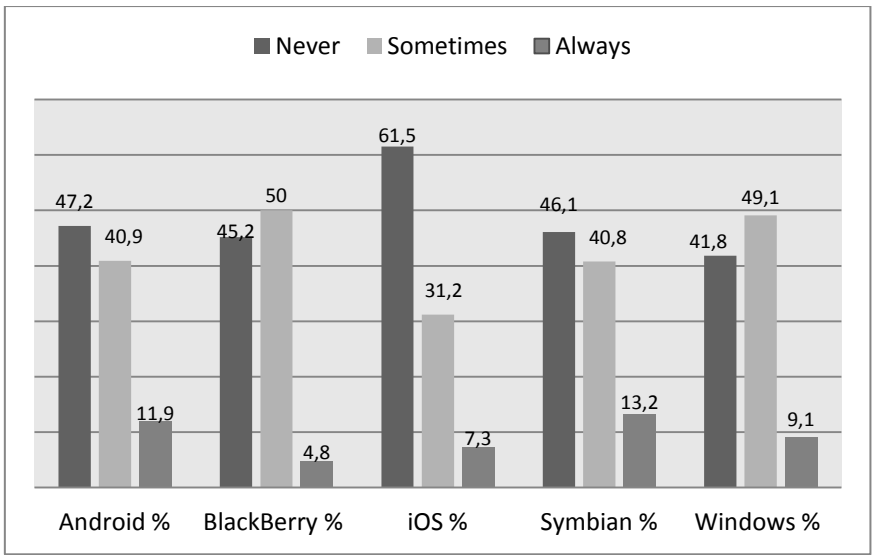


Figure 9: Per platform examination of agreement messages

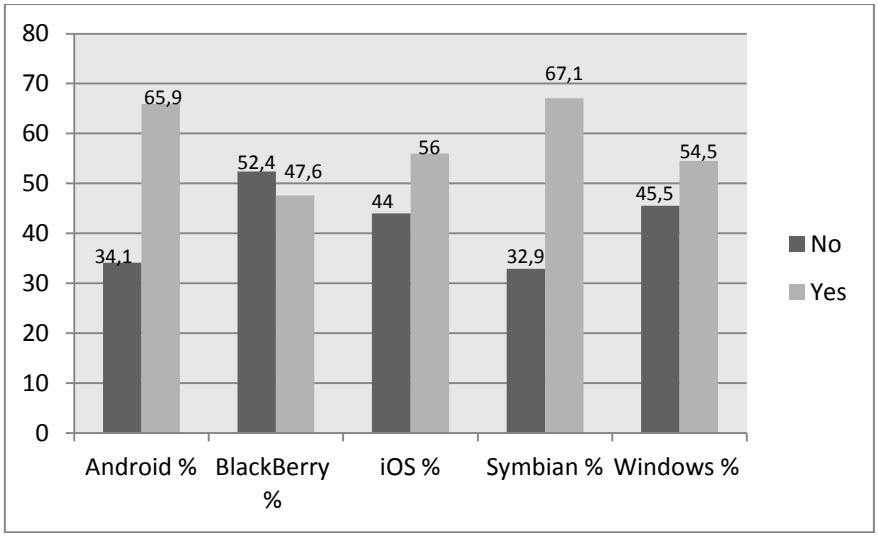


Figure 10: Preference to pirated application in each platform

4.4.1.5 Adoption of security controls

Sample analysis findings suggest a poor adoption of security controls. This holds for physical controls (i.e. encryption, device password lock, remote data wipe, remote device locator), which pre-exist in most smartphones (hereinafter for brevity reasons we refer to this group of physical controls as 'pre-installed' controls), and for third-party security software that had to be installed by the user (question 17), e.g. antivirus, anti-theft software, etc.

As far as third-party security software is concerned, 24.5% of the survey participants used smartphone security software (referred hereto as smartphone *secsoft*). On the contrary, 85.8% of the sample uses secsoft in their personal computer (PC). This is a clear indication of the asymmetry, with respect to security awareness of the available threats and security controls in the two platforms. This indication is augmented as: (a) 60% of the sample population reported that they have not searched the app repository for smartphone secsoft that is provided without cost, and (b) 27% were not aware that smartphone secsoft exists. One should note, however, that the heterogeneity of the smartphone security models often imposes restrictions on the availability of third-party security software. For instance, the inability to place hooks in the OS due to sandbox restrictions limits the antivirus availability or the security features that it can offer in smartphones. As a result, users often 'root' their devices, so as to install such security software. This often breaks the security offered by the platform and may also void their warranty.

Moreover, 34.3% of the respondents believed that smartphone secsoft is not essential. This attitude may stem from technological parameters, e.g. battery consumption, overall device performance (i.e. delays), and/or cognitive limitations (i.e. unawareness of existing threats and available controls), and/or psychological parameters, i.e. the trust in the application mar-

ket itself. Nonetheless, smartphone secsoft poses an additional line of defence against malware threat, in particular against the threat of non-sophisticated attackers (Mylonas et al., 2011a).

Fig. 7 presents the per-platform responses to the above mentioned questions. On average, Android users were more often found to enable smartphone secsoft, especially antivirus software. This may be due to the DroidDream incident that took place in February 2011, where 50 malicious applications were detected in the Google Play (Lookout, 2012). In addition, only 18.4% Symbian participants used secsoft, even though the majority of them consider them essential. This poor adoption is alerting given that several malware variants target Symbian (Coursen, 2007). On the other hand, the notably poor adoption of secsoft that was found in iOS may be due to (a) the restrictions imposed by Apple's sandbox (i.e. users are not allowed to install third-party antivirus software in their device), or (b) the fact that iOS applications undergo security analysis during their submission. However, iOS users can opt for third-party anti-theft software (e.g. remote data wipe, remote device locator, etc.), which offer more functionality than the pre-installed software offered by Apple.

Users reported low percentages in searching for free smartphone secsoft, except in Android, where the percentage is 50.6%. Finally, on average, only 66.5% of the users in every platform considered smartphone secsoft essential.

As far as the pre-installed security controls are concerned, the survey demonstrates a poor adoption. Device password lock (64.4%) was, on average, the most adopted control, while the rest had the following adoption: encryption (22.7%), remote data wipe (15.1%), and remote device locator (23.1%). Another finding was that 27.9% of the survey respondents did not use any of them. This finding, as well as the poor adoption of these controls, suggests a serious risk of unauthorized access, due to the device's portability and small size. Also, the percentage of users who do not use these controls is significantly larger than the percentage of the participants who do not use secsoft in their devices (PC and smartphone, 7.2%).

Fig. 8 illustrates the adoption of the above controls in each smartphone platform. We note that (a) remote data wipe and remote device locator had the greatest adoption in iOS users, and (b) a misconception about encryption exist in iOS users. For the latter, when a user password protects her iPhone, she also enables encryption with a key derived from the provided PIN. In the survey, 72.6% of the iOS users who enabled password lock believed that they had encryption disabled.

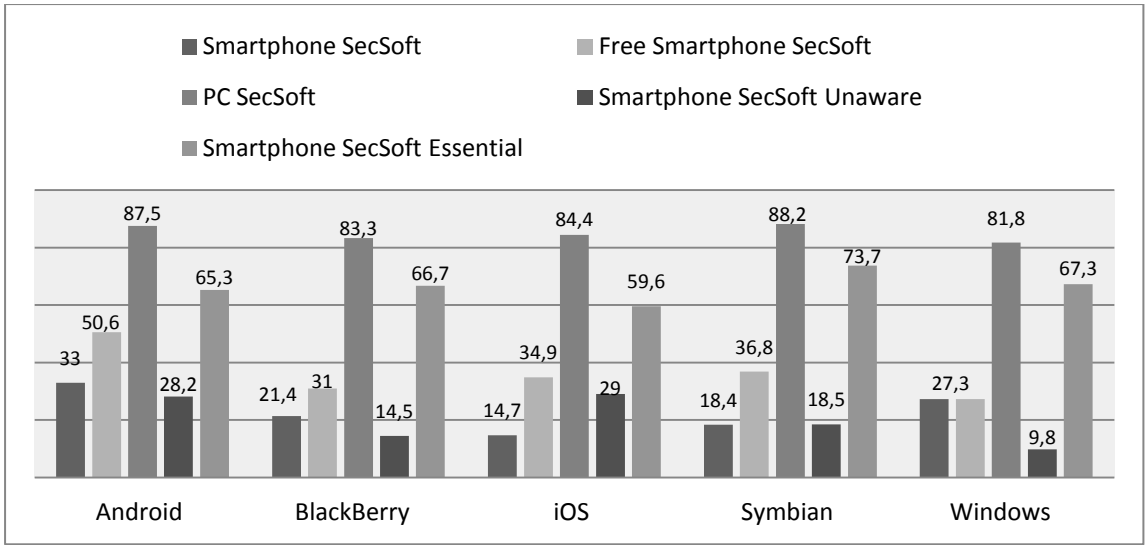


Figure 11: Third-party SecSoft adoption in smartphone platforms

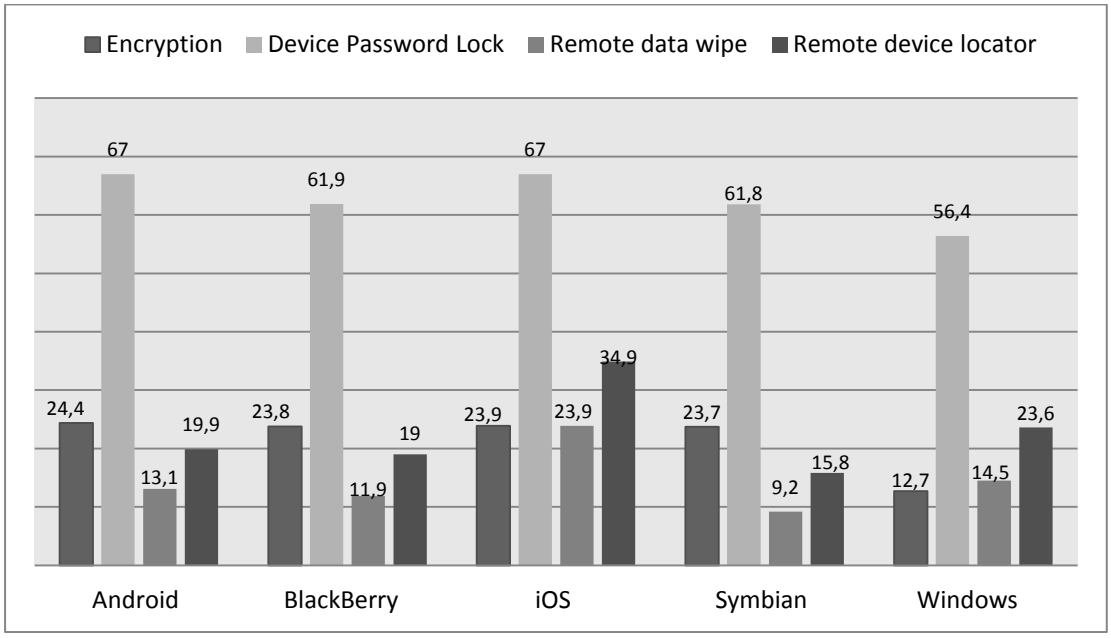


Figure 12: Adoption of pre-installed security controls in each platform

4.4.1.6 Application selection criteria

The examination of the survey results, revealed eight application selection criteria, namely: (a) cost, (b) developer, (c) review, (d) reputation, (e) security/privacy, (f) usefulness, (g) usability, and (h) efficiency. Smartphone users identified these criteria directly, e.g. "*if the application is useful for me*", "*how expensive it is*" or indirectly, e.g. "*how many stars it has*", "*I read what other users think about it in the application page or forums*".

Survey findings indicate that smartphone users disregard security issues when it comes to application selection from the official app repository. A smartphone user who is concerned about her security and privacy is expected to spend time on the application's reviews, its reputation, and its functionality, in particular in permission-based security models (e.g. An-

droid's, Windows phone) where access to sensitive functionality is user-granted. The survey revealed that only 10.5% of the respondents delve into reviews and only 8.7% consider reputation. More than the half of the sample population (58.5%) identified usefulness as an application selection criterion. Usability, efficiency, and cost had a lesser proportion of respondents. Finally, the application source (developer), as well as security or privacy issues (e.g. drop the installation of an app because a combination of requested permissions may impair privacy), were identified by a low proportion of respondents (3.7% and 3.5%, respectively) (Table 22).

Table 22. App selection criteria

App criterion	% of respondents
Usefulness	58.5
Usability	15.7
Efficiency	15.1
Cost	13.1
Reviews	10.5
Reputation	8.7
Developer	3.7
Security/Privacy	3.5

4.4.2 Results from inferential statistics

This section describes the findings from the statistically significant correlation between the responses in the questionnaire. We group the findings into: (a) user beliefs and perceptions, (b) user practices, and (c) app selection criteria.

4.4.2.1 Associations of user beliefs and perceptions

This subsection includes findings from inferential statistics, with respect to: (a) user trust in the app repository, (b) users regard of smartphone secsoft being essential, (c) privacy concern and preference to pirated applications, (d) awareness of the smartphone secsoft existence, and (e) awareness of application testing in the app repository.

User trust in the app repository. Table 23 depicts the associations between trust in the app repository and the rest of the responses in the questionnaire. The findings suggest that users who trust the app repository are more likely to: (a) be non-security savvy, and (b) have either good or moderate IT skills. Moreover, users that are unaware of smartphone malware are more likely to trust the app repository. This is alerting because the app repository may be more easily used as a malware distribution point, as users tend to be unaware of the threat and incapable of discovering it themselves.

Table 23. Associations of *TrustRep* variable

Variable Label	x^2	df	p	φ
Application testing in OAR (No)	11.410	1	0.001	-0.158
BlackBerry OS	5.323	1	0.021	0.108
IT expertise (Moderate)	5.137	1	0.023	0.106
IT expertise (Good)	11.697	1	0.001	0.160
IT expertise (Excellent)	22.518	1	< 0.001	-0.222
No use of security software	18.161	1	< 0.001	-0.199
Security messages (Always)	7.871	1	0.005	-0.131
Security messages (Sometimes)	8.196	1	0.004	0.134
Security savvy user	10.893	1	0.001	-0.154
Smartphone malware existence	16.448	1	< 0.001	-0.190
Smartphone security software existence	4.673	1	0.031	-0.101
Store personal data	5.685	1	0.017	0.111

Survey findings indicate that smartphone secsoft is more likely not to be in place, as users tend to be unaware of it. In this context, as these users were also found more likely to store personal data, the risk of unauthorized data access via malware, which is spread in the app repository increases, especially in app repositories lacking app testing. Also, users who trust the app repository are less likely to scrutinize security messages. They were found more likely to occasionally inspect them. This is a serious vulnerability of smartphone security models, as they delegate users to permit access to protected resources via these messages. Also, BlackBerry users are more likely to trust the *BlackBerry App World*, even though several available security controls for app repositories are not enabled in it (Mylonas, 2011b).

Table 24. Associations of secsoft essential

Variable Label	x^2	df	p	φ
No use of security software	23.334	1	< 0.001	-0.226
Privacy Concern	8.841	1	0.003	0.139
Searched for free smartphone security software	9.997	1	0.002	0.148
Security or Privacy	5.782	1	0.016	0.112
Security savvy user	10.803	1	0.001	0.154
Smartphone malware existence	12.322	1	< 0.001	0.164
Use device password lock	8.435	1	0.004	0.136
Use remote locator	4.749	1	0.029	0.102
Use security software in PC	10.928	1	0.001	0.154
Use security software in smartphone	77.329	1	< 0.001	0.411

Our findings suggest that users who do not trust the app repository are more likely users with excellent IT skills and security savvy. Also, they tend to be aware of smartphone malware, as well as smartphone secsoft. Finally, the users who do not use security software on any device (i.e. smartphone, PC) tend not to trust the application repository.

Users who consider smartphone secsoft essential. Survey results (Table 24) indicate that those who consider smartphone secsoft essential are more likely to use security software in their PC. Nonetheless, the results do not suggest likewise for the adoption of smartphone secsoft, an attitude also found in (Androulidakis and Kandus, 2011). Participants who use smartphone secsoft tend to regard them as essential and users who consider smartphone secsoft essential are less likely to respond that they do not use security software in any of their devices. The results revealed a positive correlation with awareness of smartphone malware and search for free smartphone security software in the app repository. Also, the users are more likely to be security savvy and concerned about their privacy. Nonetheless, the results also indicate that the users who do not regard secsoft as essential are more likely not to consider security issues during application selection. Thus, they are exposed to malicious applications, especially when app testing does not take place in the repository.

As far as the pre-installed security controls are concerned, a positive correlation was found between users who regard secsoft essential and the use of device password lock and remote device locator. On the other hand, a statistically significant positive correlation with encryption and remote data wipe was not found. As a result, unless a third-party secsoft is installed, these users are exposed to unauthorized access to their data (e.g. theft, data leakage, etc.).

Privacy concern and preference to pirated applications. Table 25 illustrates that privacy-concerned smartphone users are more likely to regard smartphone security software as essential. These users are more likely to respond that they occasionally inspect agreement messages and less likely to ignore them.

Table 25. Associations of privacy concerns

Variable Label	x^2	df	p	φ
Agreement messages (*Never*)	7.098	1	0.008	-0.124
Agreement messages (*Sometimes*)	4.735	1	0.030	0.102
Smartphone security software essential	8.841	1	0.003	0.139

The results indicate that users who prefer pirated smartphone applications tend to ignore agreement messages. In specific, we found a negative correlation with the respondents who scrutinize agreement messages and a positive correlation with the ones who occasionally inspect them (Table 26).

Table 26. Associations of prefer pirated apps

Variable Label	x^2	df	p	φ
Agreement messages (*Always*)	6.357	1	0.012	-0.118
Agreement messages (*Never*)	9.625	1	0.002	0.145

Smartphone security software existence. Survey findings suggest that users with moderate IT skills are less likely to be aware of the existence of smartphone secsoft. Also, users who are unaware of smartphone secsoft are more likely to trust the app repository and tend to be unaware of smartphone malware. Therefore, the risk of malware is considerable and, in the case that these users download a malicious application from the app repository, they will be unable to find that they are infected. The results revealed that participants who use security software tend to be aware of their existence.

Table 27. Associations of secsoft existence

Variable Label	x^2	df	p	φ
Android OS	7.480	1	0.006	0.128
Application testing in OAR (*Don't know*)	13.373	1	< 0.001	-0.171
IT expertise (*Moderate*)	12.678	1	< 0.001	-0.166
Smartphone malware existence	65.791	1	< 0.001	0.379
Trust app repository	4.673	1	0.031	-0.101
Use device password lock	21.010	1	< 0.001	0.214
Use remote locator	10.025	1	0.002	0.148
Use security software in smartphone	55.041	1	< 0.001	0.347

Moreover, the results suggest that Android users were more likely to be aware of the existence of smartphone secsoft. This may be due to the fact that a lot of smartphone malware target the Android platform (Felt et al., 2011b; Zhou et al., 2012b, Zhou and Jiang, 2012). The users who are aware of smartphone secsoft existence are more likely to have the device password protected. Also, they are more likely aware of smartphone malware and less likely unaware whether application testing takes place in the app repository. Finally, the results suggest that users who are unaware of the existence of smartphone secsoft are more likely to have the remote device locator disabled. Thus, in the case of temporal misplace or permanent device loss the device can hardly be recovered. A summary of the findings appears in Table 26.

Application testing. Survey findings (see Table 28) show that users who do not know if application testing takes place in the app repository are less likely users with excellent IT skills. Also, the users who respond that application testing takes place in the app repository are more likely to be users with excellent IT skills. Nonetheless, 42.3% of the Android users and 43.5%

of the Blackberry users who were technically savvy responded, somewhat erroneously, that apps undergo security analysis during their submission in the app repository.

Our findings suggest that users with good IT skills are more likely to respond that they do not know whether application testing takes place in the app repository. Nonetheless, users with good IT skills tend to trust the app repository, albeit they are not aware if application testing takes place in it. Moreover, users who do not use smartphone secsoft and are unaware of smartphone malware tend to be uncertain if app testing takes place in the app repository. This suggests that these users ignore endpoint security, even though they are uncertain if apps have been security analyzed in their submission. They are also ignorant about the malware threat.

The results revealed that users who are uncertain if application testing takes place in the app repository are less likely to scrutinize security messages. This suggests that ignorance of security messages is not a result of the trust in the efficiency of the app testing mechanism.

Finally, a negative correlation between iOS users and the users who respond that they do not know whether application testing occurs in the Apple's App Store was found, which was expected due to the walled garden approach employed by Apple. Nonetheless, a positive correlation with the users who answer that app testing happens in the Apple's App Store was not found.

Table 28: Associations of app testing

Variable a Label	Variable b Label	x^2	df	p	φ
Application testing in OAR (*Don't know*)	iOS	6.421	1	0.011	-0.118
Application testing in OAR (*Don't know*)	IT expertise (*Excellent*)	9.231	1	0.002	-0.142
Application testing in OAR (*Don't know*)	IT expertise (*Good*)	5.089	1	0.024	0.105
Application testing in OAR (*Don't know*)	Security messages (*Always*)	12.873	1	<0.001	-0.168
Application testing in OAR (*Don't know*)	Smartphone malware existence	12.426	1	<0.001	-0.165
Application testing in OAR (*Don't know*)	Use security software in smartphone	12.412	1	<0.001	-0.165
Application testing in OAR (*Yes*)	IT expertise (*Excellent*)	9.132	1	0.003	0.141

4.4.2.2 Associations for user practices

In the sequel we refer to the results of the inferential tests, with respect to: (a) user adoption of smartphone secsoft, (b) lack of adoption of secsoft in any user device, (c) adoption of physical security controls, (d) storage of personal and business data, and (e) examination prompted messages.

Use of secsoft in smartphone. The survey findings suggest that the smartphone users who use smartphone security software in their smartphone were also aware of their existence, considered them essential, have searched for free secsoft in the repository, and were aware of smartphone malware (see Table 29). Moreover, these users tend to respond that application testing takes place in the app repository. This suggests that users opt for both centralized security and endpoint security.

Our findings indicate that smartphone users are more likely to have encryption, device password lock, and the remote locator mechanism disabled when they do not use smartphone secsoft. This poor adoption of available security controls leaves users exposed to several smartphone threats, such as the ones listed in (Theoharidou et al., 2012). The risk of malware threat is severe, as these users were found unaware about smartphone malware. In addition, these users were more likely to be unaware if application testing takes place in the app repository. Therefore, this poor adoption of security controls seems not to stem from users' trust in the application testing effectiveness, but from users' unfamiliarity with the available smartphone threats.

Finally, the survey results indicate that users who use smartphone secsoft are more likely Android users and less likely iPhone users. Also, Android users were more likely to have searched the app repository for free secsoft (x^2=12.708, df=1, p≤0.001, φ=0.171). The former is true because, currently, more smartphone malware target Android (Felt et al., 2011b; Zhou et al., 2012b, Zhou and Jiang, 2012). For iOS users, the lack of adoption of smartphone secsoft may steam from Apple's walled garden approach.

Table 29. Associations of smartphone secsoft usage

Variable Label	x^2	df	p	φ
Android OS	11.180	1	0.001	0.156
Application testing in OAR (*Don't know*)	12.412	1	< 0.001	-0.165
Application testing in OAR (*Yes*)	19.261	1	< 0.001	0.205
iOS	7.399	1	0.007	-0.127
No use of security software	11.512	1	0.001	-0.159
Searched for free smartphone security software	57.785	1	< 0.001	0.355
Smartphone malware existence	19.486	1	< 0.001	0.206
Smartphone security software essential	77.329	1	< 0.001	0.411
Smartphone security software existence	55.041	1	< 0.001	0.347
Use device password lock	7.252	1	0.007	0.126
Use encryption	20.782	1	< 0.001	0.213
Use remote locator	26.787	1	< 0.001	0.242

No adoption of security software. The results that appear in Table 30 indicate that users who do not use security software on any device (i.e. either PC or smartphone) tend to believe that

smartphone security software is not essential (and also tend not to use them). Also, these users are more likely not to trust the app repository. Nonetheless, any correlation with the rest users' security practices, especially regarding the adoption of pre-installed security controls and attitude towards security messages, was not found. Thus, a safe deduction about their level of awareness cannot be performed.

Table 30. Associations of users who do not use secsoft in any device

Variable Label	x^2	df	p	φ
Smartphone security software essential	23.334	1	< 0.001	-0.226
Trust app repository	18.161	1	< 0.001	-0.199
Use security software in smartphone	11.512	1	0.001	-0.159

Associations of physical security controls. Tables 33-36 summarize the statistically significant correlations of the answers regarding the 'pre-installed' security controls (i.e. encryption, device password lock, remote data wipe, remote device locator) with the rest of the answers in the questionnaire. The results suggest that the technical skills of the respondents tend to affect the adoption of security controls. In specific, users were found to enable encryption, remote data wipe, and remote device locator when they tend to be technically savvy. A similar correlation was not found in the adoption of device password lock. Our results indicate that users who do not password-protect their devices are more likely to be non-security savvy. This is a correlation which was not found in the other security controls. Also, the results revealed that participants are more likely to disable together the security controls: encryption, remote data wipe, and remote device locator. In this case, the risk of unauthorized physical access is severe (e.g. theft, data leakage, etc.).

Users having good IT skills tend not to encrypt their data. The respondents who encrypt their data are more likely aware of smartphone malware and use device password lock. The results suggest that these users are more likely to have searched the app repository for free smartphone secsoft. Moreover, the survey results indicate that participants who do not encrypt their data are more likely not to use smartphone secsoft, be unaware about smartphone malware, as well as not search for free smartphone secsoft in the app repository. Thus, the risk of a remote unauthorized access attack is serious. Also, users who either occasionally inspect security messages, or ignore them at all, are more likely to disable encryption. Thus, they are exposing their data to remote unauthorized access. In contrast, users who always inspect security messages are more likely to encryption their data.

Table 31. Associations of user with encryption enabled

Variable Label	x^2	df	p	φ
IT expertise (*Excellent*)	9.198	1	0.002	0.142
IT expertise (*Good*)	5.048	1	0.025	-0.105
Searched for free smartphone security software	26.123	1	< 0.001	0.239
Security messages (*Always*)	37.705	1	< 0.001	0.287
Security messages (*Never*)	6.352	1	0.012	-0.118
Security messages (*Sometimes*)	18.335	1	< 0.001	-0.200
Smartphone malware existence	7.121	1	0.008	0.125
Use device password lock	17.609	1	< 0.001	0.196
Use remote data wipe	6.749	1	0.009	0.121
Use remote locator	9.954	1	0.002	0.147
Use security software in smartphone	20.782	1	< 0.001	0.213

The survey results suggest that users who tend to disable device password lock are less likely to use smartphone secsoft and tend to disable remote data wipe, remote locator as well as encryption. Thus, they are exposed to physical unauthorized access attacks without any relevant control in place. The users who password-protect their device are more likely to be aware about the existence of smartphone malware and smartphone secsoft and consider the latter as essential. Nonetheless, our results do not indicate that they also tend to use smartphone secsoft. Also, users who do not password lock their devices are more likely not to use smartphone secsoft and not have searched for free secsoft in the app repository. In this case, the device and its data are exposed to unauthorized access attacks.

Table 32. Associations of use device password lock

Variable Label	x^2	df	p	φ
Searched for free smartphone security software	27.105	1	< 0.001	0.243
Security savvy	5.743	1	0.017	0.112
Smartphone malware existence	19.852	1	< 0.001	0.208
Smartphone security software essential	8.435	1	0.004	0.136
Smartphone security software existence	21.010	1	< 0.001	0.214
Use encryption	17.609	1	< 0.001	0.196
Use remote data wipe	15.773	1	< 0.001	0.186
Use remote locator	18.775	1	< 0.001	0.202
Use security software in smartphone	7.252	1	0.007	0.126

Survey results show that users with good IT skills tend not to enable remote date wipe. Users who do not enable remote data wipe are less likely to enable device password lock and remote device locator. In this case, the protection against unauthorized access attacks depends on the

strength of device password lock mechanism. In smartphones this mechanism may suffer, apart from the traditional attacks against device passwords (e.g. guessing, shoulder-surfing, etc.), from smudge attacks when graphical passwords are used (Aviv et al., 2010).

Table 33. Associations of remote data wipe control

Variable Label	x^2	df	p	φ
IT expertise (*Excellent*)	9.121	1	0.003	0.141
IT expertise (*Good*)	7.723	1	0.005	-0.130
Use device password lock	15.773	1	< 0.001	0.186
Use encryption	6.749	1	0.009	0.121
Use remote locator	161.497	1	< 0.001	0.594

Participants with good IT skills are less likely to enable remote locator in their devices. Users who enable remote locator tend to be aware of the existence of smartphone malware and security software. Nonetheless, our results do not suggest that they tend to use smartphone secsoft in their device, even though they tend to consider them essential. On the contrary, our results revealed that participant are more likely not to adopt both remote device locator and smartphone secsoft. As far as the pre-installed security controls are concerned, users who enable the remote device locator control are more likely to have the device password control enabled, as well as the remote data wipe. As a result, again the protection against unauthorized access relies on the strength of the device lock mechanism.

Table 34. Associations of remote locator

Variable Label	x^2	df	p	φ
IT expertise (*Excellent*)	10.505	1	0.001	0.151
IT expertise (*Good*)	6.983	1	0.008	-0.123
Smartphone malware existence	7.598	1	0.006	0.129
Smartphone security software essential	4.749	1	0.029	0.102
Smartphone security software existence	10.025	1	0.002	0.148
Use device password lock	18.775	1	< 0.001	0.202
Use encryption	9.954	1	0.002	0.147
Use remote data wipe	161.497	1	< 0.001	0.594
Use security software in smartphone	26.787	1	< 0.001	0.242

Storage of personal and business data. The associations of users who store personal data in their device are summarized in Table 35. A significant association with the controls that thwart unauthorized access to data was not found. This may result in the unauthorized exposure of sensitive data when these controls are not enabled, as it was also found in (Androulidakis and Kandus, 2011). This was also true with the users who have misplaced their device.

Table 35. Associations of personal data

Variable Label	x^2	df	p	φ
Security messages (*Always*)	5.118	1	0.024	-0.106
Security messages (*Sometimes*)	7.514	1	0.006	0.128
Smartphone malware existence	4.545	1	0.033	-0.100
Store business data	45.775	1	< 0.001	0.316
Trust app repository	5.685	1	0.017	0.111

Survey results suggest that users who store business data in their devices are most likely to also store personal data. This indicates that the same device is used for personal and business purposes, and, as a result, the impact of unauthorized access to the data is greater. The results also reveal that users who store personal data are more likely to: (a) trust the app repository, and (b) occasionally inspect security messages. This is a notable finding, as the likelihood that a user installs from the app repository a privacy breaching application that gains unauthorized access to her data is serious. Finally, users who are unaware of smartphone malware are more likely to store personal data on their device, being exposed to unauthorized access to their data. Such an attitude was also found in (Androulidakis and Kandus, 2011), where users that stored personal data did not follow security practices, e.g. backups.

4.4.2.3 Security and agreement messages

Our findings reveal that users who always check an app's security messages are more likely to have excellent IT skills. On the other hand, users with moderate IT skills are more likely to respond that they sometimes inspect security messages. This is alerting, as users who are not technically savvy are not able to make the appropriate security decisions, which is not in accordance with the expectations of permission-based smartphone security models and, thus, shows a serious vulnerability. Also, users who occasionally check security messages are more likely to be non-security savvy users, while those who scrutinize them are tend to be security savvy. This suggests, in contrast to what permission driven security models (e.g. Android) assume, that users who are non-security savvy cannot make appropriate security decisions. Users who scrutinize security messages tend to be aware of smartphone malware and smartphone secsoft.

The analysis of the correlation between agreement messages and security messages revealed that: (a) users who always check security messages are less likely to respond that they ignore agreement messages, (b) users that scrutinize agreement messages are less likely to respond that ignore security messages, (c) users who scrutinize agreement messages are more likely to always scrutinize security messages too. Our findings also revealed that users who never check security messages are more likely to never check agreement messages too. This

suggests that some smartphone users totally disregard any prompt from applications. Thus, different methods to communicate security authorization are needed, such as visual notifications (Howell and Schechter, 2010), especially in the case of permission driven security models. Also, it should be noted that this user behaviour, where users click through security messages, is known to exist in other platforms, e.g. Windows User Account Control Dialogs (Motiee et al., 2010), in browser messages that warn about invalid SSL certificates and phishing scams (Egelman et al., 2008; Sunshine et al., 2009). In this context, it is surprising that some smartphone security models delegate users to make all authorization decisions for app access requests. Moreover, our results indicate that users who ignore security messages tend to be unaware whether apps undergo security analysis during their submission. The survey demonstrates that this security complacency does not stem from user trust in the efficiency of the app testing mechanism used by the app repository. These findings are depicted in Table 36.

Table 36. Associations of security and agreement messages

Variable a Label	Variable b Label	x^2	df	p	φ
Agreement messages (*always*)	Security messages (*Always*)	59.801	1	< 0.001	0.361
Agreement messages (*always*)	Security messages (*Never*)	7.709	1	0.005	-0.130
Agreement messages (*never*)	Security messages (*Always*)	52.431	1	< 0.001	-0.338
Agreement messages (*never*)	Security messages (*Never*)	57.022	1	< 0.001	0.353
Security messages (*always*)	IT expertise (*Excellent*)	16.578	1	< 0.001	0.190
Security messages (*always*)	IT expertise (*Good*)	4.632	1	0.031	-0.101
Security messages (*always*)	IT expertise (*Moderate*)	10.330	1	0.001	-0.150
Security messages (*always*)	Security savvy user	8.097	1	0.004	0.133
Security messages (*always*)	Smartphone malware existence	8.554	1	0.003	0.137
Security messages (*always*)	Smartphone security software existence	6.628	1	0.010	0.0120
Security messages (*never*)	Application testing in OAR (*Don't know*)	5.264	1	0.022	0.107
Security messages (*sometimes*)	IT expertise (*Excellent*)	9.100	1	0.003	-0.141
Security messages (*sometimes*)	IT expertise (*Moderate*)	8.250	1	0.004	0.134
Security messages (*sometimes*)	IT expertise (*Moderate*)	8.250	1	0.004	0.134

Security messages (*sometimes*)	Security savvy user	6.485	1	0.011	-0.119

4.4.2.4 Application selection criteria

Survey results suggest that users who inspect an app's reputation during application selection are more likely to be security savvy users. On the other hand, users who are non-security savvy are more likely not to examine the reputation of an app during application selection.

Our findings indicate that users who select usefulness as an app selection criterion are more likely to ignore the app's review. Security-wise, this is important as in app repositories where the security model is community driven, e.g. in Android, the reviews often give indications whether an app is malicious or not. In the same context, users are more likely to ignore the app's reputation. Reputation is important in any commodity market, even thought the literature has identified shortcomings of reputation systems (Mármol and Pérez, 2009). In the context of app repositories, an application with good reputation does not necessary respect the user's privacy, but reputation often poses an additional line of defence against malevolent users. An app's reputation may be low because several users have found it to be malicious or permission-hungry, i.e. requesting more permissions than those that are expected from its functionality, suspiciously draining resources (e.g. CPU, battery). Finally, the users who consider security or privacy issues during application selection are more likely to be security savvy users and regard smartphone security software essential. The above are depicted in Table 37.

Table 37. Associations of application selection criteria

Variable a Label	Variable b Label	x^2	df	p	φ
Reputation	Security savvy user	6.319	1	0.012	0.117
Security or Privacy	Smartphone security software essential	5.782	1	0.016	0.112
Security or Privacy	Security savvy user	12.949	1	< 0.001	0.168
Usefulness	Reputation	12.219	1	< 0.001	-0.163
Usefulness	Reviews	6.270	1	0.012	-0.117

4.4.3 Response diversity

This section focuses on the two groups' diversity of responses. Our analysis revealed statistically significant differences in the two groups' responses in the questions that are presented in Table 38.

As one would initially assume, significantly less *SECSAV* users were unaware of smartphone malware ($x^2=8.623$, $p=0.003$). Also, considerably more *SECSAV* users regard smartphone security software (hereto 'secsoft') as essential ($x^2=10.803$, $p=0.001$) and pass-

word protect their devices (x^2 =5.743, p=0.017). Regarding the security messages, significantly more *SECSAV* users scrutinized them (x^2=8.097, p= 0.004), while significantly more *NSECSAV* users occasionally (i.e. sometimes) inspected them (x^2= 6.485, p= 0.011). However, this attitude is orthogonal with the expectations of smartphone platforms, in which users make informed decisions by scrutinizing any displayed security messages (Mylonas et al., 2011a).

The results revealed that the majority of *NSECSAV* users consider apps that are indexed in the app repository as secure for installation in their devices (hereinafter this is referred to as 'trust in the app repository'). This trust in the repository is a serious vulnerability according to our previous analysis. Significantly less (x^2=10.893, p=0.001) - but still a majority of - *SECSAV* users trust the app repository. Moreover, even though significantly more *SECSAV* users considered the reputation (x^2=6.319, p=0.012) and security and privacy issues (x^2= 12.949, p<0.001) during app selection, they were the minority of users in both groups. This clearly suggests that current risk signals in app repositories are not effective.

Finally, the results indicate that significantly more *SECSAV* users are technically savvy (x^2=20.566, p<0.001), while significantly more *NSECSAV* users have good (x^2= 11.257, p=0.001), or moderate (x^2=4.102, p=0.043) IT skills.

Table 38. Differences in responses between user groups

Question	SECSAV	NSECSAV
IT skills (*excellent*), (Q3c)	60.5%	39.1%
IT skills (*good*) (Q3b)	32.5%	48.1%
IT skills (*moderate*) (Q3a)	7.0%	12.8%
Privacy or security app criterion (Q8e)	7.0%	0.8%
Reputation app criterion (Q8d)	12.5%	5.8%
Security messages (*always*) (Q11c)	46.0%	32.9%
Security messages (*sometimes*) (Q11b)	41.5%	53.5%
Smartphone malware existence (Q16)	87.5%	76.7%
Smartphone security software essential (Q19)	74.0%	59.3%
Trust app repository (Q9)	68.5%	81.8%
Use device lock (Q20b)	70.5%	59.7%

4.4.4 Correlation diversity

This section focuses on the two groups' correlation diversity in the user responses. Tables 49-54 include the details of the tests of independence in each group and also summarize the findings of each subsection section. We organize our findings into: (a) adoption of smartphone security software, (b) adoption of physical controls, (c) trust in app repository, (d) inspection of security messages, (e) user's background influences their security.

4.4.4.1 Adoption of smartphone security software

The analysis revealed that users in *both groups* ignore endpoint security (*Finding 1, Fd1*), as: (a) they regard smartphone secsoft essential, but tend to use them only in their PC and not in their smartphone ($Fd1_a$). This finding shows a clear asymmetry of security awareness in the two platforms (i.e. PC, smartphone). Our results revealed that (b) even though users in *both groups* ignore endpoint security they tend to be unsure if submitted apps in the repository undergo security analysis ($Fd1_b$). Also, (c) iOS *SECSAV* users tend to regard that smartphone secsoft is not essential ($Fd1_c$). This erroneous security posture, which may stem from Apple's *walled garden* approach (and marketing strategy), has been proven invalid since malware has been found in iOS (Egele et al.,2011).

On the other hand, we found that some participants - the minority of the sample, as discussed beforehand- opt for endpoint security (Fd2): (a) users in both groups tend to use smartphone secsoft when they respond that app testing takes place in the repository (*Fd2a*), thus not relying solely on the centralized protection, (b) *SECSAV* users consider smartphone secsoft essential and search for free secsoft in the app repository (Fd2b), implying that these users try to protect their devices, and (c) *NSECSAV* Android users tend to use smartphone secsoft ($Fd2_C$), which may result from the fact that various smartphone malware families target Android (Zhou and Jiang, 2012).

NSECSAV users who inspect an app's reputation are less likely to regard smartphone secsoft as essential (*Fd3*). This is a rather interesting finding, since smartphone secsoft can be used as an additional line of defense against malware, especially the ones from mediocre attackers [16-17]. Reputation score alone cannot guarantee that an app is benign, as popular apps do not necessarily respect a user's privacy (Chia et al., 2012). The aforementioned are summarized in the following table.

Table 39: Adoption of smartphone security software

#	Finding (*Fd*)	SECSAV $(x^2,p,\varphi)_1,..,(x^2,p,\varphi)_n$ †	NSECSAV $(x^2,p,\varphi)_1\ldots(x^2,p,\varphi)_n$
1.	Users ignore endpoint security	$(4.863,0.027,0.156)_a$, $(5.654,0.017,-0.168)_b$, $(4.056,0.044,-0.142)_c$	$(4.885,0.027,0.138)_a$, $(6.735,0.009,-0.162)_b$
2.	Minority of users opt for endpoint security	$(8.344,0.004,0.37)_a$, $(8.337,0.004,0.204)_b$	$(7.975,0.005,0.199)_a$, $(7.594,0.006,0.172)_C$
3.	Users who inspect app's reputation tend to not regard smartphone secsoft essential	-	(4.450,0.035,-0.131)

† x^2 test of independence value, significance level, association's direction (positive, negative).

4.4.4.2 Unauthorized access

Our results (see Table 40) revealed that users in *both groups* are exposed to unauthorized *remote access* (*Fd4*). This is because they were found not to: encrypt their data, use third-party secsoft, and scrutinize security messages. Thus, if they grant malware or greyware access to sensitive resources or an attacker gains unauthorized remote access via vulnerability exploitation, then their resources will be unprotected. Specifically: (a) users who do not use smartphone secsoft are less likely to encrypt their data ($Fd4_a$) and (b) users who occasionally read security messages (i.e. responded sometimes) are less likely to encrypt their data ($Fd4_b$). Also, we found *SECSAV* users who (c) are unaware of smartphone malware and are less likely to encrypt their data ($Fd4_c$) and (d) tend to have encryption and remote wipe disabled ($Fd4_d$). Finally, *NSECSAV* users who ignore security messages are less likely to encrypt their data ($Fd4_e$).

Furthermore, our results revealed multiple cases where users in *both groups* are exposed to unauthorized physical access (*Fd5*). This happens since users do not adopt physical security controls (i.e. device password, encryption, remote device locator, remote wipe) and/or third-party security software, which can proactively (e.g. encryption), or reactively (e.g. remote wipe) protect against this threat. Specifically: (a) users are less likely to use device password lock when they are not using encryption ($Fd5_a$), (b) users who do not password protect their devices are less likely to enable remote wipe ($Fd5_b$), (c) users who do not password protect their devices tend not to use the remote locator ($Fd5_c$), (d) users tend to have the security mechanisms remote wipe and remote locator both disabled ($Fd5_d$), and (e) users who do not use smartphone secsoft are less likely to encrypt the data ($Fd5_e$). Moreover, our results revealed additional occasions where *NSECSAV* users were exposed to physical access, namely: (f) users tend to disable both device encryption and remote device locator ($Fd5_f$), (g) users who do not password protect their device are less likely to use smartphone secsoft ($Fd5_g$), and users who were ignorant about smartphone secsoft were found not to h) password protect their devices ($Fd5_h$), and i) use encryption ($Fd5_i$).

The analysis revealed cases where the impact of unauthorized access attacks in the *NSECSAV* groups, increases (*Fd6*) since users who: (a) do not encrypt their data tend to store personal data in their devices ($Fd6_a$), (b) store personal data are less likely to scrutinize security messages ($Fd6_b$), (c) store personal data are less likely aware of smartphone malware ($Fd6_c$), and (d) do not enable remote device locator tend to have misplaced their device in the past ($Fd6_d$).

Table 40: Correlations for unauthorized access

#	Finding (*Fd*)	SECSAV $(x^2,p,\varphi)_1,..,(x^2,p,\varphi)_n$ †	NSECSAV $(x^2,p,\varphi)_1 \ldots (x^2,p,\varphi)_n$
4	Users are exposed to unauthorized remote access	$(5.770;0.016;0.170)_a$, $(14.305,p<0.001,-0.267)_b$, $(6.491, 0.011, 0.180)_c$, $(5.931,0.015, 0.172)_d$	$(15.978,p<0.001,0.249)_a$,$(5.112,0.024,-0.141)_b$, $(5.878,0.015, -0.151)_e$
5	Users are exposed to unauthorized physical access	$(5.4145,0.020,0.165)_a$,$(10.525,0.001,0.229)_b$, $(8.776,0.003, 0.209)_c$, $(34.333, p<0.001, 0.414)_d$, $(5.770,0.016,0.170)_e$,	$(11.984,0.001,0.216)_a$,$(6.828,0.009,0.163)_b$, $(9.959,0.002, 0.196)_c$, $(137.857, p<0.001, 0.731)_d$, $(15.978,p<0.001, 0.249)_e$, $(7.731, 0.005,0.173)_f$, $(8.292,0.004, 0.179)_g$, $(13.001, p<0.001,0.224)_h$, $(4.472,0.034, 0.132)_i$
6	Increased impact of unauthorized access	-	$(4.593,0.032,-0.133)_a$, $(6.889, 0.009,-0.163)_b$, $(6.643,0.010,-0.160)_c$, $(7.755,0.005,-0.173)_d$, $(8.768,0.003,0.184)_e$
7	Users who scrutinize security messages opt for physical security controls	$(19.715, p<0.001,0.314)_a$, $(8.085,0.004,0.201)_b$	$(17.352, p<0.001, 0.259)_a$,

† x^2 test of independence value, significance level, association's direction (positive, negative).

Our analysis revealed that users who scrutinize security messages opt for physical security controls (*Fd7a*), since: a) users in *both groups* who scrutinize security messages tend to encrypt their data (*Fd7a*) and (b) *SECSAV* users who scrutinize security messages tend to password protect their smartphone (*Fd7b*). Thus, these users have an encouraging security posture against unauthorized access.

4.4.4.3 Trust in the app repository

Our previous results revealed the trust in the app repository as a severe vulnerability, as it lowered the sample's security posture. Our further analysis revealed that this trust may expose users to malware/greyware residing in the repository (Enck et al., 2011; Zhou et al., 2012) (*Fd8*). This is so, since: (a) users in *both groups* who are unaware of smartphone malware tend to trust the app repository ($Fd8_a$), (b) *SECSAV* users who trust the repository tend to occasionally inspect the prompted security messages ($Fd8_b$), and (c) *NSECSAV* users who trust the repository are less likely to search the app repository for free secsoft ($Fd8_c$). The impact in case a *NSECSAV* user installs malware/greyware from the repository increases, because *NSECSAV* users who trust the repository tend to store personal data in their devices ($Fd6_e$).

Furthermore, the results suggest that *NSECSAV* users trust the app repository, but tend to be unaware whether apps are securely analyzed during their submission (*Fd9*). In this context, the results suggest that the trust in the repository does not stem from the perceived efficiency of the app vetting mechanism. On the other hand, our analysis revealed some *NSECSAV* users who are not complacent with security provided by the app repository and try to amend it. This is so, as users who do not trust the app repository tend to search in it for free secsoft (*Fd10*).

Table 41 summarizes the results of this subsection.

Table 41: Correlations for trust in the app repository

#	Finding (*Fd*)	SECSAV $(x^2,p,\varphi)_1,..,(x^2,p,\varphi)_n$ †	NSECSAV $(x^2,p,\varphi)_1 \ldots (x^2,p,\varphi)_n$
8	Users are exposed to malware indexed in the app repository	$(5.035,0.025,-0.159)_a$, $(9.824, 0.002,0.222)_b$,	$(9.167,0.002,-0.188)_a$, $(5.337,0.021,-0.144)_c$
9	Users trust the app repository but are unaware of app testing	-	(5.057,0.025, 0.140)
10	Uses who do not trust app repository tend to search in it for free secsoft	-	(5.337,0.021,-0.144)

† x^2 test of independence value, significance level, association's direction (positive, negative).

4.4.4.4 Inspection of security messages

The security models of smartphone platforms assume that users scrutinize any messages that they prompt, in order to make informed security decisions. Our results, which are presented in Table 42, suggest that in *both groups*, users exist who scrutinize security messages and tend to scrutinize agreement messages (*Fd11*). Although this suggests that these users are concerned about their security and privacy, they are the sample's minority as discussed in the previous subsection.

On the other hand, there are users in *both groups* who ignore prompted messages (*Fd12*). Namely: (a) there are users who ignore both security and agreement messages ($Fd12_a$) and (b) users who occasionally inspect security messages are less likely to scrutinize agreement messages ($Fd12_b$). Also, *NSECSAV* users who ignore security messages tend to be uncertain if apps are securely tested during their submission (*Fd13*). This indicates that the ignorance of security messages is not a result of the trust in the efficiency of the app testing mechanism.

Table 42: Correlations for inspection of security messages

#	Finding (*Fd*)	SECSAV $(x^2,p,\varphi)_1,..,(x^2,p,\varphi)_n$ †	NSECSAV $(x^2,p,\varphi)_1 \ldots (x^2,p,\varphi)_n$
11	Minority of users scrutinize prompted messages	$(21.480, p<0.001, 0.328)_a$	$(40.338, p<0.001,0.395)_a$
12	Users ignore prompted messages	$(22.688,0.000,0.337)_a$, $(12.195,p<0.001, -0.247)_b$,	$(34.262,0.000,0.364)_a$, $(20.452,p<0.001,-0.282)_b$,
13	Users who ignore security messages are uncertain if apps undergo security analysis	-	(7.297,0.007, 0.168)

† x^2 test of independence value, significance level, association's direction (positive, negative).

4.4.4.5 User's background influences their security posture

The results suggest that user's security posture is influenced by their IT skills (*Fd14*). Firstly, in *both groups*: (a) technically savvy users are less likely to trust the app repository ($Fd14_a$), and (b) users who scrutinize security messages tend to be technically savvy ($Fd14_b$). In *SECSAV*, (c) non-technically savvy users tend to occasionally inspect security messages ($Fd14_c$). In *NSECSAV* the results suggest that: (d) non-technical savvy users tend to be unsure if the repository's apps are security analyzed ($Fd14_d$), whereas (e) technically savvy ones are less likely to respond that they do not know if apps undergo security analysis ($Fd14_e$). Also, (f) non-technical savvy users are less likely to search the repository for free secsoft ($Fd14_f$) and (g) users who inspect app reviews tend to be technically savvy ($Fd14_g$). Finally, participants who use: (h) remote wipe, and (i) remote locator, tend to be technically savvy, ($Fd14_h$) and ($Fd14_i$) respectively.

Furthermore, the results revealed users in *both groups* who are ignorant of smartphone secsoft and smartphone malware (*Fd15*). This is alarming, since in both groups there are users exposed to smartphone malware/greyware.

Finally, our results suggest that *NSECSAV* users disregard important aspects of security in the smartphone app ecosystem (*Fd16*), as users who are unaware if app testing happens in the app repository tend to: (a) ignore security messages ($Fd16_a$), (b) ignore agreement messages ($Fd16_b$), and (c) be unaware of smartphone malware ($Fd16_c$).

#	Finding (*Fd*)	SECSAV $(x^2,p,\varphi)_1,..,(x^2,p,\varphi)_n$ †	NSECSAV $(x^2,p,\varphi)_1 \ldots (x^2,p,\varphi)_n$
14	User's security posture is influenced by IT skills	$(6.029,0.014,-0.174)_a$, $(4.538, 0.033,0.151)_b$, $(12.122, 0.000,0.246)_c$	$(12.272,p<0.001,-0.218)_a$, $(8.471,0.004, 0.181)_b$, $(5.853, 0.016,0.151)_d$, $(8.839,0.003,-0.185)_e$, $(8.628,0.003, 0.183)_f$, $(12.093,0.001,0.217)_g$, $(12.051,0.001,0.216),2)_h$, $(16.502,p<0.001,0.253)_i$
15	Users are ignorant of smartphone secsoft and smartphone malware	$(41.976, p<0.001, 0.458)_a$	$(31.629, p<0.001,0.350)_a$
16	Users disregard security notions in the smartphone app ecosystem	-	$(7.297,0.007,0.168)_a$, $(10.377,0.001,0.201)_b$, $(19.988, p<0.001,-0.278)_c$

† x^2 test of independence value, significance level, association's direction (positive, negative).

4.4.4.6 Other findings

The results revealed users in *both groups* who store business data in their device and also tend to store personal data in it (*Fd17*). This suggests that the device is being used for business and personal purposes, thus the impact of unauthorized access increases.

In addition, the results suggest that users disregard security indicators during app selection (*Fd18*), as: (a) users in *both groups* who consider an app's usefulness tend to disregard its reputation score during app selection ($Fd18_a$) and (b) *NSECSAV* users who consider an app's usefulness tend to disregard its reviews during app selection ($Fd18_b$). As discussed beforehand, the descriptive statistics indicate that usefulness is the most popular app selection criterion. Therefore, these negative correlations suggest that users tend to disregard the two risk signals. The above-mentioned are summarized in the following table.

Table 43: Rest correlations found in the subgroups

#	Finding (*Fd*)	SECSAV $(x^2,p,\varphi)_1,..,(x^2,p,\varphi)_n$ †	NSECSAV $(x^2,p,\varphi)_1 \ldots (x^2,p,\varphi)_n$
17	The device is used for personal and business purposes	(19.682, p<0.001,0.314) $_a$	(27.087, p<0.001,0.324) $_a$
18	Users disregard security indicators during app selection	(5.283,0.022,-0.163) $_a$	(4.584,0.032,-0.133)$_a$, (11.194,0.001,-0.208)$_b$

† x^2 test of independence value, significance level, association's direction (positive, negative).

4.5 Overview and discussion

This section presents an overview and discussion of the results of the user study. Its presentation is organized into two subjections; the first for the analysis of the whole sample and the second for the analysis of the SECSAV and NSECSAV groups, respectively. Finally, this subsection ends with a discussion of the survey's limitations.

4.5.1 Findings from analysis of whole sample

The smartphone platforms' security models delegate users to make security decisions while downloading apps from official app repositories. This delegation is heterogeneous, ranging from simply authorizing access to some protected resources, to authorizing the user to deduce whether an app may impair her security and privacy. The survey findings (see Table 44) are not in-line with the expectation of smartphones' security models for a reasonable security aware user. In contrast, they suggest that users are not adequately prepared to make appropriate security decisions.

The findings of the survey analysis show that the majority of smartphone users believe that downloading applications from the app repository is risk-free. It is unclear why this security complacency exists, as the majority of smartphone users who trust the app repository were unaware whether application testing mechanism is enabled in it. Two factors may be decisive in misleading users. First, the fact that an app is distributed from an official app repository may mislead the users into believing that the app is secure or that security controls exist in the app repository. Second, smartphone users may not be able to realize that their device is not

just a telephone. Users who use mobile telephony - and in particular those using fixed telephony - have been accustomed to use voice services in a context with limited threats. In contrast to feature phones and fixed landline devices, smartphones combine the capabilities of a feature phone and a handheld computer and, as a result, it is exposed to the union of threat targeting both devices (Theoharidou et al., 2012).

On the contrary with what popular permission-driven smartphone security models (e.g. Android, Symbian, Windows Phone, etc.) may expect, our findings suggest that users tend to ignore security messages that are prompted to them. Also, some users were found to be more likely to ignore all messages that an application displays. This is a vulnerability that violates the trust model of smartphone security models. This is since these models assume that smartphone users will thoroughly inspect these messages in order to make correct decisions that affect their privacy and security. It should be noted that this user behaviour, where users click through security messages, is known to exist in other platforms, e.g. Windows UAC (Motiee et al., 2010), browser messages that warn about invalid SSL certificates and phishing scams (Egelman et al., 2008; Sunshine et al., 2009). Moreover, users are trained to click through interruptive messages, which appear while they are completing a task (Motiee et al., 2010), as well as tend to ignore consequent warnings, especially the ones which resemble EULAs. This may partially explain why users did not scrutinize security messages that smartphones prompt them. The importance of this vulnerability is increased due to permission overdeclaration of smartphone applications, i.e. developers tend to request access to more permissions than needed for the API they use (Felt et al., 2011a). This permission overdeclaration may be exploited by deputy attacks (Felt et al, 2011c, Grace et al., 2011), or by malicious advertisements (Shekhar et al., 2012) that share the permissions of an application.

Our analysis shows that only technically and security savvy users tend to inspect these messages. This is also important, as smartphone devices are becoming more widespread and thus more users who are not security or technical savvy tend to use them. Moreover, the findings show that user security or privacy (e.g. protection from surveillance, unauthorized access to personal data) is not a criterion during app selection from the official repository.

Smartphone users were found exposed to several security and privacy threats, as the majority of the available security controls were disabled. Moreover, the results suggest that the respondents' technical skills tend to affect the adoption of security controls. In specific users were found to enable encryption, remote data wipe and remote device locator when they tend to be technically savvy. A similar correlation was not found with the use of device password lock. This is expected since this control is available in feature phones and users have been, in general, trained to use simple authentication mechanisms. Nonetheless, our results indicate that users who do not password protect their devices are more likely to be non-security savvy - a correlation which was not found in the other security controls.

Table 44. Main findings of the survey

Q1: Do smartphone users enable security controls on their devices?
Finding 1: Smartphone secsoft is poorly adopted. (24.5% of sample)
Finding 2: Smartphone secsoft are considered not essential (34.3%)
Finding 3: Most users have not searched the repository for free smartphone secsoft (60%)
Finding 4: Few users are unaware of the smartphone secsoft existence (27%)
Finding 5: Poor adoption of 'pre-installed' security controls
Finding 5.1: Encryption (22.7%)
Finding 5.2: Remote data wipe (15.1%)
Finding 5.3: Remote device locator (23.1%)
Finding 5.4: No adoption of any pre-installed security control (27.9%)
Finding 6: Users tend to have disabled smartphone secsoft along with encryption, device password lock and remote device locator
Q2: Do users consider security while choosing and downloading applications?
Finding 1: Only a few users scrutinize security messages (38.6%)
Finding 2: Only a few users scrutinize agreement messages (10%)
Finding 3: Users prefer pirated apps (60.7%)
Finding 4: Users disregard security during app selection
Finding 4.1: Only a few users read reviews (10.5%).
Finding 4.2: Only a few users inspect reputation (8.7%).
Finding 4.3: Only a few users consider security or privacy issues (3.5%).
Finding 5: Users who occasionally inspect security messages or ignore them at all are more likely to disable encryption
Finding 6: Users who always inspect security messages are more likely technically and security savvy users
Finding 7: Users who ignore security messages are more likely to also ignore agreement messages
Finding 8: Non-security savvy users are more likely not to inspect app's reputation
Finding 9: Users with usefulness as app selection criterion tend to ignore app reviews and reputation
Finding 10: Users who consider security or privacy issues during app selection are more likely security aware users
Q3: Do smartphone users trust the app repository?
Finding 1: Users believe that downloading apps from the repository is secure (approximately 3/4 users in each platform).
Finding 2: Misconceptions were found regarding app testing in the repository
Finding 2.1: Users were unaware if app testing takes place in the app repository (~half of the sample population)
Finding 2.2: Users expected app testing in repositories that did not use it
Finding 2.3: Users reported that app testing was not applied in 'walled gardened' repositories
Finding 3: Users were found exposed to malicious applications:
Finding 3.1: Users who trust app repository are more likely non security and IT savvy
Finding 3.2: Unaware users of smartphone malware more likely trust the app repository
Finding 3.3: Users who trust the repository tend to be unaware about smartphone secsoft and tend not use them
Finding 4: Users who trust app repository are less likely to always inspect security messages
Finding 5: Users with good IT skills more likely to trust app repository, regardless that they tend to be unaware whether app testing takes place in it
Other findings:
Finding 1: Users who store business data are more likely to store personal data
Finding 2: Users unaware about smartphone malware are more likely to store personal data
Finding 3: Users who store personal data are more likely to trust the app repository and occasionally inspect security messages

Our results indicate that users tend to use the same device both for business and personal purposes. Therefore, the impact of unauthorized access to stored data is greater. Moreover, the majority of smartphone users did not install any third-party security software on their device. However, these users reported that they facilitate security software in their PCs. This denotes a clear asymmetry in the security awareness of users, between the PC and smartphone platform. Furthermore, a considerable number of users reported that smartphone security software is not essential. This attitude towards secsoft may stem from technological parameters, e.g. battery consumption, overall device performance (i.e. delays), false positives and/or cognitive limitations (i.e. unawareness of existing threats and available controls), and/or psychological parameters, i.e. the trust in the application market itself. However, smartphone security software adds an additional line of defence against malware and can protect non-security savvy users at least from the threat of non-sophisticated attackers (Mylonas et al., 2011a).

4.5.2 Findings from subgroups of the sample

In this chapter, we extended our analysis on smartphone security awareness by examining whether a generic security background, deriving from sources either academic (e.g., university security courses) or industrial (e.g., information security certifications), affects the survey participants' security awareness. For doing so, we splitted the sample in two groups, comprising of security savvy users and non-security savvy users and then examine and compare their security posture.

One would expect that if the security background affected the security awareness of smartphone users, then the security findings, which were presented in the previous sections, would be splitted between SECSAV and NSECSAV users. More specifically, the findings of our analysis that suggest a user who is concerned about her security and privacy – they are marked in Table 45 with the symbol '☑' – would be mostly found in the SECSAV group and not in the NSECSAV group. Similarly, the majority of findings that suggest a user who is not concerned about her security and privacy – these findings are marked with '☒' - would be absent or infrequent in the SECSAV users and would appear more often in NSECSAV users.

As summarized in Table 45, our results do not contain such a separation of the security findings. On the contrary, the results indicate that the participants' security background has slight impact on their awareness about the security and privacy in the smartphone ecosystem.

Firstly, users ignored endpoint security even in cases where they were uncertain about centralized security (i.e. app testing in the repository). We discovered occasions that even security savvy users were exposed in unauthorized physical access, which is a critical finding due to the device's small size and mobility. Furthermore, we found users in both groups who were exposed to smartphone malware/greyware, which is indexed in the app repository, due to their

Table 45. Findings from correlations between responses (Table notation: ☑ is used when a finding indicating a user *with concern* for security and privacy is present in the group, ☒ is used when a finding indicating a user *without concern* for security and privacy is present in the group)

#	**Finding** (Fd_i)	**SECSAV**	**NSECSAV**
	Adoption of smartphone security software		
1.	Users ignore endpoint security	☒	☒
2.	Minority of users opt for endpoint security	☑	☑
3.	Users who inspect app's reputation tend to not regard smartphone secsoft essential		☒
	Unauthorized access		
4.	Users are exposed to unauthorized remote access	☒	☒
5.	Users are exposed to unauthorized physical access	☒	☒
6.	Increased impact of unauthorized access		☒
7.	Users who scrutinize security messages opt for physical security controls	☑	☑
	Trust in app repository		
8.	Users are exposed to malware indexed in the app repository	☒	☒
9.	Users trust the app repository but are unaware of app testing		☒
10.	Uses who do not trust app repository tend to search in it for free secsoft		☑
	Inspection of security messages		
11.	Minority of users scrutinize prompted messages	☑	☑
12.	Users ignore prompted messages	☒	☒
13.	Users who ignore security messages are uncertain if apps undergo security analysis		☒
	User's background influences their security		
14.	User's security posture is influenced by IT skills	☒	☒
15.	Users are ignorant of smartphone secsoft and smartphone malware	☒	☒
16.	Users disregard security notions in the smartphone app ecosystem		☒
	Other findings		
17.	The device is used for personal and business purposes	☒	☒
18.	Users disregard security indicators during app selectio	☒	☒

† x^2 test of independence value, significance level, association's direction (positive, negative).

trust to the app repository, disregard of security messages and in some cases due to the unawareness of the threat itself. Our results also suggest that current security indicators that are being used by app repositories are ineffective. Therefore they must be redesigned and the users must be trained to use them. However, we found instances where NSECSAV users were concerned about their security and privacy, e.g. by inspecting security messages and using security messages.

Overall, our results provide proof that smartphone users require awareness training specifically tailored for smartphone security. To this end, the current common body of knowledge

for the security domain (e.g. (Theoharidou and Gritzalis, 2007) must be extended to include the necessary background (such as impact of authorization decisions via security messages, unique attacks, significance in the adoption of security controls, etc.) to enable smartphone users cope with the challenges in the smartphone ecosystem.

4.5.3 Limitations

One limitation of our study is that findings may be affected by the sample's demographics. It might be the case that significant differences exist in the security awareness of smartphone users between different demographics (e.g. different country and/or continent). Moreover, as the majority of respondents is Android or iOS users, male, and aged [15-30], the results may be biased towards the population's device, gender, or age. During our analysis we found only a few statistically significant differences in the responses between users of different gender and platform. Also, users aged [15-30] tend to be the early adopters of technology and, hence, we consider that our findings provide considerable insight about smartphone security awareness in general. This is also validated by the common security findings from our research and the existing relevant literature, which are comparable and suggest a clear security unawareness of smartphone users.

Our data collection relies on self-reported statistics, e.g. we asked users to classify their IT expertise and whether they had an information security background (at minimum comprehension of the security notions of threat, risk, safeguards, attacks) - deriving either from academic sources (e.g. undergraduate/postgraduate information security courses) or industrial sources (e.g. certifications on information security). Although we decided to avoid a direct validation of these responses (e.g. with a background quiz or by inspecting the user's device), to keep the length of the instrument short and avoid the user's fatigue, our discussion with the user ensured the: (a) validity of her responses, (b) comprehension of the questions, and (c) comprehension of technical terms.

Furthermore, during the discussion the researchers were cautious not to reveal directly that the final purpose of the survey was to measure the level of security and privacy awareness, as we expected that this would inflate their responses, e.g. as in (Androulidakis and Kandus, 2012). This was in fact the reason why in the beginning of the survey we claimed to do a survey about a smartphone app usage and not about smartphone security. Nonetheless, the findings showed that we successfully avoided such misleading behaviors, since the majority of the answers we collected are alerting in terms of the user's security awareness. Finally, we used in the questionnaire control questions (namely Q5, Q7 and Q17-18, c.f. Appendix), in order to ensure that the researcher did not accidentally overlook any ambiguous responses from outliers during the interview. In the case that the control questions identified outliers

during the sample's analysis, the relevant data were excluded from the analysis (leaving us with 458 cases).

Another limitation of the survey is that it is cross-platform (i.e. focuses on five different popular smartphone platforms), thus it may suffer by the heterogeneity of security software controls that are available in each platform. In this context, a security control may be provided by the platform as a service e.g. (remote wipe in iOS is provided with iCloud), whereas it may require the installation of a third-party application from the app repository in another platform (e.g. in Android). Moreover, a security control may not be supported, due to restrictions of the platform's security model, e.g. antivirus in iOS. In such cases, we excluded the relevant smartphone population from the analysis.

Also, the same security control may be implemented differently in these platforms and, thus, the provided security, as well as the available attacks against it may differ significantly. For instance, Android offers device locking using graphical passwords, which - apart from the traditional attacks against passwords (e.g. guessing, shoulder-surfing, etc.) - may suffer from smudge attacks (Aviv et al., 2010). Furthermore, software exists in some platforms, which protects access - e.g. via encryption, password, etc. - to selected assets, such as corporate documents or multimedia files. This software may be available either as a standalone app in the app repository, or as part of a Mobile Device Management (MDM) solution.

Nonetheless, our survey did not focus on the above-mentioned details regarding security controls. It examined whether users adopt these security controls, without focusing on the implementation details or the software origin (i.e. third-party or offered by the platform), finding that the majority of participants do not use them.

4.6 Summary

Smartphone users increasingly download and install third-party applications from official application repositories. Attackers may use this centralized application delivery architecture as a security and privacy attack vector. This risk increases since application vetting mechanisms are often not in place and the user is delegated to authorize which functionality and protected resources are accessible by third-party applications.

In this chapter, we mount a survey to explore the security awareness of smartphone users who download applications from official application repositories (e.g. Google Play, Apple's App Store, etc.). The survey findings suggest a security complacency, as the majority of users trust the app repository, security controls are not enabled or not added, and users disregard security during application selection and installation. In addition, we further explore the sample to discover if a security background affects the smartphone security awareness of the survey participants, by splitting them in two groups comprising of security savvy and non-

security savvy users. The results of our analysis indicate that the participants' security background has slight impact on their security awareness in the smartphone ecosystem.

Chapter 5: User centric mitigation of smartphone threats

"Security is a chain; it's only as secure as the weakest link" B Schneier

6.1 Introduction

Smartphones' popularity lies mainly with their pervasiveness, which stems from their small size, advanced processing and connectivity capabilities, reduced cost, and their ability to host multi-purpose third party applications. Smartphones host heterogeneous data such as multimedia, sensor data, communication logs, data created or consumed by applications, etc. A smartphone user carries the device on multiple locations throughout the day, and allows connections to various networks that are often not secure. As the same device may be used for both, work and leisure purposes, smartphones often contain a combination of valuable personal and business data.

The complexity of administrator attempts to secure organisation assets rises, as users continue to bring their own smartphones in corporate premises (Nachenberg, 2011). Often, organizations are not prepared to manage smartphone heterogeneity, especially when the required resources or expertise is not present (consumerization). Smartphones extend the business perimeter, while existing security and privacy perimeter-oriented mechanisms are inadequate (Oppliger, 2011). In this context, the importance of smartphone data, in conjunction with their ability to interact with corporate assets, make them economically attractive to attackers (Caldwell, 2011). Hence, traditional risks (e.g. theft, fraud, etc.) may reappear with increased impact. They can pose a security challenge (Dlamini et al., 2009) or take place using new attack vectors, e.g., using smartphone location capabilities for surveillance (c.f. §2).

Traditional risk assessment methods treat smartphones as an asset of a business information system, similarly to a personal computer or a laptop. They treat the smartphone as a single entity, where threat and vulnerability assessment are performed on the asset as a whole. Although a smartphone can be viewed as a kind of small scale information system, making existing methods applicable, such an assessment is not ideal for risks that target device (sub)assets, e.g. GPS sensor (i.e. surveillance), logs (i.e. call logs disclosure), etc. This is due to the fact that they do not take account smartphone-specific threats, neither the unique vulnerabilities that a smartphone security model introduce. Furthermore, most risk assessment methods are not intended for users, but mainly for businesses. Thus, a targeted risk assessment method is useful, so as to assess user-specific parameters and smartphone-specific threats, in a consider-

ably more fine-grained fashion. We contribute towards this direction by proposing a risk assessment method specifically tailored for smartphones.

In addition, smartphones implement different security models, which make traditional countermeasures ineffective. At the same time, as we discussed in the previous chapter, users: (a). do adopt the security controls that are available for their device (either pre-existing (e.g. password lock) or third-party (e.g. secsoft)) and (b). believe that applications deriving from the official repository are secure, which is a serious vulnerability as this complacency tends to lower the users' defenses (c.f. §XX). Contrarily to security unaware users of (a), users who *trust the apps that are downloaded from the repository* cannot be identified with the use of software, such as smartphone agents, Mobile Device Management (MDM) (Redman et al., 2011), etc. For this reason, we propose and evaluate the effectiveness of a prediction model that identifies users who trust applications from the app repository. The prediction model was constructed with the user input from the user case study (i.e. from questionnaire) and its effectiveness was validated with a different user group, which was collected from the UK.

Overall, this chapter includes the following contribution:

- *Trust repository prediction model*. We construct a prediction model that identifies the users who believe that apps that are installed form the official app repository are risk free. . The model is assessed, evaluated, and proved to be statistically significant and efficient
- *Risk assessment for smartphones*. We present an abstract risk assessment method for smartphones, which does not treat the device as a single entity. On the contrary, it identifies smartphone assets and provides a detailed list of specific applicable threats. For threats that use application permissions as the attack vector, risk triplets are facilitated. The triplets associate assets to threats and permission combinations. Then, risk is assessed as a combination of asset impact and threat likelihood. The method utilizes user input, with respect to impact valuation, coupled with statistics for threat likelihood calculation.
- *Privacy impact assessment*. We present a more concrete method for accessing privacy risk of smartphone users, which a builds on the abstract method for smartphone risk assessment and is tailored to the 'nuts and bolts' of Android.

The chapter is organized as follows. Section 2 describes, assesses and evaluates the trust repository prediction model. In Section 3 the proposed abstract risk assessment method is introduced. The method is applied in Section 4, in privacy impact assessment in Google Play. Finally, the chapter ends with a discussion and a summary in Section 5 and Section 6 respectively.

6.2 Trust Repository Prediction Model

The survey findings revealed strong indications regarding the lack of security awareness in smartphone users (c.f. §XXX). The users' belief that applications deriving from the official repository are secure is a serious vulnerability. This holds true, since such trust in the application repository is more likely to: (a) be encountered in users who are not technology and security savvy, and (b) increase the users' security complacency, as they are more likely not to scrutinize security messages and not use secsoft software. Moreover, as discussed in §XX, this trust is not always justified by the availability of security controls in the app repository (e.g. app testing) or their effectiveness regarding the mitigation of malware. As a result, this security complacency of users regarding trust in the app repository may be exploited by attackers using the repository as a malware infection vector.

The users with this security complacency cannot be identified via software solutions (e.g. software agents, MDM, etc.). Therefore, we provide a prediction model that can be used by an organization to identify them and subsequently engage them in a relevant security training so as to raise their awareness (Thomson and von Solms, 1998). The rest of this section describes the construction of the prediction model, as well as its statistical assessment and experimental evaluation.

6.2.1 Model's Estimation

To construct the prediction model we computed the logistic regression model for the categorical variable *TrustRep*, denoting user trust in the app repository. The logistic regression equation for *TrustRep* is (Hosmer and Lemeshow, 2000):

$$\text{logit}\,(p) = \log\,(p/1\text{-}p) = b_0 + b_1 * x_1 + \ldots + b_n * x_n \quad (1)$$

where p= Probability (*TrustRep*=1) and b_i are the factors of a set of independent variables (x_i), which predict the *TrustRep*, given in log-odds units. *TrustRep* is the dependent binary categorical variable, denoting user trust in the app repository (*TrustRep*=1) or absence of trust in the app repository (*TrustRep*=0) respectively.

Let:

$$z = b_0 + b_1 * x_1 + \ldots + b_n * x_n \quad (2)$$

, then the probability that a user trusts the app repository can be given from equation 2.

$$p = \exp(z) / (1 + \exp(z)) \quad (3)$$

The prediction model seeks to maximize the probability of equation 3. To this end, we applied stepwise Forward Wald (FW) (Harrell et al., 1984), so as to discover the coefficients (independent variables (x_i)) of the prediction model. FW starts with an empty model, i.e. a model with no prediction coefficients, and iteratively adds coefficients that are statistically significant and maximize statistic scores. Iteration stops when no coefficient can be added.

Table 46 presents the seven (7) independent coefficients x_i, which were selected after applying FW along with: (a) their condition (i.e. when x_i =1), (b) their factors (b_i), and (c) their significance level p (i.e. whether the predictor gives a significant contribution in prediction of the dependent variable). For instance, the coefficient x_1 has b_1= 1.351, significance level p= 0.007, and, x_1=1 when the user's IT skills are "Moderate". FW did not give a statistical significant solution with a constant and as a result b_0 = 0. In this respect, equation 4 describes the prediction model of the users that trust the application repository, i.e.:

$$z = 1.351*x_1 +1.092*x_2 -1.688 *x_3 +1.523*x_4+1.314*x_5 -0.475*x_6-0.741*x_7 \quad (4)$$

The model assigns to the binary variable *TrustRep* the value *True* - i.e. predicts that the given user trusts the app repository - when the probability $p=\exp(z)/(1+\exp(z))\geq 0.5$.

Table 46. *TrustRep* Prediction model coefficients

x_i	x_i condition	b_i	p
x_1	IT = "Moderate"	1.351	0.007
x_2	IT = "Good"	1.092	0< 0.001
x_3	BlackBerry = False	-1.688	0< 0.001
x_4	NoUse = False	1.523	0< 0.001
x_5	P&S = False	1.314	0.003
x_6	Pirated = False	-0.475	0.047
x_7	Personal = False	-0.741	0.005

6.2.2 Model's Statistical Assessment

We statistically assessed the prediction model, so as to ensure its significance by performing the following tests regarding the model's: (a) statistical significance, (b) goodness-of-fit (i.e. whether the model adequately describes the data), and (c) the minimum sample size requirement for logistic regression.

Initially, we tested the null hypothesis that the dependent variable, denoting trust in the app repository, and the seven independent variables x_i are unrelated, i.e. the selected predictors variables x_i do not contribute to the prediction of the dependent variable *TrustRep*. We denote with p the probability that the null hypothesis is true and reject it if $p<0.05$. Table 47 describes the results of these tests. We reject the null hypothesis ($p<0.05$), hence our model is

statistically significant. The model has seven degrees of freedom, as each of its coefficients has one degree of freedom.

Then, the model was tested against the Hosmer-Lemeshow goodness-of-fit test (Hosmer and Lemeshow, 2000), so as to determine whether the model adequately 'fits', i.e. describes, the data. The significance level *p* in this test must be greater than 0.05, otherwise the model does not adequately fit the data. Also, we checked the ability of the model's independent variables to explain the variance of the dependent variable (Nagelkerke, 1991). For this test, the Nagelkerke R^2 value must be in the range [0,1]. As shown in Table 48, the model was successfully tested against these two tests. Finally, Harrell et al. (1984) argue that for a valid logistic regression model at minimum the sample size must have 10 cases for each candidate predictor (variable) of the instrument. Our sample size meets this requirement, as there exist 36 candidate predictors (n=458, i.e. >10*36=360).

Table 47: Model's statistical significance

x^2	df	p
193.272	7	< 0.001

Table 48: Results of Hosmer-Lemeshow and Nagelkerke tests

Statistical Test	Value
Hosmer and Lemeshow	0.828
Nagelkerke R^2	0.459

6.2.3 Model's effectiveness

The experimental results regarding the effectiveness of our prediction model, in terms of the correct percentage of its predictions in the Greek sample, are depicted in Table 49. The model is prone to Type I errors (false positives), performing 25.5% correct predictions. On the contrary, Type II errors (false negative) are negligible, as the model correctly predicted 96% of the users who trust the app repository. We argue that a significant Type I error coupled with a negligible Type II error is a good security tradeoff, as far as smartphone awareness is concerned. This is so, as the Type I error would incur nuisance to users who do not trust the app repository. For instance, in an organization these users may be asked to undergo relevant security training in order to raise their awareness. On the other hand, Type II errors may lead to security incidents.

Table 49. Evaluation of the prediction model in the Greek sample (n=458)

Observed \ Predicted	No	Yes	Percentage Correct
No	28	82	25.5%
Yes	14	334	96.0%
Overall percentage			79.0%

The prediction model was also tested against a sample different from the one that was used to estimate the prediction model. This would give an indication on whether our model, which was built via inferential statistics, suffered by the demographics of the Greek sample. In this context, the prediction model was tested against a sample (n=102, min_{age}=19, max_{age}=59) comprising of smartphone users living in the United Kingdom ('UK sample'). The data collection methodology of this sample was the same as that of the Greek sample. Concerning the demographics of the UK sample, ~30% were security savvy users, 72.5% were male, 80.4% of the population was aged [19-30] and 68.6% stored personal data on their device. Regarding their technical skills ~28.% of the respondents were technically savvy and ~16% were non-technically savvy. Most of the respondents (78.4%) trusted the app repository; ~41% preferred pirated apps and only ~6% reported security or privacy issues as an app selection criterion. Finally, only 45% and 21.6% of the users scrutinized security and agreement messages, respectively. Examining the adoption of security controls revealed that: (a) 18.6% used smartphone secsoft, (b) ~6% did not use any software secsoft in any device, (c) ~61% password protected and 18.6% encrypted their device, and (d) 10.8% used remote wipe and 14.7% remote locator.

Table 50 depicts the performance (i.e. percentage of correct predictions) of our model in the UK sample. It turns out that the accuracy of the model with the UK sample is similar to the accuracy with the Greek one. Again, the model is more prone to Type I error, having a slight increase in the correct percentage of true negatives. Regarding true positives, the model had a 5% decrease in the percentage of correct predictions and 0.6% of the overall correct predictions percentage. This indicates that the statistically inferred prediction model is correct and not dependent on the demographics of the Greek sample. Due to the small size of the UK sample, further deductions or comparisons would be unsafe.

Table 50. Evaluation of the prediction model in the UK sample (n=102)

Observed \ Predicted	No	Yes	Percentage correct
No	7	15	31.8%
Yes	7	73	91.3%
Overall percentage			78.4%

6.3 Abstract risk assessment for smartphones

Traditional risk assessment methods treat smartphones as an asset of a business information system, similarly to a personal computer or a laptop. They treat the smartphone as a single entity, where threat and vulnerability assessment are performed on the asset as a whole. Although a smartphone can be viewed as a kind of small scale information system, making existing methods applicable, such an assessment is not ideal for risks that target device (sub)assets, e.g. GPS sensor (i.e. surveillance), logs (i.e. call logs disclosure), etc. This is due to the fact that they do not take account smartphone-specific threats, neither the unique vulnerabilities that a smartphone security model introduce. Furthermore, most risk assessment methods are not intended for users, but mainly for businesses. Thus, a targeted risk assessment method is useful, so as to assess user-specific parameters and smartphone-specific threats, in a considerably more fine-grained fashion. To this end, in this section we propose a risk assessment method specifically tailored for smartphones. This subsection includes a discussion about smartphone assets and a generic impact valuation, a taxonomy of smartphone threats and an abstract risk assessment method.

6.3.1 Smartphone assets

A smartphone is viewed herein as a small-scale information system, which incorporates various assets. Jeon et al. (2011) identify as its assets: (a) *private information* (address book, calling history, location information, etc.), (b) *device* (resources, i.e. CPU, RAM, battery), and (c) *applications*. Another report identifies six assets: (a) *Personal data*, (b) *Corporate intellectual property*, (c) *Classified (governmental) information*, (d) *Financial assets*, (e) *Device and service availability and functionality*, and (f) *Personal and political reputation* (Hogben and Deker, 2010). Another taxonomy includes *Communication* (Voice communication, Messaging), *Data access* (email, Web access, Bluetooth/IR), *Applications* (Maps & Navigation, Social networking, etc.), and *Device/Stored data* (Physical device, Offline applications/Utilities, etc.) (Lederm and Clark, 2011). An assessment of security in the case of the Android platform (Shabtai et al., 2010) analyses: (a) *private/confidential content* stored on the device, (b) *applications and services*, (c) *resources* (battery, communication, RAM, CPU), and (d) *hardware* (device, memory cards, battery, camera).

Our analysis makes use of four asset types: a) *Device*, b) *Connectivity*, c) *Data*, and d) *Applications*. The *Device* asset type includes the physical device and its resources (e.g. battery, RAM, CPU etc.), but *not* the *Data*. The latter are complex and have been previously analysed (see §) according to their data source and their information type. More specifically:

Applications are viewed in this risk assessment method only as user services. A more concrete assessment of smartphone applications must take into consideration the 'nuts and bolts' of the platform's security model (see §X.4).

Smartphones use four *Connectivity* channels, namely: a). *GSM services*, i.e. messaging (SMS, EMS, etc.) and voice calls, b). *PAN interface* (e.g. Bluetooth, IrDA, etc.), a free and ad-hoc short-range data channel, c) *WLANs* (e.g. Wi-Fi, WiMAX, etc.), a fast data channel, and d) *Cellular network*, which provides Internet connectivity at variable speeds, depending on the carrier technology (e.g. GPRS, HSDPA, etc.).

To assess smartphone risk, one should first assess the impact of its assets. Then, assets should be related to smartphone threat scenarios. Impact assessment for each asset is described in the sequel.

6.3.2 Asset impact

A key concept of the proposed approach is, first, to involve the user with the initial impact valuation process. Then, the *risk analyst* should perform transparent associations and aggregations to calculate the overall risk.

Device. In typical risk assessment methods, physical assets are valued in terms of replacement or reconstruction costs, in a quantitative way. For a smartphone this refers to replacement or repair device cost, in the case of loss, theft, or damage. However, a smartphone contains, also, various information types, which need to be co-assessed in terms of impact.

Data. For information assets, a loss of confidentiality, integrity, or availability may be valued via several criteria (ISO, 2008; Lederm and Clarke, 2011). For instance, personal information disclosure, legislation violation, contractual breach, commercial and economic interests, financial loss, public order, international relations, business policy and operations, loss of goodwill/reputation, personal safety, annoyance, etc. Due to the smartphone's multipurpose nature, these impact types vary from purely personal ones, e.g. user annoyance, to typical information systems ones, e.g. commercial interests. This heterogeneity affects risk assessment. Adequate input from user is, thus, required, as clearly opposed to typical smartphone risk assessment efforts (Hogben and Deker, 2010; Shabtai et al, 2010), which use expert opinion.

In a '*personalized'* (or '*itemized'*) risk assessment, the user is asked questions aiming to determine the existing data types. Some indicative examples of these questions are the following:

- Do you *store personal* data in your smartphone?
- Do you use your smartphone for *business purposes*? If so, do you work for a *governmental institution*?

- Do you use your smartphone for *financial transactions*?

In turn, for each identified data type, the user is invited to assess the impact of the following scenarios:

- Which are the worst consequences if your <data type> are *unavailable*?
- Which are the worst consequences if your <data type> are *disclosed to the public*?
- Which are the worst consequences if your <data type> are *modified or damaged* (*deliberately/accidentally*)?

The answers lead to impact calculation for each data type, namely the unavailability impact *ImpactUA* (data_type), the disclosure impact *ImpactDS* (data_type), and the modification impact *ImpactMD* (data_type).

Our approach adopts the "worst-case scenario" principle, i.e. the *max* operator is used to calculate the total scenario impact. The answers must follow a qualitative assessment of the impact types mentioned above, evaluated by the user with a 5-item[29] Likert scale (*very low*, *low*, *medium*, *high*, *very high*). For each impact criterion, a table needs to be produced, mapping each qualitative assessment to a comprehensive description. For example, for the 'personal information disclosure' criterion, the "*very low*" valuation may refer to "*minor distress to an individual*", as opposed to the "*very high*" one, which may refer to "*significant distress to a group of individuals or legal and regulatory breach*". Again, a map to quantitative values is required, because impact criteria cannot be considered equivalent. For instance, the "*very high*" valuation on 'personal information disclosure' must not be quantitatively valued equally to the "*very high*" valuation on 'personal safety'.

Connectivity. Likewise, the user assesses the network services impact. For instance, some indicative questions for network service impacts are listed:

- Which are the consequences if you cannot use the SMS service?
- Which are the consequences if you cannot connect to a Wi-Fi network?
- Which are the consequences if your Wi-Fi connection is been monitored?

The assessment should follow the same valuation tables and scales, as the data valuation ones do. The resulting valuations, i.e. *ImpactUA*(channel), *ImpactDS*(channel), *ImpactMD*(channel), are used for risk assessment of threat scenarios which affect connectivity.

Applications. The same procedure can be used for user applications. Although this approach allows for a fine-grained impact valuation, it adds considerable complexity, as the applications may be numerous. It, also, assumes that a user has a clear perception of an appli-

[29] Any number of levels between 3 and 10 can be used [10].

cation's significance, e.g. by using the application for some time. A trade-off could be the valuation of applications that the user identifies as more important. The assessment per application should follow the same valuation tables.

Total impact valuation. Based on the above, the user has assessed the impact of various scenarios (loss of availability, confidentiality and integrity) for all four smartphone asset types. These values are combined and used to assess the risk of the threats scenarios. For instance, the data type impact values are inferred to their associated data sources (see Table 1). This means that if a user has identified 'personal' and 'financial' data types on her smartphone, then the disclosure impact for the data source *'Messaging'* can be calculated, as follows:

$$Impact_{DS}(Messaging) = max\{Impact_{DS}(personal\ data), Impact_{DS}(financial\ data)\}$$

Likewise, the overall smartphone impact is the *max* impact from all four assets, i.e., the device, data, applications, and connectivity.

Table 51. Smartphone threats

Dimension	*Threat*	*C*	*I*	*A*
Network Connectivity	T1 Spoofing	✓	✓	✓
	T2 Scanning	✓		
	T3 Denial of Service, Network congestion			✓
	T4 Spam, Advertisements			✓
	T5 Eavesdropping	✓		
	T6 Jamming			✓
Device	T7 Loss, theft, disposal or damage	✓	✓	✓
	T8 Cloning SIM card	✓	✓	
	T9 Technical failure of device		✓	✓
	T10 Unauthorized device (physical) access	✓	✓	✓
Operating System	T11 Unauthorized Access	✓	✓	✓
	T12 Offline tampering	✓	✓	✓
	T13 Crashing			✓
Applications	T14 Misuse of Phone Identifiers	✓		
	T15 Electronic tracking/surveillance/exposure of physical location	✓		
	T16 Resource abuse			✓
	T17 Sensitive Information Disclosure (SID), Spyware	✓		
	T18 Corrupting or modifying private content		✓	✓
	T19 Disabling applications or the device			✓
	T20 Client Side Injection/ Malware	✓	✓	✓
	T21 Direct billing		✓	
	T22 Phishing	✓	✓	

6.3.3 Threat

Threat likelihood is assessed on the basis of: (a). experience and applicable statistics, (b). vulnerabilities, and (c). existing controls and how effectively they may reduce vulnerabilities (ISO, 2008). In this section a list of smartphone threats is presented, together with a discussion on how threat likelihood may be assessed.

Table 51 presents a threat list expanding similar lists that are available in the smartphone literature (Becher et al., 2011; Enck et al., 2011; Hogben and Deker, 2010; Jansen and Ayers, 2007; Jeon et al., 2011; Lederm and Clarke, 2011; OWASP, 2013; Shabtai et al, 2010). Each threat is grouped in the appropriate attack vector dimension (i.e. an asset utilization may be misused to impair another one (e.g. application access rights may be misused to leak private data)). Each threat is associated with the security attribute that it impairs.

A particular application may be an asset that needs protection and, at the same time, an attack access vector to other assets. For instance, the availability of a social networking application may be considered as a significant asset by a user (*high impact*), while being the attack vector for privacy threats. This happens since in Android's security model, which is permission-based, access to protected resources happens only if the requesting application is granted the corresponding permission for the resource (Google, 2013a). Even if the requesting application is benign (i.e. not leaking any private data for malicious purposes), its privilege to access private data may be misused by another malicious application performing a deputy attack (Chin et al., 2011; Felt et al., 2011a; Grace et al., 2011), or by a malicious advertisement library that is bundled with it (Grace et al., 2012; Pearce2012, Stevens et al., 2012). Moreover, in Android's security model the majority of permission granting happens via the user. The users are expected to scrutinize permission requests and decide upon installation whether an app accesses a protected resource or not (Google, 2013a). Orthogonally to this expectation, the discussion in §XX has proven that average users tend to ignore permissions, or to not understand them at all. On the other hand, even users that tend to inspect app permission are hampered by deficiencies of Android's permission system (Google, 2013a). In specific, Android users must either accept permissions in an all-or-nothing approach, i.e. they cannot authorize only a subset of the requested permissions, or cancel the installation of the requesting app. Also, Android permissions are not fine-grained - e.g. the permission `SEND_SMS` allows an app to send SMS messages both to normal and premium numbers - making authorization decisions more difficult.

The permission acceptance likelihood differs in smartphone platforms. It depends on authorization decisions, as delegated by the platform security model. These decisions differ significantly from allowing users to make security-critical authorization decisions (e.g. Android's community driven security model), up to placing functionality control barriers in ap-

plications that enter application repository (e.g. the 'walled garden' approach of Apple's App Store) (c.f. §2).

6.3.4 Risk

A triplet is used for the abstract risk assessment of threats, which are in turn associated with specific permission access rights (i.e. threats T14-T19, T21, T22) is:

(asset, permission combination, threat).

Asset refers to the asset targeted by the threat. *Permission combination* refers to the permissions for the dangerous functionality required by the *threat*. The *permission combination* is the vulnerability the threat exploits. In turn, *threat likelihood* is valuated on the basis of: (a) the likelihood of permission combination acceptance in the smartphone platform, (b) the threat incident likelihood, i.e. statistics on threat incidents in the platform or previous incidents experienced by the user, and (c) the relevant security control existence (e.g. Use of Mobile Device Management (Redman et al., 2011)). Given that the user has assessed the asset impact, the impact may be combined with the likelihood of the threat and the permissions acceptance, so as to calculate risk by forming the triplet:

(asset impact, permission likelihood, threat likelihood) => Threat Risk

Risk assessment is calculated on the basis of a risk matrix (ISO, 2008) (see Table 52). Risk can be mapped as *Low* (0-2), *Medium* (3-5), or *High* (6-8). Since each threat is associated with specific security attributes, the relevant asset impacts are taken into account. For instance, when assessing the *Risk* of threat *T17* 'Sensitive Information Disclosure' on the data source 'Messaging', the disclosure impact value $Impact_{DS}$ *(Messaging)* is used as the *asset impact*, because the particular threat affects confidentiality.

Table 52. Risk matrix

Threat likelihood		**Low**			**Medium**			**High**		
Permission likelihood		**L**	**M**	**H**	**L**	**M**	**H**	**L**	**M**	**H**
	0	0	1	2	1	2	3	2	3	4
	1	1	2	3	2	3	4	3	4	5
Asset impact	**2**	2	3	4	3	4	5	4	5	6
	3	3	4	5	4	5	6	5	6	7
	4	4	5	6	5	6	7	6	7	8

For threats that cannot be associated with specific permissions, i.e. T1-T13, T20, such a triplet cannot be produced. In this case, threats are combined with specific assets, and their risk is calculated on the basis of asset impact and threat likelihood, where the likelihood is calculated based on threat incident statistics or previous incidents experienced by the user. This is done through a simple table (see Table 53).

Table 53. Simplified risk matrix

Threat likelihood		**Low**	**Medium**	**High**
Asset impact	**0**	L	L	M
	1	L	M	M
	2	L	M	H
	3	M	M	H
	4	M	H	H

6.3.5 Case Study: Risk assessment in Android

This section provides a demonstration of the proposed abstract risk assessment method in the case of the Android platform. Android was selected because it is: (a). is the most popular smartphone platform holding the majority of the smartphone market sales share ((79%) in Q2 of 2013 (Gupta et al., 2013), (b). open source and, hence, its security model details are publicly available, and (c). well studied platform and statistics about its threats are available.

For the purpose of this case study a HTC Hero (Android version 2.1) owner is assumed, who holds a 'high' managerial position in the Pharmaceutical industry. The user identified two data types, i.e., personal data and business data. She provided data impact valuations as follows: $Impact_{UA}$(personal)=1, $Impact_{DS}$(personal)=2, $Impact_{MD}$ (personal)=2, $Impact_{UA}$(business)=2, $Impact_{DS}$(business)=4 and $Impact_{MD}$ (business)= 3. She chose not to assess network services and applications. She also assessed the replacement cost of the device as *low*, i.e. Impact(device)=1. The aforementioned impact valuations are summarized in the following table.

Table 54: Impact valuation of personal and business data in the case study scenario

Information type	***Impact valuation***		
	Unauthorized Access ($Impact_{UA}$)	*Disclosure* ($Impact_{DS}$)	*Modification* ($Impact_{MD}$)
personal	1	2	2
business	2	4	3

In addition, the user provided the following data:

- The only smartphone security control enabled in her device is automatic password lock.
- She regularly discusses critical business issues over the carrier voice service.

- She does not consider herself a technology or security savvy user.
- No past security incident has ever come into her attention.
- She has noticed some delays in the device.
- She travels frequently in technology 'underdeveloped' country where fast 3G data connections are not available and, thus, her only way to connect to the internet is either through public free Wi-Fi hotspots, or expensive carrier network if one is not available.
- She has never updated the firmware.
- She regularly installs applications while she is using public transport (trains, subway, etc).

The demonstration of the abstract risk assessment method will focus only on the threats T5, T10, T11, T13, T20, and T21.

- *T5 Eavesdropping*. The user-identified the use of GSM voice services, in a carrier where UMTS is not supported. As a result, the possibility of abusing the discussion confidentiality is High (Becher et al., 2011). The $Impact_{DS}$(GSM Service)=*max*{$Impact_{DS}$ (personal), $Impact_{DS}$(business)}=4. Risk is assessed as High (see Table 53).
- *T10 Unauthorized device (physical) access*. The user has the automatic password device lock enabled, therefore this threat likelihood is Low[30]. As physical access to a device affects all security attributes and may cause significant damage to the hardware itself, the total impact of the asset 'device' is the max value of the replacement cost and the relevant impact valuations for the data it holds. Therefore, Total_Impact(device) = max{Impact(device), $Impact_{DS}$(personal), $Impact_{DS}$(business), $Impact_{MD}$(personal), $Impact_{MD}$(business), $Impact_{UA}$(personal), $Impact_{UA}$(business)} = 4. Therefore, the threat risk is Medium (see Table 53).
- *T11 Unauthorized Access.* The user is running an Android version that suffers by known security vulnerabilities. The vulnerabilities have been identified and patched by the device vendor without the user applying them. In addition, publicly available and stable (i.e. confirmed) shell code exists, which can exploit the vulnerabilities[31]. The shell code can be used by an attacker to gain unauthorised access to the devise with administrator privileges. Thus, the threat likelihood is High. Since T11 may affect all security attributes but not the hardware itself, the $Impact_{DS,MD,UA}$(Device) is the max impact valuation of the data it holds, i.e. it equals with 4. As a result, the threat risk is High.
- *T13 Crashing*. The user is running a buggy version of Android 2.1 that affects the device performance. An official fix (patch) for this vulnerability is available from the device vendor, thus, the threat probability is High. T13 affects the device's availability. The unavaila-

[30] For the sake of the discussion's simplicity, we consider that an attacker cannot exploit a vulnerability in the password lock mechanism to gain unauthorized physical access to the device.

[31] http://www.exploit-db.com/exploits/15548/

bility impact of the asset 'device' is the max value of the relevant impact valuations for the data it holds. Therefore, $Impact_{UA}$(device) = max {$Impact_{UA}$(personal), $Impact_{UA}$(business)}=2 and the threat risk is High.

- *T20 Client Side Injection/ Malware*. The likelihood of this user downloading malware in her device is considered High, since: (a) the user frequently installs applications in the device, (b) user is not technically and security savvy, and (c) Android is an appealing target for smartphone malware (Zhou and Jiang, 2012). As in T11, this threat may affect all security attributes but not the hardware itself. The $Impact_{DS,MD,UA}$(Device) is the max impact valuation of the data it holds, i.e. it equals with 4. Thus, the threat risk is High.
- *T21 Direct billing*. Since the user frequently makes use of the carrier data network in order to connect to the Internet, malicious applications may abuse the Internet permission to incur direct costs to the user. The malicious application needs - apart from the permission to open network socket - access to the networking state, i.e. if the carrier data are being used. Hence, the involved triplet is: <USIMCard, Use Network Socket + Access Information about Networks, T21CarrierData>. The likelihood of the permission combination is High (≤86%)[32] (Felt et al., 2011a), and the threat likelihood is also High. Thus, the threat triplet is: <$Impact_{MD}$(USIMCard), High, High>. As the $Impact_{MD}$(USIM Data)=3 (i.e. max modification impact of associated data types - personal and business), threat risk is calculated from the triplet <3, High, High> and thus it is 7 (High) (see Table 52).

Table 55 summarizes the risk assessment of the threats included in the case study.

Table 55. Summary of the risk assessment results

Threat	Asset Impact	Permission Likelihood	Threat Likelihood	Risk
T5	ImpactDS(GSM Service) = 4	*N\A*	*High*	*High*
T10	Total_Impact(Device) = 4	*N\A*	*Low*	*Medium*
T11	$Impact_{DS,MD,UA}$(Device) = 4	*N\A*	*High*	*High*
T13	$Impact_{UA}$(Device) = 2	*N\A*	*High*	*High*
T20	$Impact_{DS,MD,UA}$(Device) = 4	*N\A*	*High*	*High*
T21	$Impact_{MD}$(USIM Data) = 3	*High*	*High*	*High*

6.4 Privacy Impact Assessment in Android

This subsection proposes privacy impact assessment on Android. Android was selected for the reasons that were presented in §XX, namely: (a). *popularity*, being the smartphone platform with the greatest user space, (b). *portability*, i.e. compatibility with different smartphone

[32] Access to network state is granted to all Android applications without user intervention.

hardware, (c). *open source*, which allows researchers to *study its* security model and *extend* it, (d). *flexibility*, i.e. allows the installation of apps which are available outside of the Google Play.

As discussed in §XXX, Privacy Impact Assessment (PIA) refers to "*a systematic process for identifying and addressing privacy issues in an information system that considers the future consequences for privacy of a current or proposed action*"' (Warren et al., 2008). This process is a risk assessment process, focused on privacy, and it is mainly associated with the collection, use or disclosure of personal information. While differences occur between approaches, the underlying principles remain the same (see "fair information principles" (Government of Canada). Herein, we start our discussion by presenting a comprehensive list of these principles, defined with respect to the Android marketplace, namely:

1. Applications must clearly state why personal information is being collected, at or before the time of collection
2. User data collected by apps should remain accurate, relevant, up-to-date, and not excessive in relation to the collection purpose
3. Apps should not retain user data for periods longer than the one needed for the purpose of the collection
4. User data must be collected by apps with the knowledge and consent of the individual
5. User data must not be communicated to third parties, except under specified conditions that might include consent
6. Application must ensure user data are kept under secure conditions
7. User data must be accessible to the individual for amendment or challenge

These principles indicate that app users should be able *to review an app's privacy policy* prior to downloading it and the *policy must clarify what types of data are collected* and for *which purpose*. Nonetheless, in *Google Play* the inclusion of such a policy is *not mandatory* for app developers (Google, 2013c). While Android's security model requires developers to ask the user to grant permissions to a specific app, user studies have proven that Android users ignore permissions, or to not understand them at all (see the relevant discussion in §X). If we consider that the permissions are not refined, there are hidden privacy risks for specific permissions, which we identify in the sequel (§XX). The main thing missing is that developers do not state why they need the data (for some permissions, the collection purpose is not obvious), how they will use these data, how they will be stored or protected and whether they will be transferred to a third party.

In this context, this subsection builds on the abstract method for smartphone risk assessment, which was presented previously, to present a more concrete method for accessing pri-

vacy risk of smartphone users, which is tailored to the 'nuts and bolts' of Android. Previous work on app risk assessment focuses either on fraud (Wang et al., 2013), i.e. the ability to change settings, apply exploits or send SMS. Moreover, other works assess the privacy risk in Android apps, by relying on static or dynamic analysis of the app to find out whether a data disclosure can take place or not (references). These efforts do not take into consideration the user's input about threat impact, assuming that the value of each asset is perceived similarly across users. For instance, access to the user's location is considered as a privacy violation, but for a specific individual this may not be so, since the individual may assess access to her location as negligible. Therefore, a per user privacy risk assessment is not achievable.

This subsection continues with a taxonomy of user data found on a smartphone and their respective Android permissions and a discussion of ways to assess the impact of their disclosure for a particular user. For each threat, we map the permissions required for the threat to occur and assess privacy risk on a per user basis by combining the likelihood of permissions with user input regarding the impact of disclosure. We present the applicability of the method with a case study.

6.4.1 Personal data types

We identified the following data types in Android that may be the target of a privacy violation/breach from a third-party app by analyzing the latest Android manifest,[33] namely:

- *Communication data*, which include SMS, MMS, Voice, Wap_Push.
- *Sensor data*, which include the Camera or Microphone. These could also be indicators of the user's location.
- *Location data*, i.e. GPS data (fine location) as well as location derived indirectly from the networks that the user connects (coarse location) or his social media feeds.
- *External storage data*, which include application data as well as personal documents, photos or emails of the user.
- *Contact data*, i.e. the smartphone's contact list or contacts derived by the social media that the user participates in.
- *History/Usage data*, which indicate the user's preferences and can be collected by bookmarks, subscriptions to feeds, the call or task logs, social media feeds, the dictionary.
- *Calendar data*, which could also be an indicator of contacts and/or the user's location
- *Identity data*, which refers to all the unique identifiers that can be used to identify a user, e.g. his device ID, e-mail addresses, his profile ID.
- *Credentials*, the user's authentication tokens, passwords and PINs

[33] At the time of our analysis it is Jelly bean (v. 4.2.2)

Each identified data type has been mapped to Android permissions that enable their use or collection by an application (see details in §XX). These permissions are listed in Generic vs. Personalized PIA

Regardless of whether we refer to a single user or a category of users, a PIA for smartphones would typical refer to (a). a *device* (with Android OS in our case) and (b). one or more *applications*. For each device, the personal information accessed or collected needs to be defined, coupled with respective access permissions of applications or controls. For each application, we need to identify the collection and use conditions of PII, privacy risks deriving from either `dangerous' (Google, 2013a) or unnecessary permission combinations or by the lack of appropriate countermeasures.

Table 56 summarizes the access level of permissions that protect either assets in each data type (e.g. SMS, exact location), or a communication channel (e.g. Internet, SMS). The majority of permissions has a self-explanatory label, which specifies the asset or channel that the permission protects (e.g. 'READ_SMS'). The rest of them are briefly described here: i. ACCESS_NETWORK_STATE and ACCESS_WIFI_STATE permissions grant access to information on the carrier and Wi-Fi network respectively, ii. AUTHENTICATE_ACCOUNTS allows an app to perform account authentication, iii. GET_TASKS provides information about apps that are or have been executed in the device, iv. PROCESS_OUTGOING_CALLS allows an application to monitor and manage outgoing calls, v. READ_PHONE_STATE grants access to identifiers such as the IMSI, IMEI, vi. USE_CREDENTIALS allows an app to request an authentication token for an account that has been stored in the device, and vii. WRITE_EXTERNAL_STORAGE provides to an application both read and write access to the external storage.

This table includes the access level, collected by parsing the platforms source code (Google, 2013d). The analysis focuses on the currently most popular Android versions, namely Jelly Bean (API 17, API 16), Ice Cream Sandwich (API 15) and Gingerbread (API 10), with a distribution among devices 5.6% 32.3% 23.3% 34.1% respectively (Google, 2013b). The majority of permissions has a dangerous access level (Google, 2013a), i.e. the user must decide before application installation whether to accept or not the permission request. Only five permissions are protected with a normal access level, i.e. they are automatically granted to any application that requests them, namely ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE, READ_EXTERNAL_STORAGE, GET_ACCOUNTS, and SUBSCRIBED_FEEDS_READ. Finally, in the latest version of Android, READ_LOGS and MOUNT_UNMOUNT_FILESYSTEMS are not available to third-party applications (*signatureorsystem* access level).

Table 56 in alphabetical order.

6.4.2 Generic vs. Personalized PIA

Regardless of whether we refer to a single user or a category of users, a PIA for smartphones would typical refer to (a). a *device* (with Android OS in our case) and (b). one or more *applications*. For each device, the personal information accessed or collected needs to be defined, coupled with respective access permissions of applications or controls. For each application, we need to identify the collection and use conditions of PII, privacy risks deriving from either `dangerous' (Google, 2013a) or unnecessary permission combinations or by the lack of appropriate countermeasures.

Table 56 summarizes the access level of permissions that protect either assets in each data type (e.g. SMS, exact location), or a communication channel (e.g. Internet, SMS). The majority of permissions has a self-explanatory label, which specifies the asset or channel that the permission protects (e.g. 'READ_SMS'). The rest of them are briefly described here: i. ACCESS_NETWORK_STATE and ACCESS_WIFI_STATE permissions grant access to information on the carrier and Wi-Fi network respectively, ii. AUTHENTICATE_ACCOUNTS allows an app to perform account authentication, iii. GET_TASKS provides information about apps that are or have been executed in the device, iv. PROCESS_OUTGOING_CALLS allows an application to monitor and manage outgoing calls, v. READ_PHONE_STATE grants access to identifiers such as the IMSI, IMEI, vi. USE_CREDENTIALS allows an app to request an authentication token for an account that has been stored in the device, and vii. WRITE_EXTERNAL_STORAGE provides to an application both read and write access to the external storage.

This table includes the access level, collected by parsing the platforms source code (Google, 2013d). The analysis focuses on the currently most popular Android versions, namely Jelly Bean (API 17, API 16), Ice Cream Sandwich (API 15) and Gingerbread (API 10), with a distribution among devices 5.6% 32.3% 23.3% 34.1% respectively (Google, 2013b). The majority of permissions has a dangerous access level (Google, 2013a), i.e. the user must decide before application installation whether to accept or not the permission request. Only five permissions are protected with a normal access level, i.e. they are automatically granted to any application that requests them, namely ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE, READ_EXTERNAL_STORAGE, GET_ACCOUNTS, and SUBSCRIBED_FEEDS_READ. Finally, in the latest version of Android, READ_LOGS and MOUNT_UNMOUNT_FILESYSTEMS are not available to third-party applications (*signatureorsystem* access level).

Table 56: Access Level of Permissions

Permission	Access Levels per API			
	17	16	15	10
ACCESS_COARSE_LOCATION	d	d	d	d
ACCESS_FINE_LOCATION	d	d	d	d
ACCESS_NETWORK_STATE	n	n	n	n
ACCESS_WIFI_STATE	n	n	n	n
AUTHENTICATE_ACCOUNTS	d	d	d	d
BLUETOOTH	d	d	d	d
BLUETOOTH_ADMIN	d	d	d	d
CALL_PHONE	d	d	d	d
CAMERA	d	d	d	d
GET_ACCOUNTS	n	n	n	n
GET_TASKS	d	d	d	d
INTERNET	d	d	d	d
MOUNT_UNMOUNT_FILESYSTEMS	s	d	d	d
PROCESS_OUTGOING_CALLS	d	d	d	d
READ_CALENDAR	d	d	d	d
READ_CALL_LOG	d	d	NA	NA
READ_CONTACTS	d	d	d	d
READ_EXTERNAL_STORAGE	n	n	NA	NA
READ_HISTORY_BOOKMARKS	d	d	d	d
READ_LOGS	s/t	s/t	d	d
READ_PHONE_STATE	d	d	d	d
READ_PROFILE	d	d	d	NA
READ_SMS	d	d	d	d
READ_SOCIAL_STREAM	d	d	d	NA
READ_USER_DICTIONARY	d	d	d	d
RECEIVE_MMS	d	d	d	d
RECEIVE_SMS	d	d	d	d
RECEIVE_WAP_PUSH	d	d	d	d
RECORD_AUDIO	d	d	d	d
SEND_SMS	d	d	d	d
SUBSCRIBED_FEEDS_READ	n	n	n	n
USE_CREDENTIALS	d	d	d	d
USE_SIP	d	d	d	d
WRITE_EXTERNAL_STORAGE	d	d	d	d

d:dangerous, s:signatureOrsystem, t:development, -:N/A, n: normal

Wang et al. (2013) use permission access level as a means to evaluate impact (`harm caused by a permission'). They follow the assumption that dangerous permissions are more harmful than normal permissions and appoint impact value to permissions in an ad hoc way. However, among these permissions there are several with `dangerous' access level that users consider more important than other in terms of privacy (Felt et al., 2012b; Lin et al., 2012). Each user has a varied risk perception and, thus, perceives the impact of disclosure to a particular type of PII differently. Therefore, he assigns a different impact value to privacy threats compared to other users. He may also consider specific types of personal data as more private or confidential. Such perceptions of impact are affected by the user's personality, awareness and technological expertise, the type of smartphone use (personal vs. business), as well as the

context of the user. Smartphone PIA can provide *generic* assessment results for the average user or a group of users, or *personalized* results based on an individual's perceived privacy impact.

6.4.2.1 Generic PIA

These types of assessments aim to assess the impact of privacy breaches to multiple users of a community, usually targeting a specific platform (Android, iOS, etc.). Such an approach gathers statistical data for multiple users and creates `risk profiles' for the `*average*' user, or categories of users. In a recent user study (Felt et al., 2012a), users assessed their perceived privacy impact in terms of `upsetness'. The results of this approach can be used as input to smartphone risk assessments, when combined with relevant threats. However, these results provide general assumptions in terms of privacy risk for multiple users. They can be used as an awareness tool, e.g. in the form of privacy alerts in the interface where the user grants permissions to apps.

6.4.2.2 Personalized PIA

This approach aims to create a personal `privacy risk profile' for a single user. This resembles a small scale risk assessment, for one device and one user, where the data owner responds to a questionnaire about his/her smartphone use or directly provides assessments for it, in a qualitative scale. The simplest way is to ask the user to assess the impact of a privacy breach or scenario. This can be achieved, as presented in the abstract risk assessment method for smartphones, by asking the user to answer predefined questions about the use of his smartphone, the data types he uses and how important they are in terms of security (including questions on confidentiality) (see §X). The idea is to ask general questions to the user that do not require knowledge of the data source (on the smartphone). The approach then maps the answers to relevant permissions, smartphone assets and threats and provides the input for smartphone risk assessment with the particular user/device in mind.

6.4.2.3 Mixed PIA

The main problem of generic PIA is that the results are static and have to be updated whenever a new user study emerges. An idea to address this issue is to collect data from personalized PIAs for users that do not wish to answer more detailed questionnaires or simply lack the time and/or technical expertise. User categories could reflect types of use (default, business, personal) or common characteristics, such as nationality, age, or interests. This could allow users to identify matching risk profiles and use input from them. A mixed approach allows a dynamic representation of the privacy concerns of a community (in this case Android users)

and assumes their willingness to participate by contributing their (personal) data (e.g. list of installed apps, installed data type, etc.).

6.4.3 Proposed PIA Method

This section describes an approach for privacy risk assessment which focuses on average smartphone users, i.e. ones without a security mindset and advance technical skills. As discussed in §XX, the users are assumed to (a). install applications (or 'apps') into their devices, on a regular basis (e.g. daily), (b). install them only from the official app repository or app market (i.e. Google Play) and (c). protect their smartphone only with the default security controls of the platform. This means that this chapter does not take into account smartphones with a modified operating system, either 'rooted', or with a custom version with additional third-party security mechanisms (software). With the former, it is assumed that every app is executed in a sandboxed, permission-based environment (Google, 2013a). The later, is in accordance with the expectations for an average user - in terms of security and ICT - as well as with the findings of the security awareness of smartphone users (see §XX). It refers to the poor adoption of third-party software that provides additional layers of security, such as anti-virus software, Mobile Device Management (MDM) solutions that isolate corporate from personal data, etc. Finally, it is assumed that the smartphone has Internet connectivity from the mobile carrier.

6.4.3.1 Threats and Permissions

In this chapter we consider the permissions of third-party applications, as *vulnerability*. This happens since, as discussed in §X.3.3, even if this application is benign (i.e. not leaking any private data for malicious purposes), its privilege to access private data may be misused by a malicious application performing a deputy attack or by and advertisement library that coexists with the application. Though, it might be the case that a malicious application masquerades as a benign application (e.g. game) luring users into downloading it and, thus, being the attack vector themselves. In this context, granting permission combinations is herein considered a vulnerability and will be used to assess the likelihood of occurrence of various privacy threats.

From our extended threat list that was presented previously, which combined similar threat lists found in the literature (namely: (Becher et al., 2011; Enck2011, Hogben and Dekker, 2010; Jeon et al., 2011; Lederm and Clarke, 2011; Shabtai et al., 2010)), only five threats were found to pose be associated with privacy impact. These threats were cross-referenced to the top mobile threats as defined by NIST (Souppaya and Scarfone, 2013), ENISA (Marinos

and Sfakianakis, 2012) and (OWASP, 2013). The resulting list of smartphone threats is presented in Table 57.

Table 57: Smartphone Privacy Threats

No.	*Threat*
T1	Tracking/ Surveillance
T2	Interception/ Eavesdropping
T3	Profiling
T4	Phishing
T5	Personal Information Disclosure

More specifically:

- *Tracking/surveillance* refers to monitoring users' context (e.g. via using the device's sensors). User's location can be retrieved either directly (fine location), or indirectly by using the sensors (e.g. a camera snapshot identifies the user's location), by retrieving information about the networks the user is connected to (coarse location), or calendar entries.
- *Interception/ Eavesdropping* refers to unlawful interception of communications and is applicable to all communication data types (including external data of communication, such as the call log).
- *Profiling* refers to user activity monitoring, but for advertising purposes.
- *Phishing* is closely related to identity theft and refers to tricking the user to disclose credentials.
- *Personal Information Disclosure* refers to the disclosure of all other types of personal information, which do not fall in the other four threat types. These include documents, media and sensitive files of the user.

It is worth noting that our analysis omits threats that could be used as the attack vector for a privacy breach. For instance, a successful spoofing attack can lead to other privacy attacks, such as phishing, eavesdropping, SSL downgrading etc. (Souppaya and Scarfone, 2013). Moreover, threats that exploit vulnerabilities in the OS, which are not related to a specific permission, or refer to vulnerabilities introduced by `jailbroken' or `rooted' smartphones (e.g. weak ssh credentials as in IKee (Cluley, 2009) are also out of the scope of the analysis. Also, spyware is not included as a threat, as it would cover several of the above threats depending on the collection purpose. In this subsection, every application that has a hidden mechanism to collect personal data without the user's consent is considered to be `spyware'. For these threats, risk assessment is more complex and needs to take into account several other factors, such as non-privacy threats and vulnerabilities, installed controls, user habits in terms of OS updates and patches, and user security awareness. The aforementioned fall outside the scope of a user-centric PIA and will not be discussed in the sequel.

In Table 58 we map the personal data of an Android user to permissions that can enable a privacy breach and identify applicable threats to these data types. In order for an application to disclose these information to a third party, a data channel is required. There are three (3) available channels, namely: (a). Internet connection, (b). short messages and (c). Bluetooth. Each of them is protected with a *dangerous* permission, namely: (a). `INTERNET`, (b). `SEND_SMS` and (c). `BLUETOOTH`.

If the user's application list is available during the privacy assessment, we can then assess the threat likelihood of privacy threats, by identifying `suspicious' permission combinations that may enable them in the smartphone of a user. Finding an app with a privacy-related permission does not mean, however, that the application is malicious; it means that the application could be potentially used for other purposes than the stated ones. Installing an application that retrieves the application list of a user may be considered too intrusive and may add additional vulnerabilities to the risk profile (of the smartphone/user) we are currently studying. On the other hand, asking the user to 'manually' provide the application list may be cumbersome, especially when the user has limited time and/or many installed applications, whereas asking her device for a manual audit from an analyst may also considered too intrusive for the user. In this context, a less intrusive approach is explained in the sequel.

The user describes the type of smartphone use, by declaring her popular application categories. This way she indicates what apps she installs and uses often (e.g. on a daily basis). Based on this input and our statistical analysis on the applications in the Google Play (see section X.4), each category yields varied privacy-sensitive combination of permissions, thus, a varied level of privacy vulnerabilities. Since each threat is enabled by a set of permission combinations (c.f. Table 58), *per user threat likelihood* is the *avg* value of the above frequencies in each category specified by the user. Finally, the *per user vulnerability level* is assessed based on the *avg* frequency of a particular permission combination in the selected categories, based on the following empirical semi-quantitative scale:

1. *Negligible*: <10%
2. *Limited*: ≥ 10% and <40%
3. *Significant*: ≥ 40% and <70%
4. *Maximum*: ≥ 70%

The four levels of vulnerability CNIL are *dynamically* created by periodically clustering (k-means algorithm) the likelihood values for the top combinations of channel and asset permissions (see Table 60). A four-item scale was selected in order to match the impact assessments values, adapted by (Government of Canada).

6.4.3.2 Threats and Impacts

In order to assess the impact of a privacy breach to a smartphone, we adjusted the `*Methodology for Privacy Risk Management*' by CNIL (Government of Canada) to the smartphone context. Initially, for each permission identified in Table 58, we assign a *level of identification*. This parameter refers to the ability to identify an individual solely by having access to the data that this permission protects. This is important because in order for a privacy threat to have an effect, it needs to correspond to a particular individual whose privacy has been breached. It is worth noting that the ability to identify this individual must not be confused with *profiling* (T3), which refers to monitoring a user's activity (e.g. among different

Table 58: Mapping of Data Assets, Permissions and Threats

Data Category	Data Asset Type	Permission	Privacy Threats				
			T1	T2	T3	T4	T5
Communication	SMS	RECEIVE_SMS		✓			
		READ_SMS		✓			
	MMS	RECEIVE_MMS		✓			
	Voice	PROCESS_OUTGOING_CALLS		✓			
	Wap Push	RECEIVE_WAP_PUSH		✓			
Sensor/Location	Video	CAMERA	✓	✓			
	Audio	RECORD_AUDIO	✓	✓			
	Location	ACCESS_COARSE_LOCATION	✓		✓		
		ACCESS_FINE_LOCATION	✓		✓		
		BLUETOOTH_ADMIN	✓		✓		
		ACCESS_NETWORK_STATE	✓		✓		
		ACCESS_WIFI_STATE	✓		✓		
		READ_SOCIAL_STREAM	✓		✓		
External Storage		WRITE_EXTERNAL_STORAG		✓	✓		✓
		READ_EXTERNAL_STORAGE		✓	✓		✓
Contacts		READ_CONTACTS		✓			✓
		READ_SOCIAL_STREAM		✓			✓
History/ Usage		READ_CALL_LOG		✓	✓		
		READ_HISTORY_BOOKMARK			✓		
		GET_TASKS			✓		
		READ_LOGS			✓		
		READ_USER_DICTIONARY			✓		✓
		READ_SOCIAL_STREAM			✓		
		SUBSCRIBED_FEEDS_READ			✓		
Calendar		READ_CALENDAR	✓				✓
Identity		READ_PROFILE			✓	✓	✓
		GET_ACCOUNTS			✓	✓	✓
		READ_PHONE_STATE			✓	✓	✓
Credentials		READ_SMS- RECEIVE_SMS				✓	
		AUTHENTICATE_ACCOUNTS				✓	
		USE_CREDENTIALS				✓	

applications, web sites) for advertising purposes. The *Identification Level* of a permission is assessed in the following four-item scale:

1. *Negligible*: Identifying a user using this permission appears to be virtually impossible.

2. *Limited*: Identifying a user using this permission appears to be difficult but is possible in certain cases.
3. *Significant*: Identifying a user using this permission appears to be relatively easy.
4. *Maximum*: Identifying a user using this permission appears to be extremely easy.

The proposed levels of identification per Android permission can be found in Appendix A. This static mapping of permissions to levels of identification should be adjusted when changes in permissions occur.

The user is then asked a few questions, in order to assess his *individual impact* of a privacy breach. Each question describes the effect of disclosure or misuse on various personal data. Therefore, based on Table 58, each question corresponds to specific applicable threats. An example of the short questionnaire, coupled with applicable threat scenarios can be found in Appendix B. The questionnaire's answers are also predefined. Each one describes the effect of the privacy threat to the user. In Appendix D we map predefined answers to their corresponding *Severity Level*, based on the following qualitative scale:

1. *Negligible*: The user is either not affected or may en- counter a few inconveniences, which he/she can overcome without any problem.
2. *Limited*: The user may encounter significant inconveniences, which he/she will be able to overcome despite a few difficulties.
3. *Significant*: The user may encounter significant consequences, which he/she should be able to overcome albeit with serious difficulties.
4. *Maximum*: The user may encounter significant, or even irreversible, consequences, which they may not overcome.

The overall *Impact Level* of a specific threat scenario is then assessed as the sum of *Identification Level* and *Severity Level*, based on the following scale:

1. *Negligible*: <5
2. *Limited*:= 5
3. *Significant*: =6
4. *Maximum*: >6

6.4.3.3 Risk of Privacy Threat

Since each privacy threat requires the presence of specific combinations of permissions on the user's smartphone, the *Level of Risk per privacy threat* is assessed as a sum of *Impact Level* and *Vulnerability Level*, in the following scale:

1. *Negligible*: 1 to 3
2. *Limited*: 4 to 5
3. *Significant*: 6 to 7
4. *Maximum*: 8

This value corresponds to a user's specific risk profile (*personalized PIA*}), as (a). impact was assessed based on the user's input and reflects his own assumptions regarding the potential effects of a privacy breach and (b). threat likelihood (i.e. vulnerability level) is assessed according to the type of smartphone use the particular user describes, by identifying her common app categories.

6.4.4 Case study: Privacy assessment of Android users

This section describes a proof-of-concept case study of our proposed method for privacy assessment.

6.4.4.1 Case study scenario

In this proof-of-concept case study we examine the following two use cases.

- *User A* is a teenager who uses his smartphone mainly for leisure, which includes socialising with his classmates and friends, making phone calls and texting, playing games (only sports games) and listening to music. The user specified that his applications fall mainly on the following categories: (a). *Communication* (such as web browser, custom email (e.g. yahoo), chatting apps, etc.), (b). *Sport_Games*, (c). *Social* (e.g. Facebook, Twitter) and (d). *Music_And_Audio*.
- User B is a businessman who uses his smartphone for both business and leisure. His daily use of Android applications includes (a) reading *News_and_Magazines* (i.e. apps displaying the content of news web sites (e.g. as CNN)), (b). *Socializing* (Facebook, Twitter) with colleagues or clients, (c). consulting the *Weather* and (d). reading maps and navigating by GPS (*Travel_And_Local* category) as he regularly travels to visit his clients.[34]

Error! Reference source not found. summarizes the responses of the two users in questions Q1, Q6-8 and Q10 (see appendix B for the corresponding text of the questions and answers), as well as their mapping to the severity level of our proposed method for privacy assessment.

[34] Any business data, such as corporate files (e.g. pdf) stored on the external storage, were not included in the case, as these are not considered PII, do not affect privacy and are under different regulatory requirements. We only examine the effect to a person's reputation, which falls into the scope of privacy and may affect his working conditions.

Table 59: Questionnaire answers of the case study

Question	*UserA*		*User B*	
	Response	Severity Level	Response	Severity Level
Q1 (Fine)	N3	1	L2	2
Q1 (Coarse)	N3	1	L2	2
Q6	N/A	N/A	N1	1
Q7	S5	3	N3	1
Q8	N1	1	N3	1
Q10	L4	2	S1	3

6.4.4.2 Statistics for Combinations of Permissions

To compute the frequency of permission combinations, we crawled apps residing in Google Play from May to June 2013. We collected all the available apps that are indexed in each app category of Google Play, namely 27673 apps. We then analyzed all possible combinations that include pairs of permissions that protect an asset described in Table 58 and a transmission channel. Our sample contains 89 such pairs permission combinations (see Appendix). Among them, the most frequent pairs in permission combinations are presented in the sequel (see Table 60).

Each question examines a different threat scenario and, therefore, it requires a different permission combination present, in order to be realized. For readability reasons, the subsequent analysis will include only the following permission combinations C_i, which correspond to the threats that are covered with the above questions, namely:

- C_1: `INTERNET, ACCESS_FINE_LOCATION`
- C_2: `INTERNET, ACCESS_COARSE_LOCATION`
- C_3: `INTERNET, READ_PHONE_STATE, READ_CALENDAR`
- C_4: `INTERNET, GET_ACCOUNTS, READ_HISTORY_BOOKMARKS`
- C_{5A}: `INTERNET, READ_PHONE_STATE, READ_CONTACTS`
- C_{5B}: `INTERNET, READ_PHONE_STATE, READ_CALL_LOG`
- C_6: `INTERNET, READ_EXTERNAL_STORAGE, READ_PHONE_STATE`

Per user threat likelihood is the *avg* value of the frequency of the above combinations in each app category that matches the user's profile. Assuming that the businessman possesses a smartphone with a more modern Android version than the teenager, then the combination for C_5 differs. Access to call history is protected by the permissions `READ_CALL_LOG` and `READ_CONTACTS`, for the businessman and teenager respectively (Google, 2013d). Hence, the threat that is realized with the occurrence of C_5 (i.e. T3) gets different scores in the *vulnerability level* in the common category (i.e. Social), as a result of the two different combinations C_{5A}, C_{5B}. Table 61 presents the frequency of the 6 combinations of permission that are studied in section 4.

Table 60: Top 30 combinations for privacy violating permissions (n=27673 apps, collected May-June 2013)

Channel	*Access to Data*	*Frequency*
INTERNET	ACCESS_NETWORK_STATE	81,13%
INTERNET	WRITE_EXTERNAL_STORAGE	55,14%
INTERNET	READ_EXTERNAL_STORAGE	55,08%
INTERNET	READ_PHONE_STATE	48,92%
INTERNET	ACCESS_WIFI_STATE	31,06%
INTERNET	ACCESS_COARSE_LOCATION	28,67%
INTERNET	ACCESS_FINE_LOCATION	27,75%
INTERNET	GET_ACCOUNTS	18,85%
INTERNET	CAMERA	8,19%
INTERNET	GET_TASKS	7,17%
INTERNET	READ_CONTACTS	6,94%
INTERNET	READ_HISTORY_BOOKMARKS	6,75%
INTERNET	READ_CALL_LOG	5,67%
INTERNET	RECORD_AUDIO	4,63%
INTERNET	READ_LOGS	3,36%
INTERNET	USE_CREDENTIALS	1,92%
SEND_SMS	READ_PHONE_STATE	1,75%
INTERNET	RECEIVE_SMS	1,72%
SEND_SMS	ACCESS_NETWORK_STATE	1,71%
SEND_SMS	WRITE_EXTERNAL_STORAGE	1,68%
SEND_SMS	READ_EXTERNAL_STORAGE	1,68%
INTERNET	READ_SMS	1,45%
INTERNET	BLUETOOTH_ADMIN	1,29%
SEND_SMS	ACCESS_FINE_LOCATION	1,26%
SEND_SMS	ACCESS_COARSE_LOCATION	1,20%
SEND_SMS	READ_CONTACTS	1,18%
INTERNET	READ_CALENDAR	1,16%
SEND_SMS	READ_CALL_LOG	0,98%
SEND_SMS	ACCESS_WIFI_STATE	0,91%
SEND_SMS	RECEIVE_SMS	0,89%

Table 61: Frequency of permission combinations for *user A* (n=27673 apps, collected May-June 2013)

Combination	**Sport**	**Social**	**Communication**	**Music**
C_1	21,5	36,3	22,0	26,5
C_2	27,7	33,1	19,9	28,9
C_3	0,3	1,6	1,6	1,1
C_4	4,3	2,5	2,1	3,5
C_{5A}	5,0	12,4	27,7	12,2
C_6	43,4	33,4	37,4	45,1

Table 62: Frequency of permission combinations for *user B* (n=27673 apps, collected May-June 2013)

Combination	**Social**	**Weather**	**Travel**	**News and Magazines**

C_1	36,3	51,7	72,0	25,2
C_2	33,1	49,9	53,3	26,5
C_3	1,6	0,6	1,1	0,5
C_4	2,5	1,9	1,5	0,7
C_{5B}	9,2	2,1	5,2	1,0
C_6	33,4	18,6	27,5	36,5

6.4.4.3 Risk of Privacy Threats

Table 63-61 summarize the vulnerability levels for per user threat, as well as the identification level, impact level and risk level of this case study.

The case study includes permission combinations with different identification levels (c.f. Table 63). For instance, C_1 is assessed with *negligible* identification level (the device location provides only a weak correlation with the user's identity), whereas C_6 is assessed with *maximum* identification level, due to the unique identifiers (e.g. IMSI, IMEI) accessible via the `READ_PHONE_STATE` permission.

For the threats involving access to the device location, the vulnerability level of the businessman is greater due to his preference to navigation apps. The teenager has a greater vulnerability level for the threats that involve access to calling history, while the rest threats have a similar vulnerability level for the combination of permissions that are included in the case study's scope (c.f. Table 63-61).

The businessman responded that he is more upset about unauthorized access to his data in the external storage and the teenager about disclosure of his browsing history to his friends. For the former, this is the threat with the highest risk level (RL=3, c.f. Table 63). For the latter, the three threats that are realized with the combinations C_4-C_6 were assessed as highest with our method (RL=2). Finally, it is assumed that the teenager responded that he is not using his calendar. Thus, the threat that is realized with C_3 is not applicable to him and no risk score is assigned to it.

Table 63: Case study results for *user A*

Combination	*Threat*	*Identification Level*	*Impact Level*	*Vulnerability Level*	*Risk Level*
C_1	T_1	1	1	2	1
C_2	T_1	1	1	2	1
C_3	T_1	N/A	N/A	N/A	N/A
C_4	T_5	3	3	1	2
C_5	T_3	4	2	2	2
C_6	T_5	4	3	2	2

Table 64: Case study results for *user B*

Combination	*Threat*	*Identification Level*	*Impact Level*	*Vulnerability Level*	*Risk Level*
C_1	T_1	1	1	3	2
C_2	T_1	1	1	3	2

C_3	T_1	4	2	1	1
C_4	T_5	3	1	1	1
C_5	T_3	4	2	1	1
C_6	T_5	4	4	2	3

6.5 Discussion

This chapter presented two mitigation methods for tackling privacy violations of smartphone users, namely: (a) a prediction model for security unaware users, which allows organizations to identify and engage those users in awareness programs, and, (b) a risk assessment method tailored for smartphones, which does not consider the device as a single entity but considers the sub-assets that smartphones include. Both of them aim to be user centric, i.e. they are not highly dependent on the security 'nuts and bolts' of the platform and they require input from the user for their success.

As discussed in the previous chapter, a user's belief that applications deriving from the official repository are secure is considered a severe vulnerability. This holds true since, trust in the application repository is more likely to: (a) be encountered in users who are not technology and security savvy, and (b) increase the users' security complacency, as they are more likely not to scrutinize security messages and not use secsoft software (c.f. chapter X for a more thorough discussion). Moreover, this trust is not always justified by the availability of security controls in the app repository (e.g. app testing), or their effectiveness regarding the mitigation of malware. Hence, this user security complacency regarding trust in the app repository may be exploited by attackers who use the repository as a malware infection vector. The users who trust the application repository cannot be identified via software solutions (e.g. software agents, MDM). As such, in this chapter we provided a prediction model that identifies them. This model can be used by an organization to identify users who have the above mentioned security complacency regarding the app repository and engage them in a relevant training awareness program. The results of the model can be also used as input to the proposed risk assessment method for smartphone, to adjust per-user threat likelihood for the threat of installation of malicious applications from the official app repository.

The model is prone to Type I errors, which may cause frustration to a user who does not trust the app repository. For instance, such a user may be asked to undergo relevant security training, which might be unnecessary. We argue that that a significant Type I error coupled with a negligible Type II error - as found in the performance of our model - is a good security trade-off, as far as smartphone awareness is concerned. The suggested model performs well enough, even if it is evaluated in a different sample than the one used for its design. As the latter sample includes smartphone users from the United Kingdom, this suggests that the proposed model is not affected by the demographics of the Greek sample.

Our second threat mitigation method that we proposed, i.e. the risk assessment method that is tailored for smartphones, is compatible with established guidelines on risk assessment (ISO, 2008). Contrarily to traditional risk assessment methods, which treat smartphones as a single entity, this method provides a more fine-grained valuation by: (a) dividing the device into various sub-assets, (b) assessing smartphone-specific threats, and (c) taking into account the characteristics of a smartphone security model.

User input for sub-asset impact is based on a two-dimensional data taxonomy. The data analysis takes place transparently to the user and it leads to a 'personalised' risk assessment, as opposed to other generic smartphone-oriented methods, which use mainly expert opinions, such as e.g. (Hogben and Deker, 2010; Shabtai et al., 2010). The level of input detail varies according to user skill (Lederm and Clarke, 2011) - this may indeed affect the quality of results - but our approach requires minimum input, i.e. the data impact valuation. The method could be potentially used in order to extend an information risk assessment method to include smartphone-specific threats, as its theoretical basis is compatible with best practices, i.e. ISO27005 (ISO, 2008).

This is used in combination with a threat list to conduct risk assessment. The list was compiled by extending existing threat lists of smartphone literature. For threat assessment, risk triplets are introduced, which makes the approach novel. They use application permissions as the attack vector, associating assets to threats and permission combinations. Risk is, then, assessed as a combination of asset impact and threat likelihood. Finally, a demonstration of the proposed assessment method is provided, via a case study based on the Android platform.

It should be noted that generic conclusions for specific threats cannot be drawn by 'high' risk valuations of a hypothetical, single case. Risk is highly affected by the valuation of impact, which is a parameter that varies among different users. Our risk assessment method can be applied in other smartphone platforms with permission-based security models (e.g. Symbian, Windows Phone, etc.), as well as in other platforms (e.g. iOS, BlackBerry, etc.) with some minor adjustments.

Finally, this chapter also included a more concrete application of the abstract risk assessment method, which is tailored for Android's security model. It described a method for Privacy Impact Assessment (PIA) for Android applications. As opposed to previous works that rely static or dynamic analysis techniques on the app's file in order to assess privacy violations, our approach is user-centered. More specifically, the cornerstone of our assessment is impact valuation from the user, as well as his/her usage profile, which enables per user risk assessment. Threat likelihood is assessed based on the presence of specific permission combinations, which we consider to be vulnerabilities that enable privacy threat scenarios. For the demonstration of the method, a case study is presented with input from two hypothetical users and actual app data, i.e. permission combinations from apps in Google Play. Our proposed

method is envisioned as a complement to the existing protection from privacy violating apps. For instance, the privacy risk level of a user can be used to generate app analysis policies, which would filter out Android apps based on user privacy requirements, or be used to provide security awareness that is tailored to the user's risk profile.

The dynamic computation of permission combinations, which are used as input by our method, is limited only to applications that are available from Google Play. The frequency of permission combinations in other app marketplaces may be different, implying different threat likelihood. Our method assumes the participation of the user and is prone to the subjectiveness of his/her impact perceptions. Also, our analysis could be extended to include the existence of safeguards that may decrease threat likelihood - even though our work has proven that currently smartphone users poorly adopt them.

6.6 Summary

Personal information can be found in almost every smartphone. This holds true, since these devices manage a variety of data and are often used as a single device both for business and personal purposes. The proliferation of smartphone applications and, along with them, the increasing presence of privacy violating apps in official app repositories or marketplaces, such as the Google Play, pose a significant privacy risk for average smartphone users.

This chapter presented an abstract method for risk assessment that is tailored for smartphones and a prediction model for security unaware users, as safeguards against privacy violations that occur from malicious. Our risk assessment method does not treat this kind of device as a single entity. Instead, it identifies smartphone assets and provides a detailed list of specific applicable threats. For threats that use application permissions as the attack vector, risk triplets are facilitated. The triplets associate assets to threats and permission combinations. Then, risk is assessed as a combination of asset impact and threat likelihood. The method utilizes user input, with respect to impact valuation, coupled with statistics for threat likelihood calculation. As a case study, we demonstrate the realization of a privacy impact assessment of Android applications, which are available in Google Play. Finally, we built a prediction model to indentify users who trust the app repository, an erroneous perception that was found to ‘lower the users’ defenses’ and cannot be identified with software solutions (e.g. software agents, MDM). The model is assessed, evaluated, and proved to be statistically significant and efficient. The model can be used to identify security unaware smartphone users, which can be later be engaged in an appropriate security training program.

Chapter 6: Proactive smartphone forensics

"*'There is nothing like first-hand evidence.*" —A Study in Scarlet

7.1 Introduction

The proliferation of smartphones, which are devices with increasing popularity due to their processing and network capabilities, low cost, and dimensions, introduces new challenges to digital forensics. This holds true, as the heterogeneity and progressive evolution of smartphone platforms increase the complexity of a digital investigation, together with the skills and toolset required to respond to a crime or incident. For instance, the different hardware and software, which is offered by smartphone platforms to increase their market share, hinder a forensic examination. This happens since dedicated software has to be implemented and cables to be acquired per case. Moreover, as smartphone cables (data or power cables) tend to be available only as long as a smartphone model is in the market. More difficulties appear in the examination of old devices that are not supported anymore.

Smartphones are characterized by mobility, context-awareness, and diversity on the data sources that they integrate (see §XX). Along with the popularity of smartphones, the one of sensors (i.e. hardware that measures the user's context) is also increasing in third-party software, such as games and navigation apps. Sensor data may be useful in certain crimes in order to lawfully deduce the user's context. This context may provide crucial information for law enforcement authorities in a digital investigation both for the detection and sanction and the prevention of crime. In the same time, this kind of data collection is often intrusive and may raise ethical and legal issues pertaining to the secrecy of telecommunication and the privacy of the concerned persons. Due to the interference with these fundamental rights, such forms of surveillance and collection of the respective digital evidence may be implemented only in relation to crimes that are regarded by the law as "severe" or "national security threats". A generalized and indiscriminate collection of probably useful evidence against individuals at random would perceive all citizens as potentially future wrongdoers. Such a model of routine intelligence gathering against random citizens could "blow up the cornerstones of the rule of law state" (Hoffmann-Riem, 2002).

Smartphone forensics is an emerging discipline which currently appears to be still in its infancy. Nonetheless, the consumerization of smartphones has triggered the forensic community to focus on these devices' technical details as well as the collection of their data. Therefore, tools have been created which perform either physical (i.e. low level, bit to bit), or logical (i.e. data organized in logical objects, e.g. directories) data acquisition in: (a) Windows Mobile

(Casey et al., 2010; Klaver, 2010; Rehault, 2010; Grispos, 2011), (b) Symbian (Mokhonoana and Olivier, 2007; Distefano and Me, 2008; Pooters, 2010; Thing and Chua, 2012; Thing and Tan, 2012), (c) iOS (Zdziarski, 2008; Bader and Baggili, 2010; Husain and Sridhar, 2010; Jung et al., 2011), and (d) Android (Lessard and Kessler, 2010; Thing et al., 2010; Vidas et al., 2011; Hoog, 2011; Mutawa et al., 2012; Park et al., 2012; Sylve, 2012).

In this chapter we discuss *proactive forensics* for smartphones. More specifically, we propose the *ad-hoc collection* of smartphone data by law enforcement authorities in a digital investigation both for the detection and sanction and the prevention of crime. In specific cases and under certain conditions, the law may allow collection of smartphone data for proactive use and for the purpose of crime prevention. Furthermore, proactive collection of a smartphone's data may take place in critical infrastructures for incident response and forensic readiness reasons - when the device is considered part of this infrastructure and the use of such intrusive control fully complies with the organization's security policy, its culture and the existing legal framework. For instance, the security policy of a critical infrastructure may specify employee usage profiling - assuming the employee's informed consent - in order to mitigate the insider threat via an insider prediction model (Kandias, et al., 2010). To avoid the misuse of the collection mechanism by law enforcement - or any malicious individual - in a fashion of routine intelligence gathering where data are collected from random citizens, we describe a scheme for the ad-hoc collection of smartphone data.

Finally, we implement a system for remote and ad-hoc (sensor) data acquisition from smartphones. Until now the practicality of sensor data collection has not been adequately studied in the forensics literature. This is because they are time-sensitive and volatile data and therefore not available for analysis during post-mortem forensics.

The chapter contributes the following:

(a) Discussion of proactive forensics in smartphones, i.e. the ad-hoc collection of smartphone data by law enforcement authorities in a digital investigation both for the detection and sanction and the prevention of crime.
(b) Proposition of a scheme to avoid the misuse of the data collection mechanism of proactive forensics for smartphones.
(c) Development of an investigation frameworks' taxonomy for smartphone forensics and a taxonomy of smartphone data with transport channels and digital evidence.
(d) Study of the smartphone sensors as a potential data source for digital forensics.
(e) Identification of vulnerabilities in Android's security model that enables the ad-hoc acquisition of sensors data.

(f) Implementation of the acquisition software and the transport protocol of the proactive investigation scheme of (b) for Android devices. Evaluation of the implementation and sharing of the experience regarding the collection of sensor data.

(g) In depth discussion of legal and ethical issues that arise from the ad-hoc collection of sensor data.

The chapter is organized as follows. The next section discuses proactive digital forensics on smartphones, including a taxonomy of the digital investigation frameworks that can be used in smartphone forensics as well as a taxonomy of digital evidence with smartphone data sources and their transmission channels. In section 4, we present our implementation for remote, ad-hoc sensor data acquisition from smartphones. Then, we summarize our experience regarding the data collection of smartphone sensors by our tool and describe a number of scenarios regarding its use and preparation (section 5). Section 6, includes the ethical and legal issues that arise from the ad-hoc data acquisition from smartphone sensors. Finally, section 7 includes a discussion of our approach and conclusions.

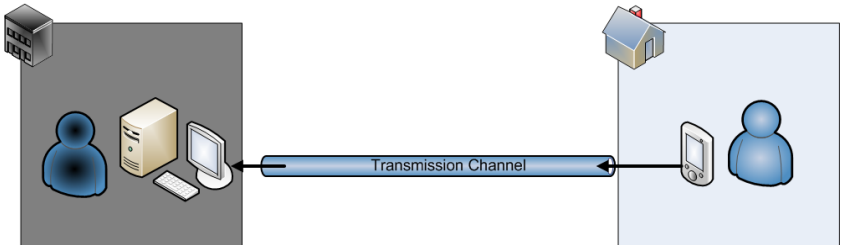
7.2 Proactive forensics for smartphones

Smartphones, as ubiquitous devices, merge with a person's everyday life. They are characterized by mobility, context-awareness, and diversity on the data sources that they integrate. In a crime investigation, the aforementioned characteristics can be used for forensic purposes, not only after a crime, but even proactively both for the detection and sanction and the prevention of crime. For instance, in some crimes, which the in place legal and regulatory context regards as 'severe' (e.g. crimes against public or the state, pedophilia, etc.), proactive acquisition of smartphone data may be required. Ad-hoc collection of smartphone data requires the same technical skills and techniques (e.g. spyware, social engineering), which are currently used by attackers to gain unauthorized access to access data from smartphone users (see

Figure 13). Similarly, under proactive forensics for smartphones, law enforcement accesses smartphone data, which must be forensically sound, only from a suspect and in the specific cases that were noted beforehand.

Currently, in a Lawful Interception (LI) of cell phones, Law Enforcement Agencies engage via the carrier's infrastructure, the interception of specific data, such as phone calls, messaging services, and network data traffic (ETSI, 2009). The need for direct and in-time access to forensics data is also present in other technological contexts.

Figure 13: Attacker collects smartphone data from a user (privacy violation)

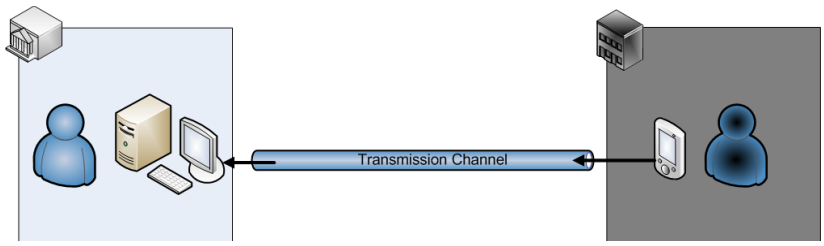


In the organizational context, Grobler et al. (2010) refer to proactivity, as "creating or controlling a situation rather just responding to it". They stress that a Proactive Digital Forensics (ProDF) as a process, ensures the facilitation of the investigation in a successful and cost-effective manner for enterprise systems. ProDF refers, for example, to the deployment of *forensic readiness* processes (Tan, 2001), which aim to maximize an environment's ability to collect credible digital evidence and, at the same time, minimize the forensics cost during incident response. These processes incorporate tools and techniques for active/live forensics that acquire volatile data for detecting criminal activity (Sutherland et al., 2008).

In traditional, i.e. *post mortem*, forensics the need for *forensic triage* (Mislan et al, 2010; Rogers et al., 2006) emerges. Forensic triage refers to the notion of *on-site forensics*, which allows an investigator to directly assess a crime scene. Assessment is feasible for a subset of the available digital evidence. Consequently, delays on results, stemming from time-consuming processes in the forensics lab, are avoided. Furthermore, in on-going crime investigations, forensic triage can assist an investigator to determine critical issues, such as subjects in immediate need or the suspect's call activity (Walls et al., 2011), and act as appropriate. Also, mobile live memory forensic techniques focus on the acquisition of volatile data, which reside in device's memory (Thing et al., 2010).

The rest of this subsection includes a discussion of the digital evidence types and the ways they can be remotely collected from a smartphone as well as a taxonomy of the investigation frameworks that have been proposed from smartphone forensics.

Figure 14: Law enforcement collects smartphone data from a suspect (proactive forensics)



7.2.1 Smartphone Evidence

As discussed in §XX, smartphones host a plethora of heterogeneous data generated from hardware or software sources. This section associates smartphone data sources with evidence types and then correlates these sources with smartphone evidence transport channels. These associations will be used in the sequel in the scheme for ad-hoc evidence acquisition from smartphones.

7.2.1.1 Smartphone Evidence Taxonomy

In order to associate smartphone data to evidence types, a data-oriented analysis was followed. The analysis used a smartphone data source taxonomy that was presented in §XX. Under this taxonomy smartphone data are categorized, with respect to their source, as:

- Messaging Data
- Device data
- (U)SIM Card Data
- Usage History Data
- Application Data
- Sensor Data
- User Input Data

These data sources were organized in *a taxonomy* of *evidence types*, which derived from a set of questions - i.e. {*who, where, when, what, why, how*} (Zahman, 1987). These questions are being used in digital forensics literature (Ieong, 2006; Jansen and Ayers, 2007; Blackwell, 2011) for evidence examination and analysis, as well as for evidence presentation in courts of law.

Table 65: Taxonomy of evidence types derived from Zahman's framework

Evidence Type	Description
Identity Evidence	Data identify subjects that are part of an event
Location Evidence	Data define the approximate or exact location, where an event takes place
Time Evidence	Data can be used to infer the time that an event takes place
Context Evidence	Data provide adequate context, such as user actions and activities for an event description (Lane et al, 2010), or the event nature
Motivation Evidence	Data can be used to determine event motivation
Means Evidence	Data describe the way that an event took place, or the mean that were used
Identity Evidence	Data identify subjects that are part of an event

Even if we both assume that smartphone data are generated in a deterministic and undisturbed way and their acquisition is always feasible, this does not necessarily mean that evi-

dence will always be present. Hence, we use three levels of association of *direct* relationship between a data source and an evidence type, namely: a) *Strong Correlation* to represent that data sources always (or in most cases) provide potential evidence, b) *Weak Correlation* to represent that data sources may provide potential evidence according to their state, and c) *No Correlation* to represent lack of relationship between a data source and evidence type. In this point we note that motivation evidence may be deduced *indirectly* via the combination of the other evidence types.

The rest of this section presents the correlation of the evidence types with the data sources.

- *Messaging Data*. Both traditional messaging services (e.g. SMS) and modern ones, e.g. email, often constitute potential evidence. Specifically, external communication data reveal the subjects and the communication time and, as a result, this source is strongly correlated with time and identity evidence. In addition, the communication content may reveal location evidence, motive evidence, etc., therefore this source is weakly correlated with them as well.
- *Device Data*. Data such as system identifiers (e.g. IMEI) can be used to determine a subject from the provider's records; therefore a strong correlation with identity evidence exists. File system metadata can be used to determine time (e.g. file access time). Hence, a strong correlation with time evidence exists. Data created by the user (e.g. multimedia) may reveal motive or means evidence. In addition, device data may include sensor data as metadata (e.g. in geo-tagging). This provides a weak correlation with location, context, motivation or means of an incident.
- *(U)SIM Card Data* includes identifiers (e.g. ICCID[35], IMSI, etc.) that uniquely identify a device owner, thus, a strong correlation exists. Other data that may be stored in this source (e.g. contact entries, SMS and LAI[36]) (Jansen and Ayers, 2007) can be used to deduce the rest evidence types, and, in this case a weak correlation exists.
- *Usage History Data* can infer the event time and the means used by a digital incident, or they can even reconstruct user events, thus implying a strong correlation with the respective evidence types. Furthermore, under certain circumstances, the wireless connection history (i.e. access point MAC logs) may be used to infer the device location[37], implying a weak correlation with this evidence type. Finally, Bluetooth pairing logs can be used to infer whether the user is in a crowded area and, in some cases, identify subject evidence via the device Bluetooth id.

[35] Integrated Circuit Card ID (ICCID) is the serial number of the (U)SIM card.
[36] Simply put the Location Area Identity (LAI) identifies the cell where the device is in and as a result can be used to get an approximation of device location.
[37] Via public mapping of MAC addresses to GPS coordinates, such as http://samy.pl/mapxss/

- *Application Data*. In some cases, private application data stored on the device may lead to potential evidence about an event and, thus, weak associations with the corresponding evidence exist. For instance: a) cached maps in navigation applications determine location evidence, b) cached social networking application data can be used to infer all the other evidence types depending on their content, etc.
- *Sensor Data* provide weak correlations with all evidence types, since they can be used to infer the device context. For instance, a microphone can be remotely enabled (Mylonas, 2008) and harvest speech data related to all evidence types. Nonetheless, in the case of location evidences, the correlation is strong due to the popularity and location accuracy of GPS sensors.
- *User Input Data* can be used to identify a subject via keystroke analysis and, as a result; a weak correlation with this evidence type exists. Often, the keystroke cache content may reveal other evidence types, thus, a weak correlation with them exists.

Table 66 depicts the correlations between data sources and potential evidence types, where: (✓) stands for strong correlation, (~) for weak and (✘) for no correlation.

Finally, the above evidence types can be combined to form evidence chains when they include valid timestamps. For instance, call logs combined with cached data of a navigation application can be used to reveal or confirm a subject's alibi.

Table 66. Correlation between evidence types and data sources

Evidence Types Data sources	Identity	Location	Time	Context	Motivation	Means
Messaging Data	✓	~	✓	~	~	~
Device data	✓	~	✓	~	~	~
(U)SIM Card Data	✓	~	~	~	~	~
Usage History Data	✘	~	✓	~	✘	✓
Application Data	~	~	~	~	~	~
Sensor Data	~	✓	~	~	~	~
User Input Data	~	~	~	~	~	~

7.2.1.2 Transport Channels' Taxonomy

Smartphones can use four data transport channels (or interfaces) that provide different transport services. This section discusses their ability to support evidence transfer during a proactive forensic investigation.

1. *GSM Messaging interface* (e.g. SMS, etc.) provides a remote channel appropriate for small volume data transfers, which is nearly always available. Apart from the restriction in volume, another refers to cost. Increased cost a) may limit the messaging service availability, thus, large data may not be transferrable and b) may alert suspects, who thoroughly check their carrier bills in the case a proactive investigation taking place.

2. *Personal Area Network (PAN) interface* (e.g. Bluetooth, IrDA, etc.) provides a cost free, ad hoc, remote data channel, appropriate if the data collector is in the smartphone's proximity. By not relying on any base station existence, it avoids network monitoring mechanisms, such as Intrusion Detection System (IDS). Furthermore, as this channel requires no cost, it is stealthier than others. Potential shortcomings are: a) distance constraint between the device and the collector, which increases attack complexity (e.g. Bluetooth range is 10 meters), b) the average transfer speed and c) the requirement for a pairing bypass without alerting the smartphone user.
3. *WLAN interface* (e.g. Wi-Fi) provides a fast, remote channel that is appropriate for any data volume often without a cost. Due to the general popularity of Wi-Fi, availability is considered high. A shortcoming is that data transfer speed and availability rely on the distance from a base station. This distance, though, is considerably larger than the PAN's requirement, thus, it does not add considerable complexity on evidence collection.
4. *Cellular network (CN) interface* provides a data transport channel of variable speed, which is dependent on the supported carrier network technology (e.g. GPRS, HSDPA, etc). A CN channel is not restrained by the antenna range. It provides greater mobility than any of the aforementioned channels, since the smartphone may travel through cells. Nonetheless, this channel is not considered suitable for large data volume, as: a) it suffers from connection drops, b) the network speed may vary as the user moves inside a cell or visits others, and c) this channel use has considerable cost, and thus it may be discovered by the owner.

Table 67 summarizes each channel's ability to effectively transfer each source data volume (hereto referred as 'volume'). For the association notation three symbols were used, i.e.: 1. (?) transfers small subset of the data source, 2. (~) transfers most data in this data source and 3. (✓) can transfer source type. Obviously, the WLAN channel is able to transfer all data sources. The rest of the associations are listed below:

- *Messaging data* are not volume intense, thus, they can be transferred by all channels.
- *Device data* include data with different volume requirements, ranging from small files to high definition videos. Hence, the GSMMe channel can only convey a subset of this source, whereas PAN and CN can convey, in general, this data source. Nonetheless, the latter are limited by time and transfer cost respectively.
- *(U)SIM Card Data* is not volume demanding, and can be transferred by all channels.
- *Application Data* volume requirements range from few Kbytes up to hundreds of Mbytes. As a result, PAN and CN (provided that available quota exist) are able to transfer this data type. GSMMe can only transfer application data that are not volume intense.

- *Usage History Data* are not always volume demanding (e.g. recent call logs) and, hence, can be transferred by GSMMe. Nonetheless, this data type includes OS logs, which are data volume intense and are transferable by PANs and CNs (if available bandwidth quota exists).
- Similarly to *Usage History Data*, *Sensor* data can be a) either not volume intense (e.g. GPS coordinates) and, thus, transferrable by GSMMe, or b) volume intense (e.g. accelerometer data) that can be transferred by PAN and CN.
- *User Input Data* volume requirements range from a few Bytes, which are transferable even by GSMMe, up to several Mbytes (e.g. keystroke dictionaries) that are transferable by PANs and CNs.

Table 67. Correlation between transport channels and data sources

Transport channel / Data type	GSMMe	PAN	WLAN	CN
Messaging Data	✓	✓	✓	✓
Device data	**?**	~	✓	~
(U)SIM Card Data	✓	✓	✓	✓
Application Data	**?**	✓	✓	~
Usage History Data	**?**	✓	✓	~
Sensor Data	**?**	✓	✓	~
User Input Data	**?**	✓	✓	✓

7.2.2 Digital forensic investigation frameworks

A Digital Forensic Investigation Framework (DFIF) describes procedures that take place before, during, and after the investigation of physical or digital crime scenes. These frameworks refer to methodologies and techniques capable of collecting, analyzing and interpreting digital evidence that is suitable for a forensic investigation. This investigation may be triggered either from internal sources, e.g. in a corporate investigation, or from external ones, e.g. the evidence will be presented to support a hypothesis in courts of law.

This section focuses on the DFIF that have been proposed, and are suitable, for use in cell phone forensics, i.e. digital evidence is collected from a cell phone - either feature phone or smartphone. We organize their presentation, starting from abstract DFIF to more device-specific frameworks, i.e. in: (a) abstract DFIF, (b) DFIF for feature phones, and c) DFIF for smartphones.

7.2.2.1 Abstract DFIF

Abstract investigation frameworks are widely used for the collection and analysis of digital evidence without focusing on the devices' technological 'nuts and bolts'. These frameworks

progressively build upon each other. Their procedures are commonly used as a basis for the construction of subsequent frameworks that are more technologically-specific.

Reith et al. (2002) propose an abstract investigation framework that includes nine phases, namely: identification, preparation, approach strategy, preservation, collection, examination, analysis, presentation and returning evidence. The authors characterize their work, which extends the work from the Digital Forensics Research Working Group (DFRWS, 2001), as abstract since it is independent of technology and crime type.

The investigation framework proposed in (Casey, 2004), includes seven steps, namely: authorization and preparation, identification, documentation, collection and preservation, examination and analysis, reconstruction, and reporting. Documentation is an on-going process that takes place in all of the framework's phases as a basic property of the investigation and is not considered as a standalone phase.

Carrier and Spafford (2003; 2004) propose a framework that is based on the process of investigating physical crime scenes. The framework aids the investigator to develop and test hypotheses, as well as discover insights about events that have occurred in the crime scene. This framework includes five phases: readiness, deployment, physical crime scene investigation, digital crime scene investigation, and presentation. Among them, the first phase is of interest since it includes the necessary steps for quicker incident handling, i.e. investigation unit training, collection and training on the necessary equipment. In the same perspective, reference to forensic readiness can be found in corporate systems that integrate mechanisms for security incident handling (Tan, 2001; Weise and Powell, 2005).

The investigation framework that is proposed by Ciardhuáin (2004) is independent from the organizational/technological context and aims to have an interdisciplinary nature, i.e. to be applicable not only by technically savvy users. In this context, the framework can be applied by first responders, investigators, IT executives, and auditors. The framework describes thirteen steps, namely: awareness, authorization, planning, notification, search and identification of evidence, collection, transport, storage, examination, hypothesis, presentation, proof/defense, and dissemination.

Beebe and Clark (2005) propose an investigation framework, which - in contrary to the above mentioned frameworks - is multi-tiered. The framework includes six high level phases: preparation, incident response, data collection, data analysis, findings presentation, and incident closure. The authors propose evidence triage in which collected data are prioritized according to their importance, hence reducing the volume of data that is send to the lab for examination.

In the same context, Rogers et al. (2006) elaborate on forensic triage in time sensitive investigations. Their investigation framework is tailored for on-site forensics, i.e. the investigation is conducted on the crime scene, without sending any device to a lab for an exhaustive ana-

lysis. It includes the following six phases: planning, triage, usage/user profiles, chronology/-timeline, Internet activity, and case specific evidence.

Contrarily to the frameworks in (Ciardhuáin, 2004; Beebe and Clark, 2005), which are rather detailed, the framework in (Kohn et al., 2006) is more abstract. As a result, it is more adaptive to technology changes. The framework merges the investigation phases of frameworks that have preceded it in only three phases, namely: preparation, investigation, and presentation. The authors highlight that the investigator must document each step taken, as well as are aware of the legal requirements that are in effect in each investigation.

Ieong (2006) aims to provide a framework that is understood from all the participants in a digital investigation, irrespective of their technical skills (i.e. from information technologists to legal practitioners). The framework includes forensic procedures that are based on the Zachman's framework (Zachman, 1987), as well as elaborates on the principles of reconnaissance, reliability, and relevancy. It defines different roles for each forensic investigation participant and compiles a key question list for them, namely: what (i.e. data), why (i.e. motivation), how (i.e. function), who, where, and when.

Abstract investigation frameworks often include in their initial phases guides for not trained personnel. The guides are followed when a field expert is not available. Two noteworthy first responder guides are: a) the UK Association of Chief Police Officers (ACPO) (Wilkinson and Haagman, 2007) and b) the US National Institute of Justice (NIJ) (Mukasey et al., 2008). ACPO describes basic principles of a digital investigation procedure, as well as refers to cases in which different processes may be required, such as pedophile incidents, or collection and preservation of evidence from Personal Digital Assistants (PDAs). On the other hand, the guidelines from NIJ include five steps: (a) gathering investigation tools and equipment, (b) securing and evaluating the scene by checking the device's state and/or by executing preliminary interviews, (c) documenting the scene, (d) evidence collection, and (e) packaging, transportation and storage of potential digital evidence. Both guides include detailed instructions describing the desired behavior of first responders, e.g. what steps must be followed when the device is switched on with an enabled mobile carrier connection.

7.2.2.2 DFIF for feature phones

The framework in (Jansen and Ayers, 2007) focuses on the digital investigation of cellular phones. It describes actions taking place before, during, and after the investigation. Before the start of the investigation, a number of planning actions occur, i.e. the preparation and training on forensic tools, as well as determination of responsibilities of each participant in the investigation. Then, the investigation continues with the following phases: preservation, acquisition,

examination, and analysis. Finally, the investigation findings are reported, which concludes the investigation.

Jansen and Ayers argue that their framework is also applicable in smartphones. However, as their work is now outdated, it cannot reflect the changes that occurred due to the rapid technological and commercial evolution, or address the challenges posed by the heterogeneity of smartphone platforms. For instance, its evidence source taxonomy is a subset of the current smartphone's evidence source taxonomy.

7.2.2.3 DFIF for smartphones

More recent frameworks focus more on the technical details of smartphones and their heterogeneous operating systems. In this context, the framework described in (Ramabhadran, 2009) is tailored for Windows Mobile. It includes twelve investigation phases, namely: preparation, securing the scene, survey and recognition, documenting the scene, communication shielding, volatile evidence collection, non-volatile evidence collection, preservation, examination, analysis, presentation and review. The authors refer to Windows Mobile compatible third-party software and vendor specific (proprietary) software and connectors (cables) that can be used for the evidence collection and examination phases.

Yu et al. (2009) propose a framework for Symbian, which is based on (Carrier and Spafford, 2003; Ramabhadran, 2009). The framework describes the following phases: a) preparation and version identification, b) remote evidence acquisition, i.e. command-response protocols such as AT commands that perform logical data copy, c) internal evidence acquisition, d) analysis, and e) presentation and review. The authors also focus on some details in the Symbian security model that hinder digital investigation.

Dancer et al. (2013) propose a framework that includes five phases, namely: transportation, classification, analysis, interpretation, and retention. Their framework, which is based on an abstract model of smartphone devices, aims to be cross-platform, i.e. applicable to all smartphones regardless of their operating system.

7.2.2.4 Overview

The investigation frameworks that exist in the forensic literature often name similar or even identical procedures differently. Moreover, in some cases a phase in one framework may be included as an activity of another framework's phase. Hence, since a digital investigation framework that fits all purposes can hardly be found in the literature, this complexity regarding

the framework's terminology may create confusion to a new digital investigator and increase the cost of her training. The above frameworks at least include activities concerning: (a) the investigation's preparation (e.g. training on the available tools, reviewing of the case, etc.), (b) collection and preservation (e.g. in a faraday box) of the smartphone, (c) data acquisition, (d) data analysis, and (e) presentation of the investigation results, either to a court of law or in corporate audience. Table 68 summarizes the availability or absence of the above mentioned activities in the investigation frameworks of this section, as well as their categorization into abstract DFIF, DFIF for feature phones and DFIF for smartphones.

Table 68: Overview of phases in the investigation frameworks[†]

Framework	Type	Prp	Col	Prv	Acq	Als	Prs
Casey, 2000	A	✓	✓	✓	✓	✓	✓
Reith et al., 2002	A	✓	✓	✓	✓	✓	✓
Carrier et al., 2004	A	✓	✓	✓	✓	✓	✓
Ciardhuáin, 2004	A	✓	✓	✓	✓	✓	✓
Beebe and Clark, 2005	A	✓	✓	✓	✓	✓	✓
Kohn et al., 2006	A	✓	✓	✓	✓	✓	✓
Rogers et al., 2006	A	✓	✓	✓	✓	✓	✓
Ieong, 2006	A	✓	×	×	×	×	✓
Wilkinson and Haagman, 2007	A	✓	✓	✓	×	×	×
Mukasey et al., 2008	A	✓	✓	✓	×	×	×
Jansen and Ayers, 2007	F	✓	✓	✓	✓	✓	✓
Ramabhadran, 2009	S	✓	✓	✓	✓	✓	✓
Yu et al., 2009	S	✓	✓	✓	✓	✓	✓
Dancer et al., 2013	S	✓	✓	✓	✓	✓	✓

[†] **Prp** (Preparation), **Col** (Collection), **Prv** (Preservation), **Acq**(Acquisition), **Als** (Analysis), **Prs** (Presentation), **Type{A**(Abstract), **F**(Feature Phone), **S**(Smarphtone)**}.**

The majority of the investigation frameworks are tailored for 'post-mortem' forensics, i.e. the evidence is collected after a crime has taken place or an incident has occurred. Also, the analysis takes place after the collected devices have reached a forensic lab and/or a forensically sound copy has been created, which may take hours or even days after the incident/crime. This makes these investigation frameworks rather cumbersome, comparing to forensic triage techniques (Rogers et al., 2006; Walls et al., 2011). Also, this rationale of post-mortem analysis has influenced the available data acquisition methods, omitting the collection of time-sen-

sitive and/or volatile data, such as sensor data. Nonetheless, as discussed in the next section, the ad-hoc collection of sensor data from smartphones provides context awareness during a digital investigation, which may aid the prevention of an incident/crime in proactive forensics.

7.2.3 Sensor data in digital forensics

The majority of smartphones, as well as feature phones, include multimedia hardware such as a microphone and camera (often a pair of front and back cameras). Moreover, other hardware is getting more ubiquitous in smartphones, such as: (a) location sensor (i.e. Global Positioning System (GPS)), (b) motion sensors (i.e. accelerometer, gyroscope), and (c) environmental sensors (e.g. magnetometer, proximity, light, temperature, pressure). Spyware authors have already used smartphone sensors in order to acquire context awareness either for research reasons (e.g. in (Mylonas, 2008; Niemelä, 2008; Xu et al., 2009; Schlegel et al., 2011)), or for commercial ones, for instance spyware software used for family affairs (e.g. FlexiSPY[38]). In addition, sensors provide a data source that can be collected in relation to crime investigations in order to provide context awareness. This context awareness can be used, in accordance with the existing legal framework, in a proactive forensic investigation to construct a suspect's activities, which aid the creation of a hypothesis, or rejection or acceptance of an alibi.

Nonetheless, the majority of sensor data cannot be collected during a post-mortem investigation. This stems from their inherent characteristics; sensor data are volatile and are time-sensitive. Thus, time stamped evidence derived from sensor data can hardly be found on the device after a crime's occurrence, unless they have been explicitly collected. Currently, only GPS data are collectable in post-mortem forensics. They are treated as metadata, which are either embedded in other files (e.g. geotagging), or cached by the operating system (Allan and Warden) or by other applications (e.g. Google maps, social media apps).

One may argue that investigators can infer context awareness by collecting data from a service/network provider, without accessing the suspect's device. For instance, device location can be estimated from collected data from the carrier's network via triangulation and device speed can be estimated from the distance between the cell ids that the device is visiting. However, on the one hand the accuracy of this estimation cannot be compared with the data that are generated from the device's dedicated hardware, which, in general, gives more accurate measurements. On the other hand, sensors provide unique measurements from: (a) a single hardware and (b) combinations of measurements deriving from two or more sensor hardware - these combinations are commonly referred to as virtual\software sensors. For instance, for

[38] FlexiSPY, http://www.flexispy.com/ (retrieved November 2012).

the former, the light sensor provides data that cannot be found from another source. For the latter, to the best of our knowledge, the only available method to collect the device's orientation is from dedicated hardware, e.g. the accelerometer and the magnetometer.

In a digital investigation sensor hardware can provide direct evidence, such as location evidence from a GPS. They also provide indirect evidence, i.e. evidence that is inferred from the acquired data values. For instance, an investigator may use the light sensor to infer whether a suspect is indoors or outdoors, or to infer keystrokes from a nearby keyboard via the smartphone accelerometer (Marquardt et al., 2011). Some sensor data are easily comprehensible (i.e. GPS coordinates, values from environmental and multimedia sensors), whereas the values from motion sensors are not. For the latter, existing sensing literature has elaborated on the classification of data from motion sensors with the aim to perform activity recognition (e.g. in (Choudhury et al., 2008; Liu et al., 2009; Lane et al., 2010; Kwapisz et al., 2011; Bujari et al., 2012)). While this discussion about sensor data analysis is not exhaustive, we consider that a forensic investigator can benefit from the existing literature if there is a need to infer gestures (Kern et al., 2003; Liu et al., 2009; Choe et al., 2010) as well as other activities, such as walking, running, etc. (Bao and Intille, 2004; Ravi et al., 2005; Lester et al., 2006; Long et al., 2009; Khan et al., 2010; Kwapisz et al., 2011; Bujari et al., 2012).

Finally, the combination of sensor data provides stronger indications about the smartphone's context. For instance, combined data from the GPS, accelerometer and magnetometer provide a smartphone's navigation (e.g. device orientation, speed, etc.). In addition, such data combinations often improve the collection effectiveness, e.g. the light sensor may be used as a condition of whether a camera caption will take place or not.

The next subsection introduces a scheme that is tailored for proactive forensics and aims to avoid its misuse from law enforcement or malicious individuals.

7.3 Proactive Smartphone Forensics Investigation

This section focuses on proactive forensics, where ad-hoc acquisition of smartphone evidence takes place. As discussed beforehand, proactive smartphone forensics investigation may take place for the investigation of crimes considered 'severe' by the legal and regulatory context in place (e.g. crimes against public or the state, pedophilia, etc.). In such cases, the creation and defence of an investigation hypothesis[39] may use the aforementioned associations of smartphone sources with transport channels and evidence types. In the sequel, an investigation scheme is presented and outlined.

[39] Hypothesis is a report based on the examination and the analysis of collected evidence that are admissible to court.

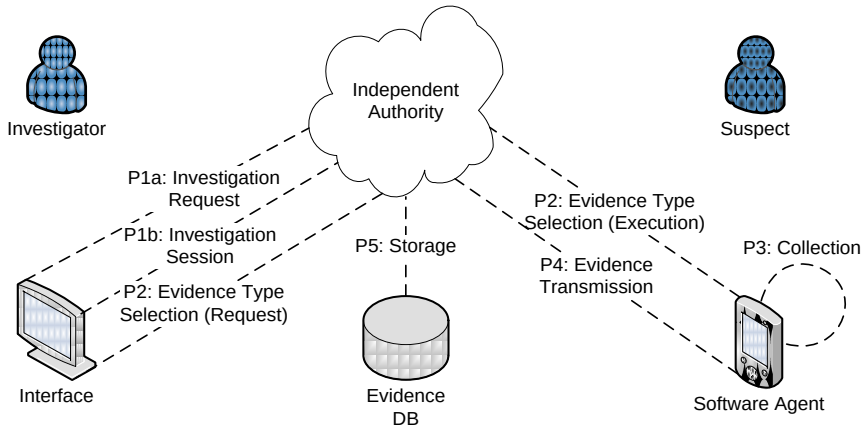


Figure 15: Proactive Smartphone Forensics Scheme

7.3.1 Proactive Smartphone Forensics Scheme

The proposed scheme consists of three entities (Figure 15), namely: a) subject(s) carrying out a proactive smartphone forensic investigation (hereinafter 'investigator'), b) an Independent Authority (IA), and c) the investigation's subject(s) (hereinafter 'suspect').

As depicted in Figure 15, the *IA* is the scheme's corner stone, as it: a) controls evidence collection from the Software Agent (SA) that resides in the suspect's smartphone, b) handles evidence storage for a time period compliant with existing laws and regulations, and c) authorizes investigator's requests for a proactive forensic investigation against individuals.

This architecture was selected so as to hinder investigators, or other individuals, from misusing the evidence collection mechanism for profiling and intelligence gathering reasons (see §XX). Hence, it is assumed that the IA allows a proactive digital forensic investigation to take place only in suspected 'severe' crimes. It is also assumed that the IA maintains a database, where evidence data are stored and protected, in terms of forensic soundness and confidentiality.

The *investigator's* role in this scheme is to create a hypothesis, i.e. collect adequate evidence suitable for use in courts of law. An investigator may request from the IA authorization for a proactive smartphone forensics investigation to take place when:

a) other mechanisms for data collection are either incapable to gather the required data (e.g. cases where the suspect's context is required), or they collect data of reduced accuracy, inadequate for evidence presentation in a court of law (e.g. the location accuracy of the GPS sensor is greater than the approximate location provided by the cell phone provider),
b) the suspected crime is considered 'severe' by laws and regulations

In this scheme, it is assumed that a Software Agent (SA), controlled by the IA, is present in the *suspect's* smartphone. Also, the IA has the capability to commence the collection of evi-

dence types from selected smartphone sources. The scheme's architecture is depicted in Figure 15, while its processes are described in the sequel.

7.3.2 Scheme Processes

The proposed scheme consists of six building blocks - processes, namely: (1) investigation engagement, (2) evidence type selection, (3) evidence collection, (4) evidence transmission, (5) evidence storage and (6) investigation completion (see Figure 16).

The first process takes place once per suspect investigation, while the remaining processes are iterative and incremental. The last process (*Process 6: Investigation Completion*) is completed when the investigation's requirements are satisfied, i.e. the hypothesis can be created and its validity can be defended in the courts of law, or when the IA's granted permission becomes invalid. The other processes are described in detail in the sequel.

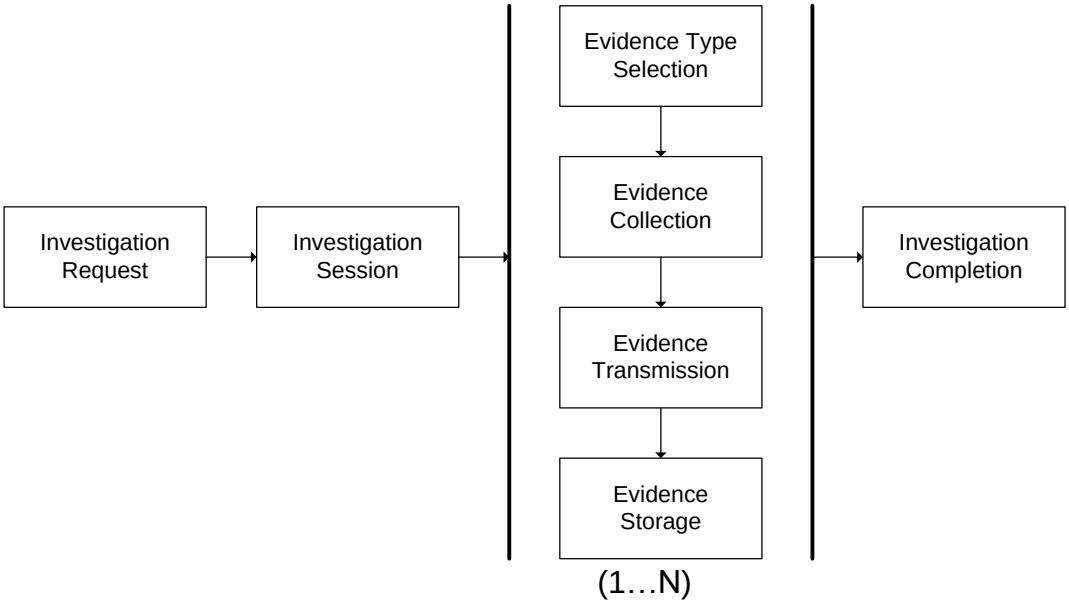


Figure 16: Processes of the Investigation Scheme

7.3.2.1 Process 1: Investigation Engagement

It consists of two sub processes that define an investigation's details:

- *Process 1.1: Investigation Request (IR).* This is the formal request of investigation authorization, addressed to the IA. The request contains a definition of investigation specific parameters, such as: (a). the investigator who creates the hypothesis, (b). the suspect, (c). the nature of the examined crime, (d). the expected investigation period, (e). the required data sources that will be collected as potential evidence, and (f). the required channels for evidence transmission.
- *Process 1.2: Investigation Session.* Once an Investigation Request is submitted to the IA, a non-automated process determines the investigation's permission level upon the suspect's

smartphone. This permission level is determined by the IR details and the evaluation criteria of each IA, which are dependent on the regulatory context. If the IR is accepted, an investigation session is provided to the investigator that is used for the following processes.

7.3.2.2 Process 2: Evidence Type Selection

This process is initiated after the investigation engagement process. It refers to the selection of evidence types and the corresponding transport channels by the investigator. This selection is carried out based on the two associations presented previously: (a). between smartphone data and digital evidences, and (b). between smartphone data and transport channels. For instance, an investigator may compile a configuration request for the SA evidence collection process. This configuration request specifies the desired data sources (e.g. motion sensors data) and transport channels (e.g. WLAN), and it is forwarded to the IA. Then, the IA executes the request by acquiring evidence from the SA, only if the configuration request parameters are conformant with the current investigation session permission level. In this way, misuse of evidence collection is avoided. Finally, this activity takes place every time the investigator makes a new request for potential evidence in the context of the same investigation session.

7.3.2.3 Process 3: Evidence Collection

It is triggered every time the SA configuration is altered. Based on configuration attributes (i.e. data source, transfer channel, interception period and duration, etc.) the SA harvests potential evidence, applies integrity mechanisms and forwards them to the evidence transmission process.

7.3.2.4 Process 4: Evidence Transmission

The transmission of evidence takes place, when the collection process ends. It is assumed that an Evidence Transmission Protocol (ETP) is applied between IA and SA. This ETP must include messages for the collection of all smartphone data sources and support the various smartphone transport channels. Furthermore, the ETP must impose security properties, such as message authenticity, integrity, liveness, and confidentiality.

7.3.2.5 Process 5: Evidence Storage

It refers to the storage of potential evidence that is received from the SA in the IA's infrastructure. The preservation of potential evidence in the forensics database must ensure their forensic soundness via employing integrity mechanisms, as well as, their confidentiality. The stored evidence is bound to be revisited during an investigation, so as to be further examined and analyzed before being presented in court. Thus, limited access (e.g. read-only) is provided to the investigator via an interface.

7.4 Themis

In this section we discuss the implementation details of Themis[40], a system for remote and ad-hoc data acquisition from smartphones. Our implementation focuses only on sensor data, since, as it was discussed in §X.2, they provide context awareness and are volatile. Due to their volatile nature, the only way to acquire them, in case such a need arises during a digital investigation, is by software that collects them when they are generated by the suspect's actions.

7.4.1 Themis' architecture

Themis implements key components (i.e. the collection agent and transfer protocol) from the proactive smartphone investigation scheme that is described in the previous subsection. In this scheme an investigator has ad-hoc access to smartphone data, to combat crime that in effect legal framework considers as "serious crimes", in the same manner as lawful interception is used today for combating them. We regard that the deployment of Themis in serious, as well as time-sensitive crimes (i.e. the investigation is sensitive to time constrains (Rogers et al., 2006)) increases the effectiveness of law enforcement by aiding the investigators to assess the context more accurately and to decide the necessary actions. In such cases (e.g. pedophilia, etc.), we consider that crime prevention has a much more positive impact on the protection of a citizen than a thorough analysis of a collected device after the crime has taken place.

When such a technology is used, specific legal and institutional guarantees and tools must be provided, so as to ensure its lawful use and prevent its abuse. In this context, we proposed in the previous subsection that the scheme's cornerstone would be a specialized Authority, which is: (a). trusted to comply with the legal framework, and (b). independent with regard to the police investigators (see Figure 17). This trusted Authority, referred hereinafter as a Law

[40] According to the Greek mythology, Themis was the Goddess of Justice, i.e. the personification of divine or natural law, order, and justice.

Enforcement Entity (LEE), hinders investigators and/or malicious individuals from misusing its data collection mechanism. Furthermore, LEE collects, controls, and stores the acquired data in accordance with the existing laws and regulations.

In Themis, data collection is controlled by a C&C server (hereinafter 'workstation'), which is managed in the LEE premises. The workstation stores and protects the acquired data. Data are collected by the Themis' agent (hereinafter referred to as 'agent') that resides in the suspect's smartphone. The data are transferred through a wireless secure channel that is set up between the agent and the workstation. Investigators have only read access to the acquired data via investigation sessions managed by the LEE (c.f. §XX).

Our implementation focuses on the components that allow the ad-hoc acquisition of smartphone evidence derived from the device's sensor hardware. The subsequent subsections include Themis' security and implementation details.

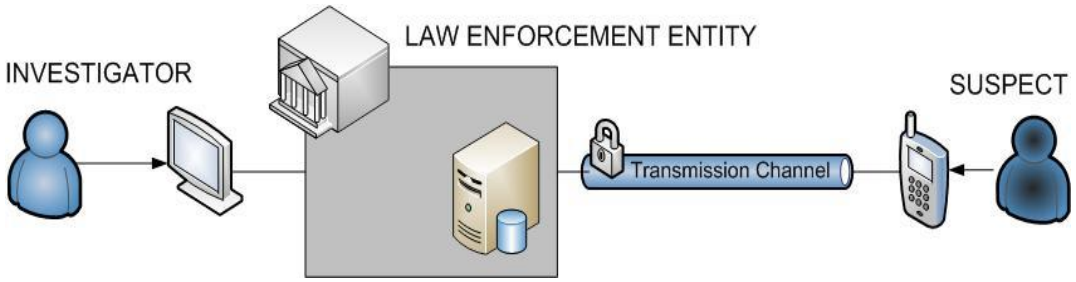


Figure 17: Overview of Themis

7.4.2 Implementation considerations

Themis uses a wireless channel for transferring collected data from the device to the workstation. To ensure the forensic soundness of transmitted data via a wireless channel, a custom protocol is required that includes mechanisms to provide:

(a) **Confidentiality**. The transmitted data must be encrypted so as to avoid their unauthorized disclosure (i.e. eavesdropping).

(b) **Integrity.** The transmitted data must be protected against intentional or unintentional unauthorized alteration/modification.

(c) **Data freshness**. The protocol must include appropriate mechanisms to avoid message replay attacks.

(d) **Data origin authentication**. The receiver must be able to verify that the collected data originate from the expected sender (i.e. protection from data fabrication).

(e) **Interruption avoidance.** The receiver must not be hindered from receiving data from the agent.

Moreover, the protocol must be cross-platform, without restricting the use of any technology, either in the workstation or in the agent's implementation. The above mentioned properties (a)-(d) of a secure wireless channel ensure the authenticity of the collected evidence (Karyda and Mitrou, 2007). On the other hand, the agent - as any data acquisition tool - must adhere to the four ACPO principles (Wilkinson and Haagman, 2007), which are summarized in (Jansen and Ayers, 2007) as follows:

1. No actions performed by investigators should change data contained on digital devices or storage media that may subsequently be relied upon in court.
2. Individuals accessing original data must be competent to do so and have the ability to explain their actions.
3. An audit trail or other record of applied processes, suitable for replication of the results by an independent third-party, must be created and preserved, to accurately document each investigative step.
4. The person in-charge of the investigation has the overall responsibility to ensure that the above-mentioned procedures are followed in compliance with laws in force.

The first principle refers to ensuring the integrity of the device and its stored data, as well as the integrity of extracted data. It is the basis of traditional digital forensics, but in current forensics practices it progressively fades (e.g. 'live' forensics, forensic triage). This is because in many occasions it is infeasible to extract the storage component of the digital device (e.g. smartphone), without losing data or causing damage to it. This principle is also infeasible for sensor data, as they are volatile. Hence, the only way to acquire them is by using software on the device. Nonetheless, the agent's installation must cause minimal alterations to the device, as well as to the sensor data sources. For the former, in the majority of smartphone platforms applications are sandboxed. For the latter, access to sensor data is performed via the platforms' API without impairing its sources' integrity. Finally, regarding the integrity of extracted data the use of a secure channel in Themis, meeting the above requirements, ensures the integrity of the extracted data.

The last three principles ensure the accountability of the digital investigation. In Themis, accountability is ensured by the LEE, which manages and controls access to smartphone data according to our proposed scheme (c.f. §XX). Investigators can query smartphone data only via the LEE's infrastructure and according to the guarantees in force in the country, if and only if the *investigation request* can justify the need for an *investigation session*. The LEE stores the acquired data as well as investigation session logs for a period that is compliant with the existing laws and regulations. This documentation of investigation sessions from a trusted LEE enables a forensic analyst to prove that acquired data have not been devised or

tampered with. Moreover, as sensor data can be deemed biometric (see §XX), the LEE must enable protection mechanisms to satisfy legal requirements or standards.

Finally, this work focuses on sensor data acquisition and not on trust and functional issues regarding the LEE, albeit, we note that in a real deployment of Themis a pair of LEE may be used, in order to satisfy the principle of separation of duties and to ensure system availability.

7.4.3 Implementation platform

The Themis' agent was implemented in the Android platform. Android was selected for the reasons that were presented in §XX, namely: (a). *popularity*, being the smartphone platform with the greatest user space, (b). *portability*, i.e. compatibility with different smartphone hardware, (c). *open source*, which allows researchers to *study its* security model and *extend* it, (d). *flexibility*, i.e. allows the installation of apps which are available outside of the Google Play, thus the agent can be installed without the need to jailbreak the device, and (e) *development support* with the existence of free and publicly available SDK and emulators. In addition, Android applications are sandboxed, i.e. their installation does not alter other applications or their files, which is a desirable property in digital investigations, since the installation of the agent causes minimum changes to the device's state.

The agent's implementation is based on Java, customized for the Android platform. Thus, our implementation may be used as a basis for the creation of agents in smartphone platforms with similar development languages - conforming to the details of the security models of each platform. In (Mylonas et al., 2011a, 2012a) the user study revealed that the effort to develop a subset of the agent's functionality (GPS collection and transfer) was comparable in smartphone platforms that had similar a) security model rationale and b) syntax in the supported programming language, i.e. Android, BlackBerry, and Windows Mobile/Phone. Moreover, our agent is compatible with other Android devices, such as tablets.

7.4.4 Platform's security considerations

This subsection summarizes our efforts to circumvent the protection mechanisms that are currently employed by Android security model to collect sensor data from the device.

7.4.4.1 Circumventing permissions

Android's security model is permission-driven, i.e., apps are sandboxed and can only access a resource (e.g. data, app components) if they are granted the permission that is protecting it (Google). Specifically, Android users are delegated to grant access to resources that are protected with the dangerous protection level. Before an app's installation the user is pre-

sented with a graphical interface, which lists its requested permissions. If the user is convinced to accept them and install the app, she cannot latter withdraw any granted permissions at runtime. Nor will the app prompt the user again for access to these resources. The user may only uninstall the app from the device to stop it from accessing these resources. Moreover, Android uses an all-or-nothing approach concerning permission authorization. In this context, the user does not have the choice to selectively grant a subset of the permissions that an app requests. Hence, she must either accept all permission requests or skip the app's installation.

Nonetheless, by examining Android's security model and its API for smartphone sensors during Themis' implementation, we found out that access to the majority of smartphone sensors is unprotected[41] (see Table 69). This holds true as they are not associated with any permission - not even one at normal protection level (Google). This is orthogonal to the documentation of Android's security model, where access to every resource must be protected with permission (Google). In this context, this vulnerability allows any Android application to access the majority of the sensors without prompting the user for authorization. Therefore, the only effective method for a user to discover whether an app accesses sensor data is by static or dynamic app analysis.

As depicted in Table 69, we found that only three sensors are protected with permission. Specifically: (a) camera is protected with the permission *Android.permission.CAMERA*, (b) microphone is protected with *Android.permission.RECORD_AUDIO*, and (c) GPS is protected with the permission `Android.permission.ACCESS_FINE_LOCATION`. The above are *dangerous* permissions, thus the user must authorize them during app installation. Nonetheless, we expect that average smartphone owners are more likely to accept them. This holds true since, as discussed in chapter X, in practice smartphone users tend to ignore Android permissions during application installation and are unable to comprehend Android permissions, as well as the risks that are related with them (Felt et al., 2012b; Kelley et al., 2012). Moreover, users in any platform are trained to click through interruptive and repetitive messages that appear while they are completing a task (Motiee et al., 2010). Android permissions appear in every app installation and interrupt users from installing and using a new app. This partially explains why they are often ignored.

Smartphone users who elaborate on Android permissions are likely to reject permission according to their perception about its impact on their security and privacy, as well as its relevance with the app's functionality. For instance, a user is expected to grant access to a permission that allows pairing to Bluetooth devices (`Android.permission.BLUETOOTH`) more easily than to one that changes her contact list (`Adroid.permission.WRI-`

[41] We have verified the existence of this vulnerability even in the latest releases of Android (i.e. v. 4.1.2).

TE_CONTACTS). In addition, a voice memo application is expected to be granted the request to access the microphone, whereas a game is not.

Table 69: Sensor data collection details in Android

#	Android Sensor	Access	Rejection likelihood	Spoofing	Unstable
1	Accelerometer	unprotected	N/A	N/A	N/A
2	Camera	user granted	High	X	X
3	GPS	user granted	Low	X	N/A
4	Gravity	unprotected	N/A	N/A	N/A
5	Gyroscope	unprotected	N/A	N/A	N/A
6	Humidity	unprotected	N/A	N/A	N/A
7	Light	unprotected	N/A	N/A	N/A
8	Linear Acceleration	unprotected	N/A	N/A	N/A
9	Magnetometer	unprotected	N/A	N/A	N/A
10	Microphone	user granted	High	N/A	N/A
11	Orientation	unprotected	N/A	N/A	N/A
12	Pressure	unprotected	N/A	N/A	N/A
13	Proximity	unprotected	N/A	N/A	N/A
14	Rotation Vector	unprotected	N/A	N/A	N/A
15	Temperature	unprotected	N/A	N/A	N/A

Table 69 includes a qualitative metric regarding the likelihood that the suspect rejects a permission protecting access to the device sensors and, due to Android's all-or-nothing approach, to cancel its installation. This metric is based on a user study conducted by Felt et al. (2012a), regarding the perceived impact of smartphone risks deriving from smartphone permissions. Smartphone users regard the camera and microphone sensors intrusive and thus we regard that their rejection likelihood is high. On the contrary, users feel more comfortable to share their location - and this may stem from the popularity of location based services and, as a result, we assess its rejection likelihood as low.

A digital investigator may circumvent the protection offered by *dangerous* permissions and hence lower the possibility of app installation rejection by Android users who elaborate on permissions. This can be achieved in Android by deploying the agent as a pair of colluding apps which share their permissions and their intended functionality matches their permission requests. Android applications may share their permissions in an either intentional (i.e. colluding apps that break the permission system) or unintentional (i.e. confused deputy attack) manner (Dietz et al., 2011; Felt et al., 2011; Schlegel et al., 2011; Grace et al., 2011). In this context, the investigator must convince the suspect or criminal to install either two individual apps, or one app that, in turn, convinces the user to install the second app (e.g. with a pop-up ad).

Finally, the agent receives commands from the workstation either from the Internet or via SMS messages. In the first case, the `Android.permission.INTERNET` is required,

which has one of the lowest rejection likelihood among the permissions. This holds true as it is the most commonly requested permission for free benign applications (Zhou and Jiang; 2012). On the other hand, receiving SMS messages requires the permission `Android.permission.RECEIVE_SMS`, which has considerable rejection likelihood[42].

7.4.4.2 Circumventing visual notifications

Amongst the protected sensors, the camera and GPS have additional level of protections to avoid their misuse. Both of them have visual notifications to alert a user that data are being collected, i.e. camera preview window and flash light, and GPS notification icon. Moreover, the camera has also a sound notification (i.e. shutter), which in some countries must be enabled even when the device is muted.

In our experiments we were able to circumvent the camera preview, as well as the shutter by abusing the API of the camera. We changed the properties of the preview window (i.e. Android Activity), e.g. set its width and height to zero and set null parameters in the relevant API to disable the shutter. It should be noted that these changes are only in effect when the agent is using the camera and other apps that use the camera are not affected. However, in our tests we found that the device drivers for the camera are implemented differently by each device's manufacturer. Therefore, we had to abuse the API differently in each of the test devices. This makes the API misuse rather unstable and, hence, it must be tested in advance by the forensic investigator. Otherwise, the agent is more likely to fail. Finally, the flash light can be easily disabled by an API parameter, without misusing its API parameters.

To the best of our knowledge there is no effective method to hide the GPS notification, which is presented in the screen's notification area, when: (a). an app is running in the background (i.e. it does not consume any space in the smartphone screen), (b). the device owner is using the device, and (c). and the app requests access to the GPS. Nonetheless, we consider that average users will not notice this notification icon when the agent is collecting location data, as: (a). users are accustomed to using location based services and thus see the notification icon often, (b). the size of notification icon is rather small and as a result easy to ignore, and (c). the device owner may be simultaneously using another location based application (e.g. for navigation) and thus the presence of the icon may not seem suspicious. In addition, in Android an app can access the last known user's location, thus the investigator can access semi-fresh location data, without the GPS symbol being shown to the user. The investigator can combine these location data with approximate location data from the mobile network provider and try to infer the suspect's location.

[42] The user study in (Felt, et al., 2012b) did not include the impact of malicious handling of incoming SMS messages.

Finally, by examining Android's API we found out that Android's security model does not provide any visual distinction between error messages that are being created by the operating system and the ones created by third-party apps. Therefore, a malicious app may exploit this vulnerability to create error messages that look exactly similar to system warnings. These spoofing messages are often used by malware to mislead smartphone users (Mylonas, 2008).

7.4.5 Agent's installation

In specific circumstances regarding the ad-hoc collection of smartphone data - only and solely for law enforcement purposes and in accordance with the existing laws and regulations - the investigation team may use software that pre-exists in the device (Rose, 2012). For instance, the device provider may cooperate with law enforcement authorities by including such software in a custom ROM, this software may be part of incidence response and thus be included in the security policy of a critical infrastructure. However, in general we expect that investigators who use Themis will come across to the problem of software installation. In this subsection, we discuss three methods for the Themis' agent installation in an Android smartphone, namely: physical access (i.e. manual installation), social engineering, and vulnerability exploitation.

7.4.5.1 Physical access

Manual installation of an app in Android is not time-consuming. Hence, if the device is left unattended, then the agent's installation depends on whether the device is locked or not. Password protection can be circumvented by: a) entering the device in debugging mode via USB (Hoog, 2011), b) guessing the password, especially if a short one is used, and c) inferring a graphical password from smudges on the device's screen (Aviv et al., 2010).

In chapter X , we found that 64% of our survey participants did not password protect their device. In this context, due to the portability and small size of smartphones, the likelihood of a temporary unauthorized access cannot be neglected.

Finally, manual installation may be used, apart from the scenarios with a temporary access to the smartphone (e.g. family affairs, corporate espionage, insider threat), in corporate environments for incidence response purposes. In this scenario, administrators install the agent in an employee's smartphone to enforce a security policy and/or to ensure forensic readiness.

7.4.5.2 Social engineering

During an installation with social engineering, the agent is installed by the device's owner**.** The owner is lured into downloading the agent, which is masquerading as an app of her interest. This app can either be a repackaged version of an existing Android app, or a new app.

Prior to app installation, the device owner must be convinced to accept any access request the app makes**.** As discussed previously, due to Android's all-or-nothing approach the user does not have the choice to selectively grant a subset of the permissions that an app requests. Hence, she must either accept all permissions or skip the app's installation. Nonetheless, as discussed in chapter X, smartphone users tend to ignore Android permissions during application installation, as well as the analysis in the previous subsection revealed that the majority of smartphone sensors are unprotected.

The agent's installation via social engineering requires prior intelligence for the suspect or criminal. For instance, in the case that the investigation team has leads to a 'severe' crime, the investigators may use relevant social engineering context in order to lure him to install the agent. Also, colluding apps may be used in order to circumvent security savvy users who scrutinize permissions. Finally, this installation option is not appropriate when time-sensitive and ad-hoc access is needed without the device owner's participation in the agent's installation. In this case, an intrusive installation vector is required, i.e. vulnerability exploitation.

Table 70: Exploits used by malware in the wild (Zhou and Jiang, 2012)

Exploit	Vulnerable Software	Exploit details
Asroot	Kernel	http://milw0rm.com/sploits/Android-root-20090816.tar.gz
Exploid	init (Android <= 2.2)	http://c-skills.blogspot.com/2010/07/Android-trickery.html
Rage against the cage	adbd (Android <= 2.2.1) and zygote (Android <= 2.2.1)	http://c-skills.blogspot.com/2010/08/droid2.html
Zimperlich	adbd (Android <= 2.2.1) and zygote (version <= 2.2.1)	http://c-skills.blogspot.com/2011/02/zimperlich-sources.html
KillingInThe-NameOf	ashmem (Android <=2.2.1)	http://c-skills.blogspot.com/2011/01/adb-trickery-again.html
GingerBreak	vold (Android <=2.3.3)	http://c-skills.blogspot.com/2011/04/yummy-yummy-gingerbreak.html
zergRush	libsysutils (Android <=2.3.6)	http://forum.xda-developers.com/showthread.php?t=1296916

7.4.5.3 Vulnerability exploitation

This installation method involves the exploitation of a vulnerability, which leads to elevation of privileges. If the exploitation is successful, then the investigator has privileged access

(i.e. with root privileges) in the smartphone. Therefore, she is able to install the agent in the device. The vulnerability may be present either in the operating system, or in an unpatched or misconfigured application (e.g. Web browser (c.f. §4)). Furthermore, the rooting process that users follow to unlock their devices often create vulnerabilities that are easily exploitable (Cluley).

Known vulnerabilities and exploits are often publicly available in databases such as Open Source Vulnerability Database[43] (OSDVB) and Exploit Database[44]. Table 70 summarizes the exploits that are currently being used by Android malware in the wild, according to the analysis conducted by Zhou and Jiang (2012).

Similarly to penetration testing, the investigation team will have to perform information gathering to select an exploit that matches a known vulnerability of the suspect's device (McClure et al., 2005). Otherwise, incorrect use of exploits, or use of exploits that are not thoroughly tested for their effectiveness, may reset the device or even brick it (i.e. render it useless).

This installation option is suitable in cases, where time-sensitive, ad-hoc acquisition of sensor data is required. It assumes that the investigation team has the expertise and the resources in order to complete it. Also, it assumes that information gathering has been performed and the investigation team has prepared or tested the appropriate exploits, so as to cause minimal changes to the device.

7.4.6 Agent's functionality

The agent's functionality is depicted in Figure 18. It includes agent's activation, agent's configuration and data acquisition.

After its installation, the *agent is activated* when one of the following conditions is met:

a) When the operating system finishes its boot procedure. This enables the agent to execute after the device restarts.
b) Each time the user interacts with the device, i.e. unlocks the device. This enables the collection of data that require the presence of device's owner (e.g. voice, camera).
c) Each time the device reconnects to the Internet, irrespective of the network channel (i.e. mobile carrier's network or Wi-Fi). This enables the agent to contact the forensic workstation in order to acquire a new acquisition configuration file and/or to transfer to it any pending data, i.e. collected data that have been stored locally.
d) When an SMS command arrives (Mylonas, 2008). This enables the agent to receive an acquisition configuration file when the device cannot connect to the Internet.

[43] http://osvdb.org/
[44] http://www.exploit-db.com/

After the first three events the agent attempts to contact the forensic workstation and receive an *acquisition configuration* file. Then, the agent decrypts and parses the acquisition configuration file and starts the data acquisition. In data acquisition, Android's sensor API is used and, thus, the collected evidence is reliable (Karyda and Mitrou, 2007) - assuming the integrity of the reported data from the sensors. The collected data are compressed, hashed and encrypted. If the agent is unable to connect to the forensic workstation, then the acquired data are stored in a local database. Otherwise, they are transferred to the forensic workstation.

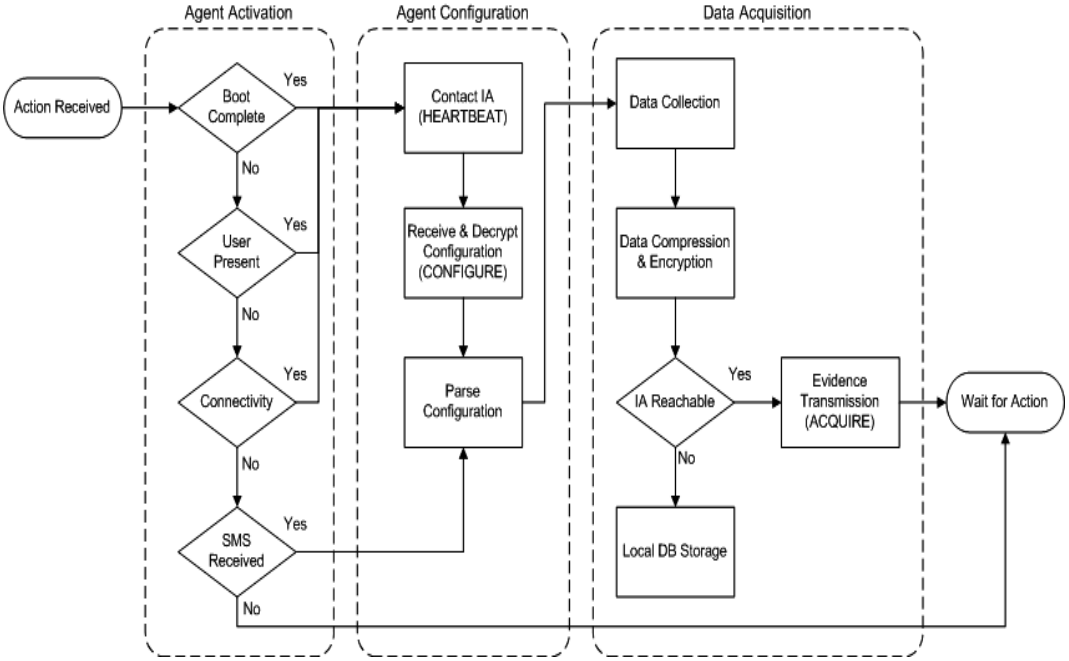


Figure 18: Agent's functionality

7.4.7 Evidence transfer protocol

We implemented a custom protocol for the transfer of sensor data from the agent to the forensic workstation, which we refer to as Evidence Transfer Protocol (ETP). ETP is a Command and Control (C&C) protocol, where the workstation sets the acquisition options in an acquisition configuration file and the agent gathers data from the smartphone sensors according to these options. Then, the data are transferred to the workstation via a secure channel.

As depicted in Figure 19, ETP includes three messages (Table 71):

(a) *Heartbeat*. The agent registers to the workstation its details with this message, i.e. current IP address and its id (i.e. Universally Unique Identifier - UUID). The first Heartbeat message, i.e. the one after the agent's installation, includes the availability and details of sensor hardware in the device.

(b) *Configure*. This message includes the acquisition options, i.e. multiple entries that specify selection of sensor data sources (e.g. GPS, accelerometer, etc.) and, optionally, collection options (i.e. capture duration, sample rate, and capture intervals).

(c) *Acquire*. This message includes the collected data to the forensic workstation. It consists of multiple entries that contain raw data and their metadata (e.g. identifier of the sensor source, collection resolution and timestamp).

Our implementation aims to be simple (i.e. with low implementation complexity in order to be used and extended for academic purposes), as well as cross-platform. For the previous reasons, the protocol's implementation is based on Hypertext Transfer Protocol (HTTP) for message exchange. However, the use of HTTP messages is not restrictive, thus any C&C protocol can be used, e.g. Internet Relay Chat (IRC). Moreover, we did not delve into any technologies that conceal data transfer, e.g. covert channels. We expect that such technologies will be used in a real deployment of Themis.

In our protocol implementation, the agent registers to the workstation with a HTTP POST message and then the workstation replies with an acquisition configuration file. Finally, the agent transfers data with a HTTP POST message. The messages' syntax adheres to JSON[45], a text format that is language independent, but easy to parse in any programming language. Table 71 summarizes the JSON attributes that ETP uses.

The implementation includes commonly used mechanisms that provide a secure channel, i.e.: all messages are encrypted (AES-256) and hashed (SHA-1) in a hash-then-encrypt manner. This provides data confidentiality and integrity, as well as data origin authentication. All messages that are exchanged are time-stamped, which provides data freshness. Moreover, every collected sensor data is time-stamped, which allows investigators to create a timeline of the events that happened.

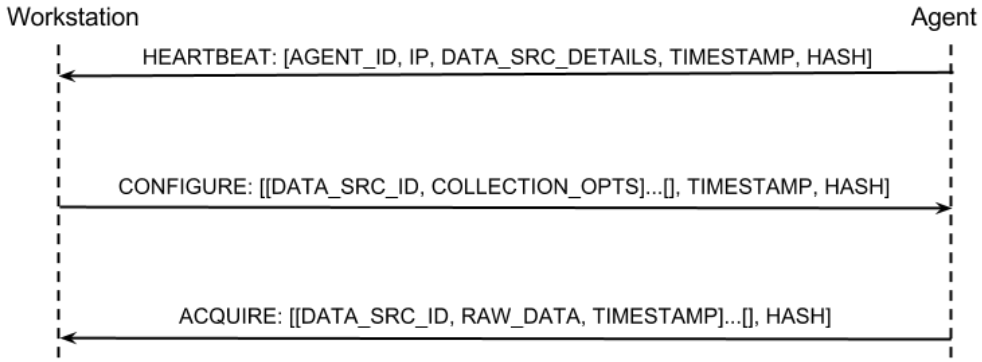


Figure 19: Overview of the evidence transfer protocol

In a real deployment of Themis we expect that the forensic workstation would incorporate mechanisms that: a) avoid message interruption (i.e. denial of service attacks), such as load balancing, server farms, etc., and b) provide key management and key exchange. These mechanisms are out of the scope of this work.

[45] http://www.json.org/

7.5 Experimental results

In this section we summarize our experience regarding data collection of smartphone sensors by Themis, as well as describe scenarios regarding its use and preparation.

7.5.1 Test environment

Before experimentation, we set-up a forensics test lab (Figure 20). The workstation was implemented with a server running Linux (kernel 2.6.32-38), PHP (v. 5.3.8), and Apache server (v. 2.2.22). Initially, we installed and debugged the agent's implementation in the emulator provided by Google. Then, we installed and tested our implementation in two Android smartphone devices: a) a LG Optimus L3 (LG-E400) with Android v. 2.3.6 and b) a Samsung Nexus S with Android v. 2.3.6.

Table 71. ETP message exchange syntax

Message	Protocol descriptors	JSON included attributes†
HEARTBEAT (single entry)	AGENT_ID	Agent's unique identifier, e.g. UUID.
	IP	Agent's current IP address.
	DATA_SRC_DETAILS (optional)	A generic identifier of the data source. It is either a single identifier, or a vector containing sensors' details in a name-value pair (Appendix C).
	TIMESTAMP	Timestamp of the agent's request.
	HASH	Hash value for the message's data.
CONFIGURE (multiple entries**)**	DATA_SRC_ID	A name-value (*sensor_type*: *value*) pair that selects a sensor data source, e.g. 8 (Proximity Sensor), 4 (Gyroscope).
	COLLECTION_OPTS (optional)	A vector of name-value pairs (*capture_duration*: *value*, *sample_rate*:*value*, *capture_interval*: *value*) that specifies capture configuration. For instance, capture_duration: 20 sec.
	TIMESTAMP	Timestamp of the agent's configuration.
	HASH	Hash value for the message's data.
ACQUIRE (multiple entries**)**	DATA_SRC_ID	A generic identifier of the data source. It is either a single sensor identifier, or a vector containing the following sensor's details in a name-value pair: (a) *RESOLUTION: value*, (b) *NAME: value*, and (c) *TYPE*: *value.*
	RAW_DATA	Contains raw data values, either a single value or a vector of values.
	TIMESTAMP	Timestamp of the data acquisition.
	HASH	Hash value for the message's data.

† See Appendix A for the description of the JSON Attributes.

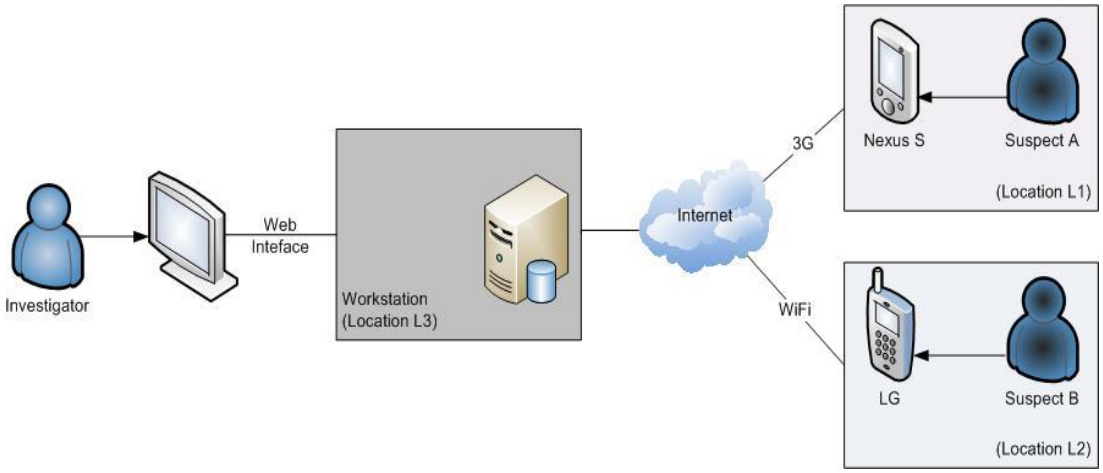


Figure 20: Laboratory generic setup

Table 72 summarizes sensor availability in the two devices and the emulator. The sensors are either hardware-based or virtual ones. The latter mimic hardware sensors and their measurements are derived by combining measurements from one or more hardware sensors. Android does not impose any sensor configuration in the underlying hardware. Thus, it is common that different devices have different sensor configurations. In this context, Samsung Nexus S includes all sensors, except for the three environmental sensors that are not commonly found in smartphones (i.e. humidity, pressure, temperature). LG Optimus L3, apart from the above mentioned three sensors, does not include the front camera, light, and gyroscope sensors. The device emulator only supports the camera, microphone and GPS sensors. Third-party software exists that adds support to the emulator for the rest sensors, e.g. OpentIntents[46]. However, to keep our tests realistic, we decided to carry out the collection of sensor data only from the real devices.

During our tests, both test devices were able to open network connections either via Wi-Fi or via the carrier's network (e.g. UMTS (3G), HSDPA). We also tested scenarios where network connectivity was unavailable after parsing the acquisition configuration file. In these scenarios the collected data were stored locally and transmitted when network connectivity was restored.

[46] OpenIntents, http://code.google.com/p/openintents/ (accessed November 2013).

7.5.2 Bandwidth and battery experiments

Effective data collection from smartphone sensors requires a proper configuration of the acquisition file. Otherwise, the device's battery or internal memory may be exhausted, which may result in making the device owner suspicious. Therefore, we expect that before the deployment of Themis its owners would perform initial stress tests, similar to those discussed herein, in order to avoid such occasions. These tests can be part of the configuration and training on the tools, which the forensic team will use. The aforementioned typically take place in the initial phase of a digital investigation, i.e. Preparation phase.

Table 72: Sensor availability in the testing devices

Sensor	Type	Samsung Nexus S	LG Optimus L3	Google Emulator
Accelerometer	hardware	X	X	-
Camera (Front)	hardware	X (X)	X (-)	emulated
GPS	hardware	X	X	emulated
Corrected Gyroscope	virtual	X	-	-
Gravity	virtual	X	X	-
Gyroscope	hardware	X	-	-
Humidity	hardware	-	-	-
Light	hardware	X	-	-
Linear Acceleration	virtual	X	X	-
Magnetometer	hardware	X	X	-
Microphone	hardware	X	X	emulated
Orientation	virtual	X	X	-
Pressure	hardware	-	-	-
Proximity	hardware	X	X	-
Rotation Vector	virtual	X	X	-
Temperature	hardware	-	-	-

In this context, we performed preliminary stress tests regarding the battery and bandwidth consumption of each sensor in the two test devices. While, these results are clearly not representative (i.e. they must be tested in more devices), they show a clear indication of each sensor's requirements.

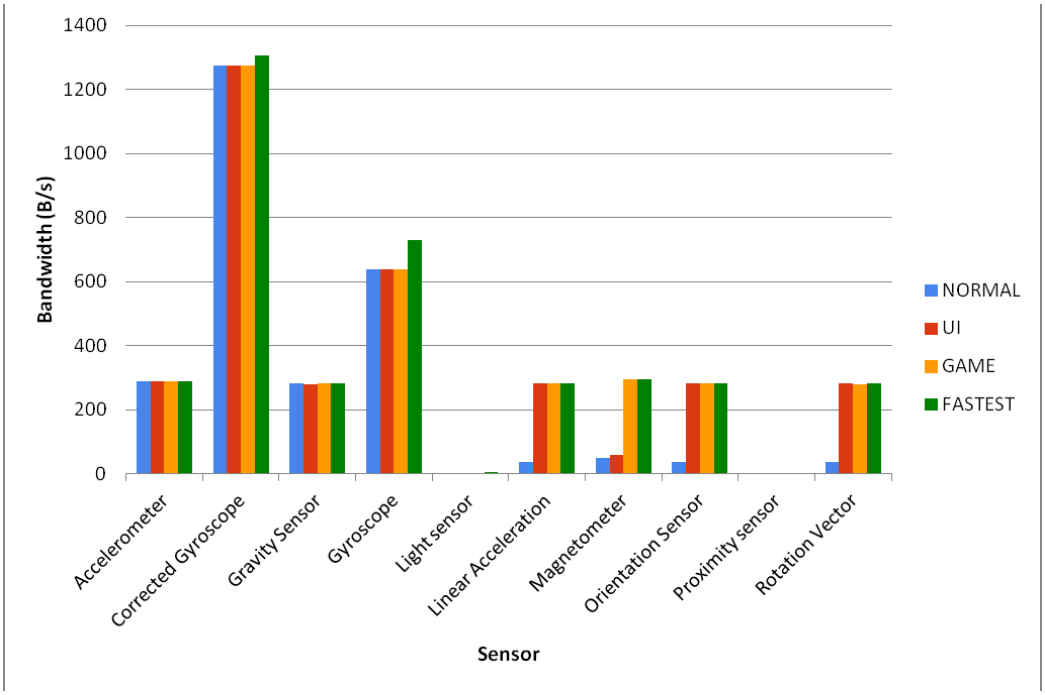


Figure 21: Bandwidth requirements for Nexus S (excluding multimedia sensors and GPS)

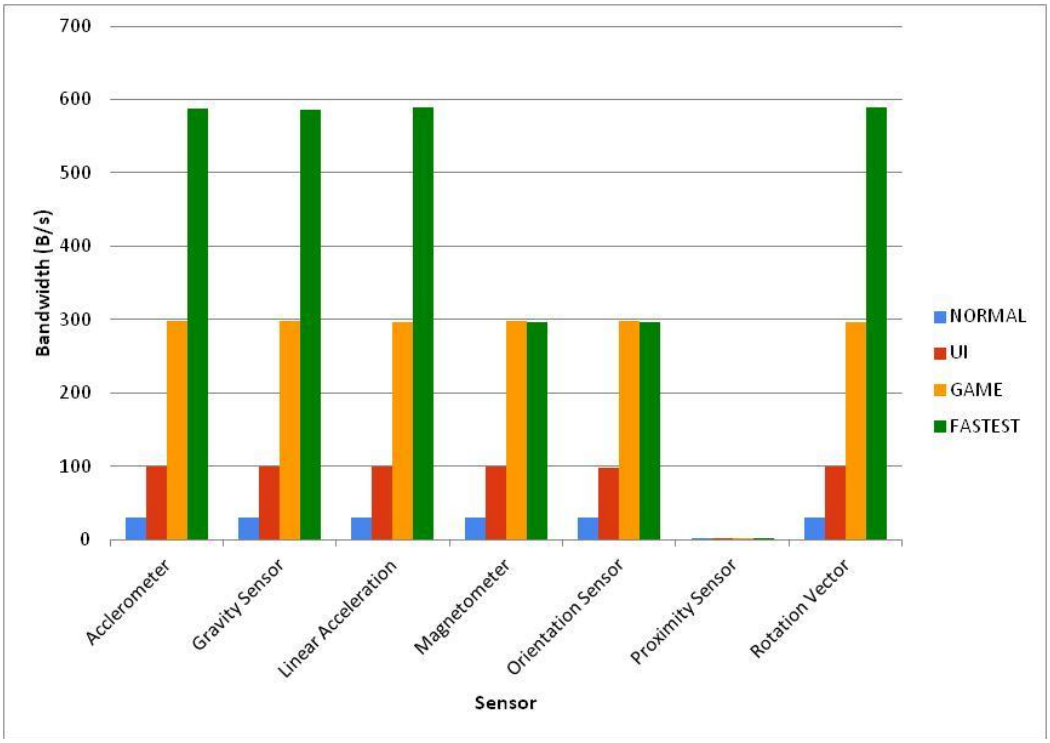


Figure 22: Bandwidth requirements for LG Optimus L3 (excluding multimedia sensors and GPS)

Figure 23: Bandwidth requirements for Nexus S (only multimedia sensors)

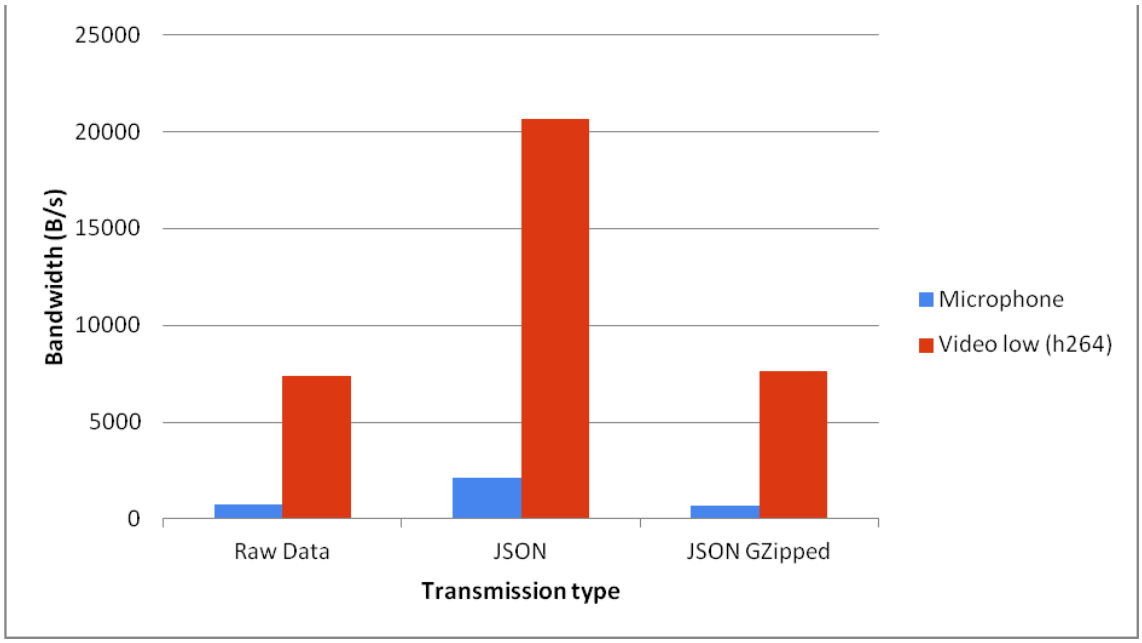


Figure 24: Bandwidth requirements for LG Optimus L3 (only multimedia sensors)

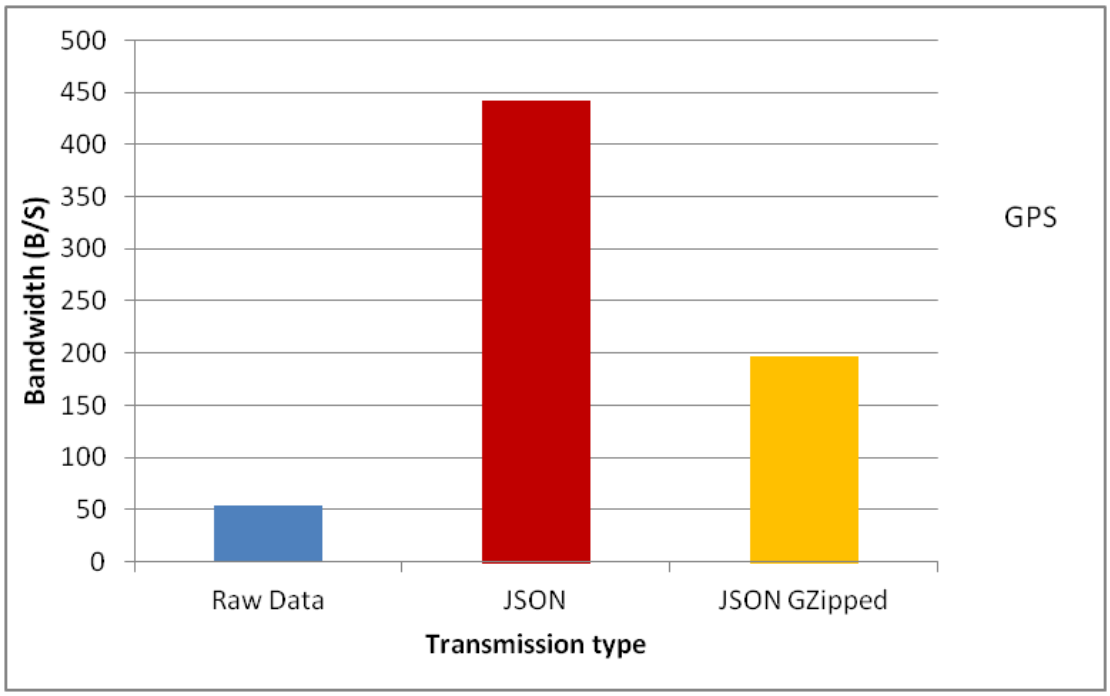


Figure 25: Bandwidth requirements for GPS (both devices)

Figures 5-9 summarize bandwidth requirements when the two devices were instructed to start data collection from all sensor and the devices were on a table (Appendix D refers to the details of these tests). We collected sensor data for 30sec in all available sampling rates of the sensor in Android[47] (where applicable), i.e. normal, ui, game, and fastest.

The results regarding bandwidth consumption are comparable, but not similar, due to the different hardware configurations of the two devices. Specifically, our results indicate that bandwidth requirements grow exponentially as the sampling rate increases. This is the case with the bandwidth experiments on LG L3. As expected the multimedia sensors consume the largest bandwidth. Among the rest sensors the gyroscope and corrected gyroscope are the most 'bandwidth hungry' sensors. Our, experiments showed that these two sensors produce more events in the given time frame. On the other hand, the light, GPS, and proximity sensors consume the lowest bandwidth.

The sensor's raw data are represented in Android with float array values. In ETP these values are inserted in JSON strings for their transmission, which introduces a bandwidth overhead (Figure 26 presents these tests for the accelerometer; the rest sensors are skipped for readability reasons). Moreover, the multimedia sensors are encoded into Base64, which introduces additional overhead (33%). This overhead is eliminated even with a simple compression algorithm (i.e. GZip), thus the final bandwidth requirements are comparable or less than the raw data requirements (Figures 7-8, 10).

[47] Sensors Overview, http://developer.android.com/guide/topics/sensors/sensors_overview.html (accessed November 2012).

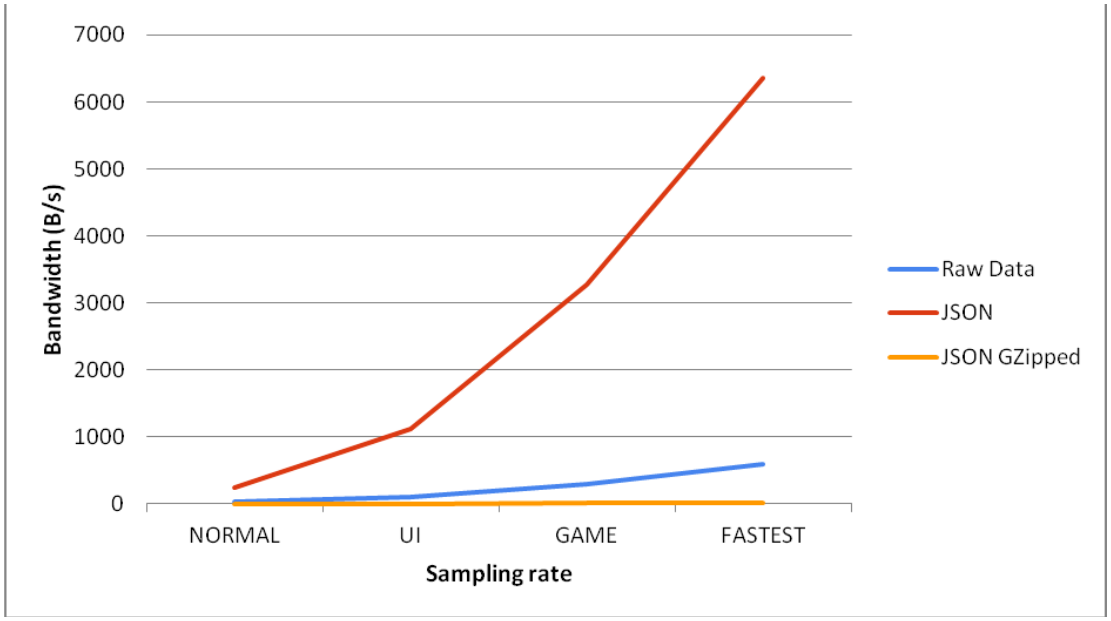


Figure 26: Accelerometer bandwidth requirements (LG L3)

Finally, we used Android's power API to get an approximation of power consumption for a subset of the device's sensors. These tests were software-based, thus their accuracy is unclear. Moreover, the API did not cover all available sensors (e.g. camera). These results are given in Figures. 11-12 (see Appendix D for details). In both tests we found that the use of the environmental sensors (i.e. light, proximity) and the accelerometer is cheap in terms of power consumption. For the latter, this is an interesting finding, since these sensors are well studied and applied in the literature to infer users activities (e.g. running, standing), or as side channel attacks (i.e. keylogging from a nearby PC keyboard).

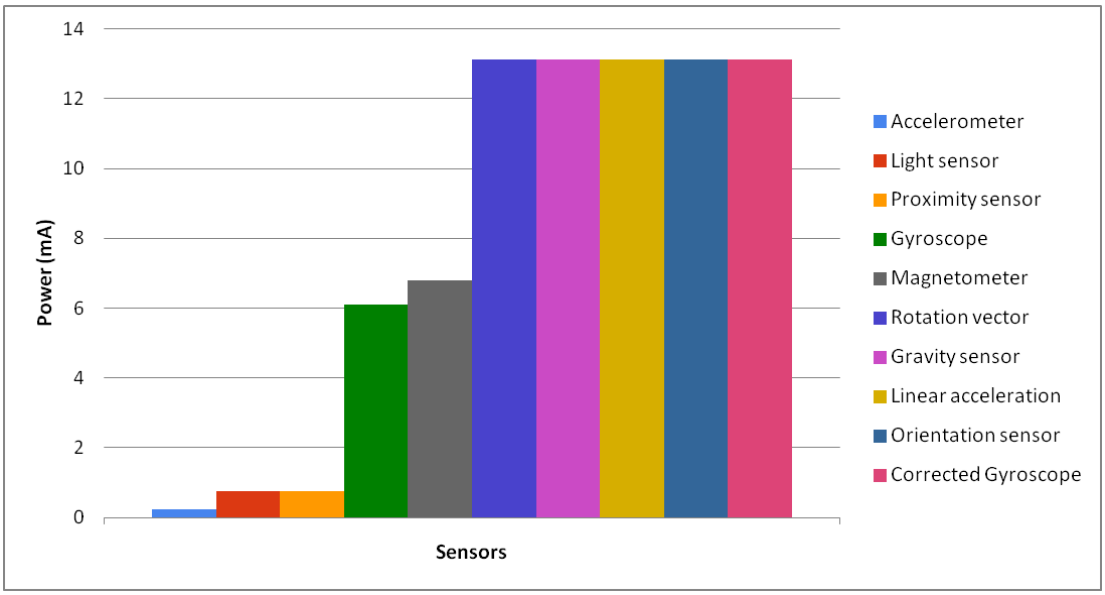


Figure 27: Battery requirements (Nexus S)

7.5.3 Themis scenarios

In all scenarios, which are presented in the sequel, we assumed that the suspect is a smartphone user. Moreover, we assume that the device has an active Internet connection via the carrier's network, as well as via Wi-Fi hotspots. Also, we assume that a digital investigation takes places for a crime that the legal and regulatory context considers severe. The scenarios that are presented do not aim to be exhaustive, but only indicative on the use of Themis in a digital investigation.

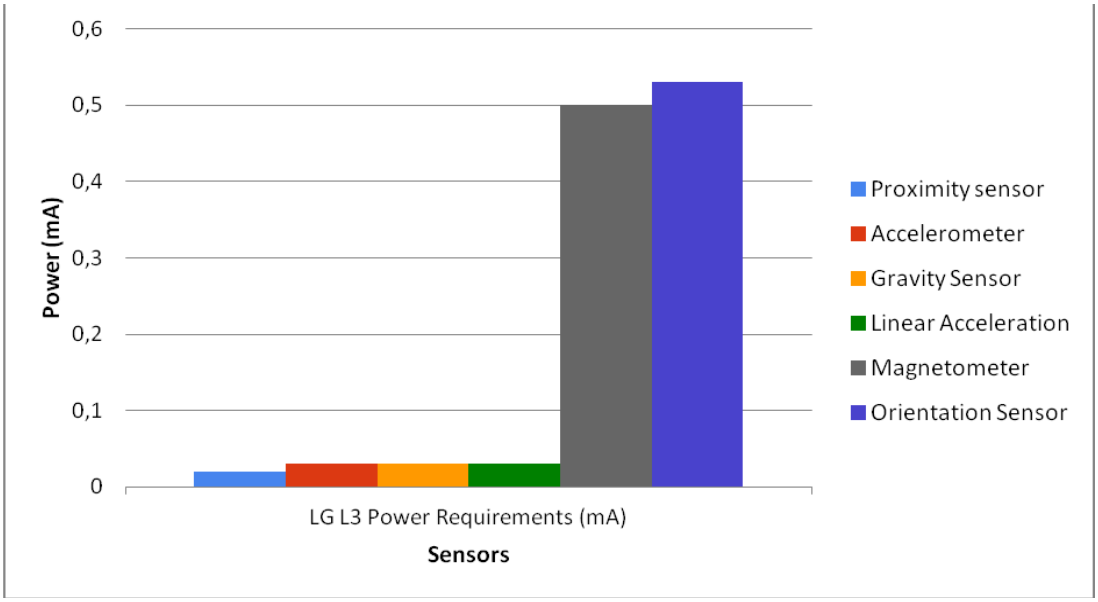


Figure 28: Battery requirements (LG L3)

7.5.3.1 Social engineering installation

The installation method that is selected to satisfy the requirements and/or restrictions of a forensic investigation affects the risk signals (i.e. requests for permission authorization) that will be presented to the device owner during app installation. Hence, if the forensic team selects manual installation or vulnerability exploitation for the agent's deployment, then the owner will not see any installation risk signals. This is because the app is installed without the device owner's 'participation' either remotely or via physical access to the device.

On the other hand, when the agent is installed via social engineering the risk signals that will be presented depend on the available sensor data sources that the investigator selects to access. In this context, a different variant of the agent may be deployed, e.g. one without access to the microphone if this data source is not used in the investigation.

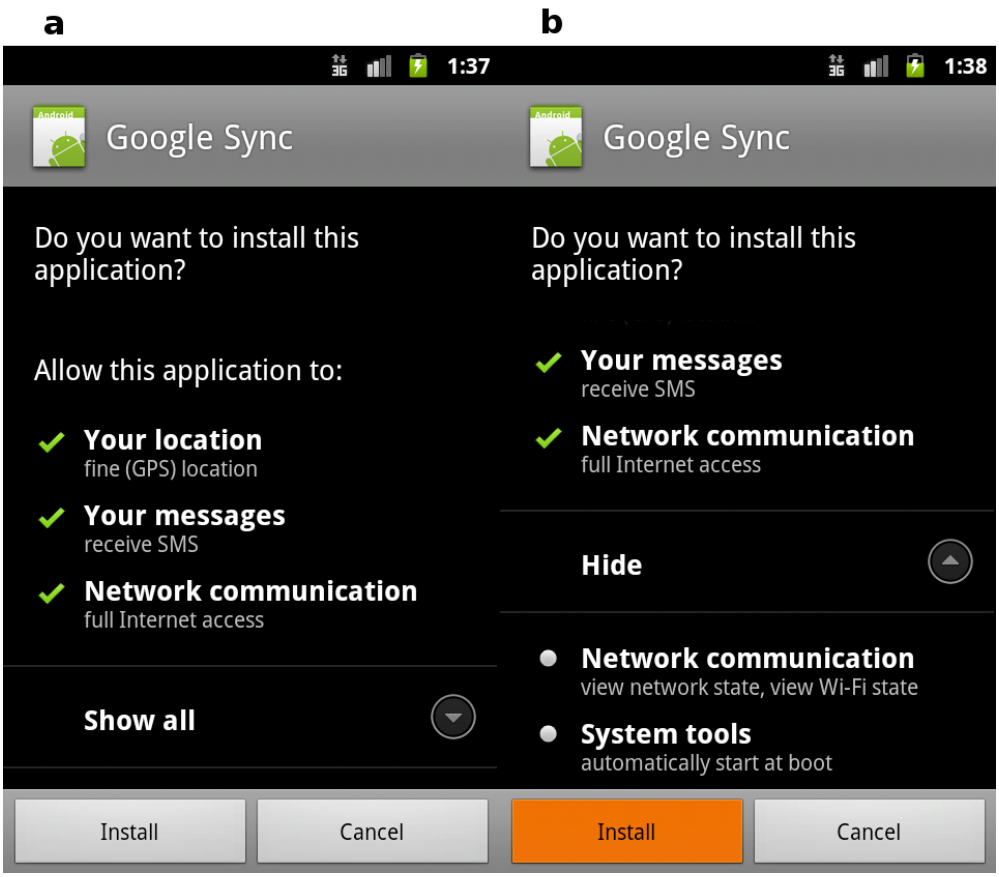


Figure 29: (a) Installation permissions. (b) Detailed installation permissions.

Herein we describe a scenario of the agent's installation via social engineering. We assume that during a digital investigation the suspect is convinced to download and install a third-party Android app, including the agent's functionality. Android allows the installation of apps that reside outside the official app repository (i.e. Google Play). Therefore, the installation of the app depends on whether the suspect will examine the apps permissions (Figure 29). We assume that the investigation team has reasons to believe that the suspect is not security and technically savvy. Hence, the suspect is more likely to accept the app's permission requests, according to the security discussion presented in section X.

In this scenario we present the use of a spoofing error message, i.e. an error message that look exactly similar to system warnings (Figure 30). Android's security model does not provide any visual distinction between error messages that are being presented by the operating system and the ones created by third-party apps. We used the spoofed error message of Figure 30b to mislead the suspect that the app is not installed. We also, ensured that the app does not appear in the device's menu with a quick launch icon. Themis will only show up in the list of running services with the app name Google Sync. In Android, other background services execute with a similar app icon as well as similar names, e.g. *Google Backup Transport*, *Google Contacts Sync*, *Google Partner Set up*. In this context, the suspect is expected to ignore the

message and continue with another task, assuming that the agent is not installed. This holds true as smartphone users do not adequately consider security issues during app installation (c.f. §X).

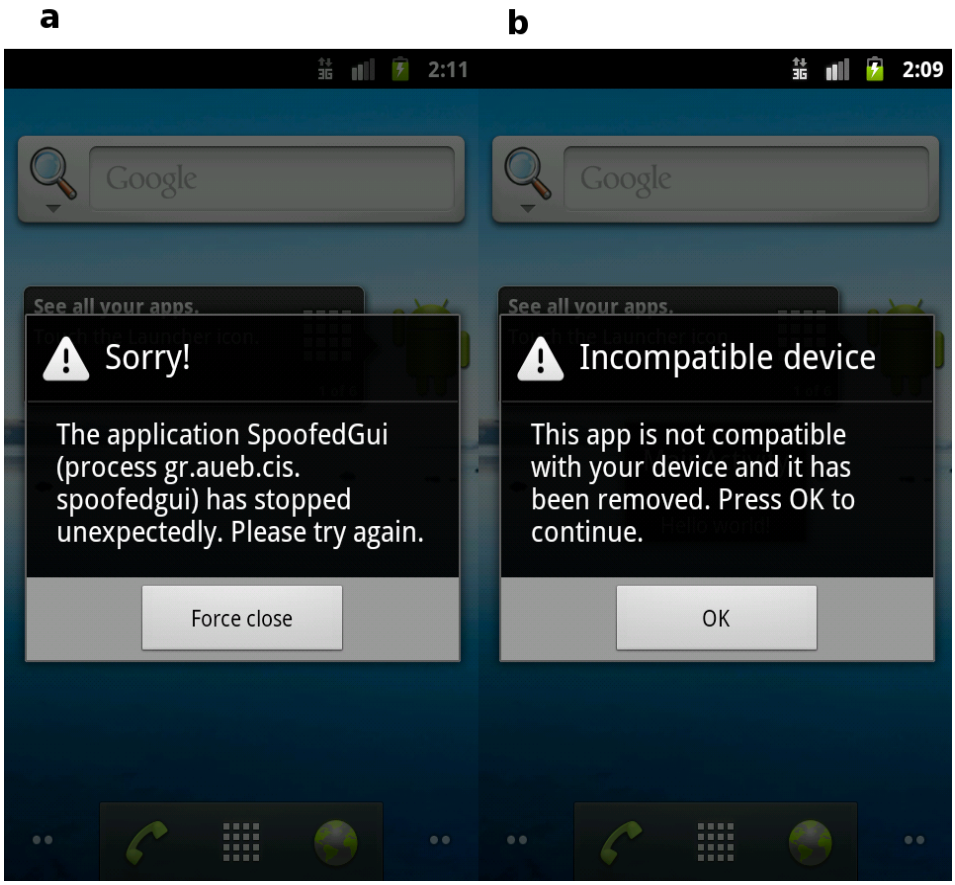


Figure 30: Legitimate error message (Android) and spoofed error message (Themis)

7.5.3.2 Validation of suspect's identity

In this scenario investigators must validate that the collected data are generated from actions belonging to the suspect. In general, a smartphone - in contrast to a computer - is a single user device, therefore it is uncommon that the device is lent or shared. However, in some cases the suspect may try to trick investigators and create false alibi, e.g. by giving the device to another individual or by placing it in a vehicle (e.g. leaving it in a bus).

In such a case, investigators may benefit from Themis by instructing the device to capture the device holder's face each time the device is accessed. The workstation's acquisition configuration string is included in Snippet 1.

Acquisition configuration entry:

[{"mdia_provider":"front_camera"},{"timestamp":"1351739491"}]|1d3b622a510b6f299fd96b9188b8bd55164e88e6

Collected data:

[{"MEDIA_RECORDER":"FRONT_CAMERA","TIMESTAMP":1351748589000",MEDIA_DATA":"<BASE64_BINARY_PHOTO>"}]959def79c843b10dc3b69035665f124a5abd5652

Snippet 1: Data used for validation of suspect's identity

The above acquisition string instructs the device to get a snapshot from the front camera, every time the user unlocks the device. In our implementation, we also access the proximity sensor every time such a command is sent to the agent. This is to deduce whether the screen's surface, and hence the front camera, is blocked with another object. For readability reasons we omit the presentation of the collected data in this subsection.

7.5.3.3 Validation of location data from the GPS

In this scenario we assume that the suspect has installed software in his device that protects his location privacy. This software tampers with data provided via the GPS (as in (Beresford et al., 2011; Hornyack et al., 2011; Zhou et al., 2011)) and returns static mocked values, namely (latitude=38.51618037248121, longitude=23.631649307180822). Moreover, we assume that the network provider is not cooperative. In this context, the only option to validate GPS data, if the investigator has reasons to believe that they have been tampered with, is to switch to an alternative location provider. Specifically, the investigator may use databases which map Wi-Fi MAC addresses or carrier cell identifiers to GPS coordinates (Android refers to these data as *network location provider*). These databases are used by location providers (e.g. Apple, Google, Skyhook), so as to improve their location-based services. The investigation team can use these databases to cross validate GPS data, if there are suspicions that they are invalid.

The investigator notices that the suspect's location remains unchanged for three hours when GPS data are collected from the device (see Figure 31 for a timeline of events[48] and Appendix D for the acquisition file as well as the collected data). Then, the investigator uses the acquisition configuration entry that is presented in snippet 2. As a result, Themis agent collects location data from the GPS, as well as the *network location provider*. By examining the col-

[48] For the creation of the timeline we extended the open source project available in http://www.simile-widgets.org/timeline/

lected data (Snippet 2), the investigator can unveil the presence of software which tampers with GPS data. This holds true, since on the one hand the GPS locations remain the same, but on the other location data from the *network location provider* change.

Acquisition configuration entry:

[{"location_provider":"gps","capture_interval":"3600", "capture_duration":"30"},{"location_provider":"network","capture_interval":"3600","capture_duration":"30"}]
0cff36a62b2dd5b0d06fac54ca5893b80e0245fe

Collected data:

[{"LOCATION":{"ACCURACY":840,"PROVIDER":"network","SPEED":0,"ALTITUDE":0,"TIMESTAMP":1360167232,"IS_LAST_KNOWN":false,"BEARING":0,"LONGTITUDE":23.7297103,"LATITUDE":37.9738075}},{"LOCATION":{"ACCURACY":10,"PROVIDER":"gps","SPEED":0,"ALTITUDE":48,"TIMESTAMP":1360167232,"IS_LAST_KNOWN":false,"BEARING":0,"LONGTITUDE":23.631649307180822,"LATITUDE":38.51618037248121}},{"LOCATION":{"ACCURACY":828,"PROVIDER":"network","SPEED":0,"ALTITUDE":0,"TIMESTAMP":1360170832,"IS_LAST_KNOWN":false,"BEARING":0,"LONGTITUDE":23.7283009,"LATITUDE":37.9767696}},{"LOCATION":{"ACCURACY":10,"PROVIDER":"gps","SPEED":0,"ALTITUDE":48,"TIMESTAMP":1360170832,"IS_LAST_KNOWN":false,"BEARING":0,"LONGTITUDE":23.631649307180822,"LATITUDE":38.51618037248121}},{"LOCATION":{"ACCURACY":838,"PROVIDER":"network","SPEED":0,"ALTITUDE":0,"TIMESTAMP":1360174432,"IS_LAST_KNOWN":false,"BEARING":0,"LONGTITUDE":23.7307187,"LATITUDE":38.0202412}},{"LOCATION":{"ACCURACY":10,"PROVIDER":"gps","SPEED":0,"ALTITUDE":48,"TIMESTAMP":1360174432,"IS_LAST_KNOWN":false,"BEARING":0,"LONGTITUDE":23.631649307180822,"LATITUDE":38.51618037248121}}]c98fb3025abeb3383f7fdc825d660f94f723d213

Snippet 2: Validation of location data

Finally, in case that in this scenario the mobile network provider is cooperative, the location data acquired from the device via the *network location provider* can be cross-validated.

Figure 31: Timeline of events

7.5.3.4 Extended context awareness

During a digital investigation, a cooperative provider may assist investigators by providing the suspects context. For instance, the carrier may provide approximation of the suspect's whereabouts. Nonetheless, sensor data provide a superset of context awareness that is available to the carrier, as well as the data measurements are more accurate. For instance, the combination of measurements from the location sensor with environmental sensors (e.g. light, temperature) and multimedia sensors (i.e. microphone, camera), may provide an extended context of a suspect. This context may be crucial to protect future victims (e.g. protection of minors, etc.) in time-sensitive crimes (Rogers et al., 2006).

In this context, Snippet 3 shows a snapshot of our experiments. In this case, combined data from the light sensor, the GPS, and the microphone were collected to get the extended context of the test device. The acquisition configuration string instructs the device to capture data from the light sensor for five seconds every five minutes, as well as data from the GPS and microphone for thirty seconds every five minutes.

The initial steps of the analysis included plotting the GPS data in a map (Figure 32). Then, we used the light sensor measurements to infer whether the suspect is indoors or outdoors. To do so we used a classifier (Appendix). The first value from the light sensor (i.e., 10874.4) suggested that the device was outdoors (right point on the map). The second value from the light sensor (i.e., 10.0) shows that the device was indoors (left point on the map). By using the timestamps of the sensor data we produced a timeline of events (Appendix), i.e. we inferred that the device's holder entered a building. This was subsequently confirmed by the holder of the test device. This timeline of events can be useful to investigators in order to evaluate a suspect's alibi.

The analysis ended by examining audio data, which were collected from the microphone. For readability and space reasons we omit the binary data of the audio. In the first audio data, loud sounds from car traffic were collected, giving additional evidence that the device was outdoors. In the second, we collected a snapshot of a conversation between two persons (a male and a female). Data from the microphone may reveal the social context of the users, thus investigators may benefit from such data in cases of restrictive measures (e.g. abuse of minor, etc.).

Acquisition configuration entry:

[{"sensor_type":"5","capture_interval":"300", "capture_duration":"5"},{"location_provider":"gps","capture_interval":"300","capture_duration":"30"},{"media_provider":"mic","capture_interval":"300","capture_duration":"30"}]
3a9a87c4430ece1cd4b10dc3b60038362577adee

Collected data:

[{"LUX":[10874.4],"SENSOR":{"RESOLUTION":1,"POWER":0.75,"NAME":"GP2A Light sensor","VERSION":1,"MINIMUM_DELAY":0,"MAXIMUM_RANGE":3000,"TYPE":5,"VENDOR":"Sharp"},"ACCURACY":[0],"REFERENCE_TIMESTAMP":1328551102117,"EVENT_TIMESTAMP":[7271133146000]},{"LOCATION":{"ACCURACY":10,"PROVIDER":"gps","SPEED":0.704008,"ALTITUDE":176,"TIMESTAMP":1328560614000,"IS_LAST_KNOWN":false,"BEARING":0,"LONGTITUDE":23.7689087212105,"LATITUDE":37.97929299895188}},{"MEDIA_RECORDER":"MIC","TIMESTAMP":1328454783000,"MEDIA_DATA":"<*BASE64_BINARY_AUDIO*>"}, {"LUX":[10.0],"SENSOR":{"RESOLUTION":1,"POWER":0.75,"NAME":"GP2A Light sensor","VERSION":1,"MINIMUM_DELAY":0,"MAXIMUM_RANGE":3000,"TYPE":5,"VENDOR":"Sharp"},"ACCURACY":[0],"REFERENCE_TIMESTAMP":1328551102117,"EVENT_TIMESTAMP":[2549520895000]},{"LOCATION":{"ACCURACY":20,"PROVIDER":"gps","SPEED":0,"ALTITUDE":188,"TIMESTAMP":1328560643000,"IS_LAST_KNOWN":true,"BEARING":0,"LONGTITUDE":23.76855537553537,"LATITUDE":37.97922642074484}},{"MEDIA_RECORDER":"MIC","TIMESTAMP":1328454783000,"MEDIA_DATA":"<*BASE64_BINARY_AUDIO*>"}]759def7cc83d08e4c27defc75665f24a52bd7353

Snippet 3: Extended context awareness data

Figure 32: Suspect's context

7.6 Ethical and legal considerations

Ubiquitous computing and communicating have a Janus Face, i.e., if a smartphone combines the functionalities and characteristics of both a cell phone and a computer it enables, on the other side, ubiquitous surveillance. A mobile device with such capabilities as a smartphone is actually a portable data carrier that may provide (also self-) incriminating data for almost every class and type of crime (Bennett, 2012). By providing tools for accessing files and messages as well as data not presently stored on the device, a smartphone offers law enforcement authorities "a treasure trove of evidence" (Morrissey, 2010). By tracking and searching a smartphone, law enforcement authorities may gain not only hard evidence, but also information and insight into valuable information, concerning the character, the habits and the context of a (punishable) behavior.

Data acquired from smartphones and sensor applications which are intended to be used as evidence, should have all the attributes of "conventional" evidence (Karyda and Mitrou, 2007). Primarily they need to be admissible, i.e. comply with the legal principles and requirements in a judicial criminal procedure), irrefutable complete and authentic, i.e. it must be possible to positively tie evidentiary material to the incident. Last but not least, they must be reliable, i.e. collected in accordance with formal requirements to establish its reliability. It is a matter of fact that, not only forensics evidence is only "as valuable as the integrity of the method that the evidence was obtained" (Bennett, 2012), but the process of every digital forensic investigation is subject to considerable scrutiny of both the integrity of the evidence and the integrity of the investigation process. Its reliability has to be demonstrated in consistence with the regulatory framework applicable in the jurisdiction in question (Kuntze et al., 2012). Infringing legal rules with regard to collection, conservation, communication and presentation of data, would compromise the admissibility of the evidence. In this respect, specific attention has to be paid to the compliance with the communicational and informational privacy requirements.

The use of forensic methods itself may constitute an intrusion of a person's fundamental right to privacy (Commission of the European Communities, 2006). Tracking a smartphone user and using sensor applications to gain information may implicate in both the communications secrecy and the privacy of a person.

(Lawful) Interception refers to the surveillance of a communication between communication partners in line with the law. However, data gathered through smartphone sensors may be deemed as external communications data (the so-called traffic data). According to the European Court of Human Rights (Case Malone v. UK, Case Copland v. UK), such data fall under the protection of correspondence (and communication) as laid down in Art. 8 of the European Convention for the Protection of Human Rights and Fundamental Freedoms. In parallel, such

data may also be regarded as personal (biometric) data, enabling the collection of sensitive information and even the profiling of the person concerned. Furthermore, installing such data collecting applications may affect the personality rights of the person under surveillance, as it may provide a revealing picture of the very core area of her private life. Both national constitutions and supranational texts (as the European Convention for Human Rights) allow restrictions on privacy and the freedom of communication in order to pursue legitimate public interests, such as national security, public safety, prevention and detection of crime. However, material and procedural rules have to ensure that collection and further processing of electronic evidence complies with the provisions guaranteeing privacy protection and communications' secrecy.

The relevant legal framework varies - sometimes considerably - across the different jurisdictions. The differences between the European approaches, in comparison to the common law countries, concern not only the substantive law requirements but also the constitutional background, the legal context, as well as the legislative technique of the relevant provisions. In US the critical constitutional framework for informational and communicational privacy consists of the Fourth Amendment and its interpretation by the courts, mainly the U.S. Supreme Court. The Fourth Amendment affirms the right of the people to be secure in their persons, homes, papers and effects, against unreasonable searches and seizure (Kerr, 2005; Salgado 2005). In general, the use of forensic methods in the course of a criminal investigation is usually subject to relatively strict procedural controls and guarantees, such as a judicial warrant (Karyda and Mitrou; 2007).

Does it mean that it is necessary to have proper legal authority in order to perform a forensic investigation of cellular telephones and mobile handheld devices? Is the acquisition of data and evidence through smartphone sensors' data comparable to a search of body or a search of premises or is it analogous with GPS tracking? The use of sensors may amount to a search, as it is capable to explore details about a person that would have previously been unknowable without psychical intrusion. Abel (2009) argues that if the data provided by such a device are remotely searched or are not presently stored within the confines of the device, it appears that the acquisition of evidence is comparable to a search of premises. It is worthy to mention that the German Federal Constitutional Court has placed strict limits upon the ability of law enforcement authorities to remotely access computers, PDAs and mobile phones. The Court has specified the rights to "informational self-determination" and "absolute protection of the core area of the private conduct of life" by lifting "security and integrity of information systems" as a fundamental right of the user (Togias, 2010; Wiebe, 2008). According to the theory this new right encompasses systems as PCs or smartphones as well as virtual hard drives or network-based application programs (Bäcker, 2009).

In the context of electronic evidence it is significant to distinguish between systems that are conceived and designed to serve a surveillance purpose and systems that are built and used with another purpose (e.g. traffic monitoring, personal use, etc.) (Coudert, et al., 2011). Especially in the latter case, the conditions of access of law enforcement authorities to the relevant information are of crucial importance. The German Federal Constitutional Court in a case concerning the use of automatic license plate recognition held that the retention of any digital data that was not predestinated for a specific use was too indiscriminate as to violate the right to privacy and – without guarantees and limitations - may amount to "complete surveillance" (Rushin, 2011). In USA the Supreme Court abstained from its previous jurisprudence about the "suspect's diminished expectation to privacy in an automobile on a public thoroughfare" (United States v. Knotts) and upheld in the case United States v. Jones (2012) the D.C. Circuit finding that an attachment of a GPS device to the defendant's vehicle to monitor the vehicle's movements, constituted a search under the Fourth Amendment and should be subject to a warrant (Reid, 2012). The D.C. Circuit pointed out the prolonged, ongoing monitoring of the suspect was much more intrusive, as it created an extensive log of a person's behaviour. The same considerations are applicable to the case of monitoring through smartphone's sensors.

The use and importance of digital evidence gathering is increasing (Rushin, 2011). This holds true due to the increasing smartphones use for every kind of communications (e.g. social networking, etc.) on the one side, as well as the technological feasibility and the decreased costs of easy and extensive data acquisition and retention on the other side. The ad-hoc data acquisition from smartphone sensors is proposed as a valuable method for combating serious crime and severe security threats both through detection and prevention.

Such methods in combination with cheap data storage capabilities provide incentives for organizations and law enforcement authorities to collect and retain data in anticipation of potential risks and future investigations. However, proactive data storage, where evidence gathering is dissociated of wrongdoing of specified suspects, has considerable and far-reaching impacts on fundamental rights. In this context, we should also consider the forensic readiness (Kuntze et al., 2012) of a device: the incorporation of design requirements aiming at ensuring availability and admissibility of data produced by the device has to comply with data protection and communication secrecy requirements.

Technological advances as well as the accessibility and the wide availability of data augments the expectations of law enforcement authorities, while lowering the privacy expectations of citizens, who sometimes may regard the trade-off between security (or even) convenience and loss of privacy worthwhile. The Carrier IQ software, which has been installed on more than 150 million smartphones and has the capacity to log web usage, to chronicle where and when, to which numbers calls and text messages were sent and received (Rose, 2012), indicates the pervasiveness of surveillance. In any case, users should be aware of the data that

may be "produced" through the applications and sensors they use. Laws that authorize the State to interfere in informational and communicational privacy must meet the standards of accessibility and foreseeability inherent in the rule of law. Thus, citizens can be aware of all circumstances in which State authorities may conduct surveillance, so that they can regulate their behavior accordingly to avoid unwarranted intrusions (European Court of Human Rights Cases Malone v. U.K., Kruslin v. France).

Concluding, proactive forensics in smartphone should be used only in restricted cases and in relation to the investigation. They may be acquired in extreme cases of prevention of illegal behavior and acts that the law defines as "serious crimes" or "national security threats" (e.g. organized crime). Moreover, protective mechanisms and procedures, such as in the proposed scheme, must be in place that hinder investigators or malicious individuals, from misusing the collection mechanism, and, in a way that allows the acquisition and preservation of data, without infringing fundamental rights.

7.7 Discussion

In this chapter, we examined the technological aspects of proactive digital forensics for smartphones. Proactive digital forensics is, per se, an interdisciplinary task, requiring the co-operation of computer and law scientists. Initially, we studied the associations between the smartphone data sources and evidence types, as well as the available connection channels in order to deliver potential evidence from a smartphone. Then, we reviewed the forensic investigation frameworks that are suitable for smartphone devices. Then, a proactive smartphone forensics scheme was developed.

Under proactive forensics, the ad-hoc collection of smartphone data can be a useful tool to law enforcement authorities in a digital investigation, both for the detection and sanction and the prevention of crime. In the same time, proactive and ad-hoc data collection is intrusive and raises ethical and legal issues. For this reason, in this chapter we have proposed an investigation scheme, which was created to avoid the misuse of the proactive, ad-hoc collection mechanism, either for law enforcement, or a malicious individual. By using this scheme, the appropriate person can collect only the smartphone data that are essential for combating 'severe' crimes.

Moreover, we implemented Themis, a generic tool for lawful remote ad-hoc acquisition of sensor data. During a data collection session with Themis sensor data are collected as soon as they are generated from an agent that is present on the suspect's device. Then, the data are transferred via a secure channel, with mechanisms that ensure their forensic soundness. Themis uses mechanisms, which are often found in malware, e.g. in spyware, to collect suspect's data. Due to the intrusiveness of the tool, it should only be used in extreme cases for detection

and prevention of illegal behaviour and acts that the law defines as "serious crimes" or "national security threats".

One may argue that the ad-hoc acquisition of smartphone data can be skipped since a subset of its data are available from the network provider (e.g. approximate location). Also, depending on the legal context and framework, it might be more convenient to acquire them from the network provider rather than the device. However, ad-hoc data acquisition from smartphone sensors is deemed as a valuable method for combating severe crime and security threats, both through detection and prevention. This holds true, since: a) the data acquisition from a smartphone may be used to validate evidence that have been collected from an provider that is not trusted, b) the smartphone may be the only available data source, since the provider may not be co-operative, c) data which are available from the provider often lack the accuracy of sensor data (e.g. location data that are collected from a provider are estimates of the real location and depend on the current cell that the device connects to), and d) some data are not available to the smartphone provider (e.g. light data).

Furthermore, we expect that the use of Themis would increase the effectiveness of law enforcement for crimes in which such intrusive data collection is allowed, especially for the time-sensitive ones (Rogers et al., 2006). This holds true as it can aid investigators to deduce the suspect's context and to act appropriately (preventively). In such cases (e.g. pedophilia, etc.), crime\incident prevention has a far more positive impact on the society, compared to a thorough analysis of collected data from a device that is performed after a crime has taken place.

Even though, Themis was tailored for proactive forensics, the tool's functionality does preclude its use for post-mortem forensics. In fact, it can be extended to perform logical acquisition for the rest smartphone data sources (e.g. SMS messages, contact list, etc.) during the collection phase (either on crime scene or in the forensic lab) of digital investigation frameworks for smartphones. In this context, Themis can be useful when connectors (i.e. data or power cables) are not available - either for budget or compatibility reasons - for a device that was found in a crime scene.

Themis can be also used as an 'anti-mugging' solution, in the context of participatory sensing (Lane et al, 2010). In this case, an individual may install the agent and enable it with a static phrase. e.g. "help". This phrase can be implemented to enable all smartphone sensors and upload them to servers operated by law enforcement. This software, on the one hand, may deter muggers from attacking individuals and, on the other hand, enable law enforcement to assess a case and thus more effectively manage police force. Finally, in corporate environment Themis can be used as a tool providing forensic readiness, or as part of an incident response system. In such a case, the use of Themis must comply with data protection and com-

munication secrecy requirements. In both cases, different trust and operational requirements may arise, but fall outside the scope of this work.

When such intrusive technology is used for the prevention of illegal behavior and acts that the law defines as “serious crimes” or “national security threats”, legal and institutional guarantees and tools must exist so as to prevent its abuse. In this context, we proposed that the intrusive technology that we implemented must be managed by a specialized Authority, which is trusted to comply with the legal framework and independent with regard to the police investigators. This trusted Authority, herein referred to as Law Enforcement Entity (LEE), hinders the abuse of the data collection mechanism by either law enforcement (i.e. intelligence gathering for random citizens irrespective of wrongdoing, etc.), or malicious individuals (e.g. espionage, cyber stalking, etc.). In this context, the LEE is a single point of failure in the proposed architecture. This can easily be managed if two different IA - equally trusted - are used. This modified architecture also complies with the separation of duties principle and achieves an increased level of trust. Furthermore, the operation of LEE is costly, since collected data must be stored ensuring their forensic soundness and compliance with the existing laws and regulations. We consider that storage requirements, with respect to the volume of the collected data, may not be prohibitive. This is since Themis is not proposed for surveillance of all citizens, but for a few malicious individuals who are involved in extreme cases illegal of behavior and acts that the law defines as “serious crimes” or “national security threats”.

The agent’s implementation is tailored for Android and its security model. Specifically, our implementation is compatible with all Android devices with version up *Gingerbread* (i.e. ver. 2.3.*). Some Android devices cannot update to a newer Android version either due to hardware incompatibility or due to the fact that the device manufacturer is not willing to provide the update. In this context, we expect that the implementation and security details that have been presented in this chapter will remain valid for a considerable population of Android devices. However, these details might not be valid for other platforms, or in the newer versions of Android. In this case, the agent must be specifically tailored for their security model.

Our implementation uncovers vulnerabilities in Android which were previously unexplored. Among them the fact that most smartphone’s sensors are unprotected, i.e. breaking Android’s sandbox rationale, is deemed as a rather serious vulnerability. This holds true, as patching this vulnerability would create considerable inconvenience in the platform’s ecosystem, since all the apps that access the previously unprotected sensors must be updated. This suggests that developers must redevelop their apps and resubmit them in the app repositories (e.g. Google Play), which in turn must manage again each submission and often analyze it for malware. Also, users must update their apps otherwise they will fail each time they try to ac-

cess sensor data. This however, apart from the cost in bandwidth and time, is often not a trivial task for novice users.

The effectiveness of data collection with Themis depends on the investigation team's available budget and expertise. The tool must be properly configured, otherwise the suspect's device resources may be consumed quickly (e.g. battery, device memory, etc.) and the suspect may become suspicious. In addition, the investigation team may need to collect the appropriate 'intelligence' regarding the suspect's device. This intelligence will enable the correct selection of exploits that Themis uses in order to be installed or to function. The collection of this intelligence is an important part of penetration testing, but is out of this chapter's scope.

Moreover, the effectiveness of data collection depends on the security level provided by endpoint security that is present. Currently, endpoint security in smartphones has to operate under the same sandbox rules that apply to all apps. This fact restricts their capabilities and for an enhanced security level it requires the rooting of the smartphone, which albeit voids their warranty. In a previous chapter, we proved that smartphone users adopt endpoint security in a really poor manner (c.f. §5). Even though in this work we did not elaborate on the evasion of smartphone endpoint security, Themis' agent could use mechanisms typically used by malware to evade security controls, such as covert channels or dynamic command execution at runtime (Zhou and Jiang, 2012).

For smartphone data collection we assume that an active Internet connection is present. In practice, such a connection may not be present (i.e. not purchased from the suspect) or, if it is present, its bandwidth quota is not infinite. In this context, the consumption of the suspect's available bandwidth quota may de-cloak the presence of a digital investigation. This can be avoided through assuming a cooperative network provider by using a 'silent' Internet channel, in the same manner that silent SMS messages[49] are used in some digital investigations, i.e. the suspect is not notified about the communication taking place.

'Traditional' digital forensics literature recommends that changing the device state must be avoided, as it is not considered a sound action. This is because it may have a negative impact on the admissibility of the evidence on courts of law (Lessard and Kessler, 2010). However, this trend constantly fades in 'modern' digital forensics. This holds true as in some circumstances rooting a device is the only effective way for data acquisition, e.g. live memory dumps are used in order to check the presence of malware, or to circumvent disk encryption. For instance, the investigator may need to acquire the memory contents from a live computer where she is logged in as a simple user (i.e. not administrator), thus, the only option to fulfill her goal is to elevate her privileges.

[49] http://www.tomsguide.com/us/cellphone-security-germany-sms-stealth,news-13717.html

In our work we change the device's state by installing an app and in some occasions by exploiting vulnerabilities for the agent's installation. In case the investigator uses rooting techniques in Themis, we argue that specific requirements must be met: a) this action has to be documented, b) this action must be performed only from a technically capable personnel, c) the exploitation vector must not infect the device with malware or open a backdoor, and d) the exploitation vector must be effective with the specific device (e.g. device model, operating system version, etc.), or else the device may be bricked. On the other hand, we argue that the documented installation of an app on the device does not affect its integrity. This is true, as apps in the majority of smartphone platforms are sandboxed, hence, the installation and execution of an app does not affect the files or execution of others. Moreover, in Themis data collection does not affect other apps and does not impair the integrity of the smartphone's sensors.

A limitation of our work is that our software assumes the integrity of the reported data from the sensor sources. Albeit, software may exist which provides mock data to any app in the device that requests access to its sensors (Beresford et al., 2011; Hornyack et al., 2011; Zhou et al., 2011). This software is not currently popular to users and it often requires the installation of a custom version of Android. However, in case such software becomes more widespread, Themis can circumvent it by hiding deeper in Android's kernel and directly accessing the sensor sources, i.e. use a functionality that is commonly used by rootkits in order to conceal their presence. Also, Themis can validate collected data from alternative sources, for instance location data from Wi-Fi MAC addresses (section 5.X)

Another limitation of our work is the validation that the collected data are generated by the suspect herself. Contrary to computers, smartphones are personalized devices that are normally used by only one individual. As a result, in a digital investigation the source of the collected smartphone data can be considered as a fait accompli. However, in some cases an investigator who collects ad-hoc sensor data may need to deduce whether the suspect actually uses her device hardware, or if he is tries to mock the investigation process (e.g. the device is left enabled in a vehicle in order to provide time or alibi to the suspect). This can be achieved with rather intrusive means or by non-intrusive ones. For the former, the investigator may try to infer if the suspect is using the smartphone by using the available technology, e.g. enabling the device's microphone and/or camera (section 5), using the suspect's gait as biometric for her identification (Mantyjarvi et al., 2005; Gafurov et al., 2006). For the latter, the investigator may simply attempt to contact the suspect through her device (i.e. place a phone call) and commence social engineering, in the same way attackers use social engineering for information gathering. In any case, the decision on whether such an action takes place or not is dependent on the nature of the crime and the expected expertise of the suspect (e.g. it may be

skipped if the investigative team has leads that the suspect is indeed using her device). This decision is normally taken during the preparation phase of the investigation.

Our work lacks the experience and input from law enforcement. This cooperation, on the one hand, could enable the testing of our implementation practicality in more realistic scenarios of digital investigations. On the other hand, law enforcement may provide feedback or training data for the creation of new classifiers for suspect's activities. For instance, classifiers could be built by getting accelerometer data from a police officer that fires a gun in a test lab. This classifier could be used to deduce when a subject is firing a gun, ensuring that either law enforcement does not misuse their powers during their service, or inferring if a suspect possesses a gun.

In the current implementation of Themis, we do not elaborate on the integrity of the agent. We rather assume that the agent is trusted since it is deployed from a trusted LEE. As a result, we assume that the agent does not insert false evidence, during data collection. In a corporate environment this assumption may not hold. Thus, the integrity of the agent must be tested before its operation (e.g. with remote attestation). However, this falls outside the chapter's scope.

7.8 Summary

Smartphones constantly interweave into everyday life, as they accompany individuals in different contexts. Smartphones include a combination of heterogeneous data sources, which can prove essential when combating crime. In this chapter, we elaborated on proactive digital forensics for smartphones. We examined potential evidence that may be collected from smartphones, as well as the available connection channels for evidence transfer during a forensic investigation. We proposed a proactive smartphone investigation scheme that focuses on the ad-hoc acquisition of smartphone evidence. We also, take into consideration the legal implications of the proposed scheme, as it is essential that the scheme includes prevention mechanisms, so as to protect individuals from misuse by investigators or malicious entities.

Among the data sources that are collectable from smartphones are the sensors (e.g. accelerometer, proximity sensor, etc.), i.e. hardware which can be used to infer the user's context. This context may be crucial in a digital investigation, as it can aid in the rejection or acceptance of an alibi, or even reveal a suspect's actions or activities. Nonetheless, sensor data are volatile, thus are not available in post-mortem analysis. Thus, the only way to timely acquire them, in case such a need arises during a digital investigation, is by software that collects them when they are generated by the suspect's actions. Therefore, in this chapter we examined the feasibility of ad-hoc data acquisition from smartphone sensors by implementing a device agent for their collection in Android, as well as a protocol for their transfer. Then, we discussed our experience regarding the data collection of smartphone sensors, as well as the

legal and ethical issues that arise from their collection. Finally, we describe scenarios regarding the agent's preparation and use in a digital investigation.

Chapter 7: Conclusions

"I never think of the future. It comes soon enough." – A. Eistein

TBC

Appendix

A. Security evaluation of web browsers

This subsection includes supplementary material for Chapter X.

List of controls. Table A.1 summarizes the security controls that were found in the browsers' interfaces, along with their indexes.

Table A. 1: List of controls

c#	Security Control label
1.	Auto update extensions
2.	Auto update plugins
3.	Block cookies
4.	Block images
5.	Block location data
6.	Block pop-ups
7.	Block referrer
8.	Block third-party cookies
9.	Browser update
10.	Certificate manager
11.	Certificate Warning
12.	Disable extension
13.	Disable java
14.	Disable javascript
15.	Disable plugin
16.	Enable do-not-track
17.	External plugin check
18.	History manager
19.	Local blacklist
20.	Malware protection
21.	Manually update extension
22.	Manually update plugin
23.	Master password
24.	Modify user agent
25.	Phishing protection
26.	Private browsing
27.	Proxy server

28.	Report rogue website
30.	Search engine manager
31.	SSL/TLS version selection
32.	Task manager
33.	Website checking

Hidden menus. The following hidden menus were encountered in the evaluation of security controls that are offered by web browsers.

Table A. 2: Hidden functionality in browsers

Browser	Hidden element	Element location
ABrowser	private menu	about:debug
Android (v. ICS, JB)	private browsing	tabs icons-> menu device key -> new incognito tab
Chrome all	private menu	about:about
Chrome	block referrer	execute Chrome with the parameter "—no-referrers"
Firefox all	private menu	about: about
Safari	developers tools	hidden menu item configurable from the advanced menu settings.
Safari	alter user-agent	develop (must be enabled see above) -> user agent-> other…

Modify user agent:

- Chrome**:** l5->Developer Tools->Settings-Overrides
- IE : l1-Developer Tools->Tools-Change user agent string

Navigation to proxy configuration. Assuming that the user is in the browser's configuration interface, i.e. either the browser's menu (e.g. Chrome Mobile, Firefox Mobile), or device menu (e.g. iPhone Safari settings), the following navigation clearly violates the three-click rule; hence it is more likely that the control will not be found.

Table A. 3: Navigation to proxy configuration widget

Browser	Path
Safari Mobile	tap back (settings) -> scroll up -> wifi -> connected wifi network (hit blue icon -> scroll down -> http proxy manual -> setting server & port
Android Gingerbread	home button-> options button from device ->settings->wireless and networks->wi-fi settings->options button from device ->advanced->wi-fi proxy->fill in proxy details
Android ICS/ JB	home button-> device's options button -> settings-> wifi->on->hold network id* -> modify network-> scroll down-> check show advanced options-> scroll down-> proxy setting -> manual -> scroll down-> fill in proxy details -> tap save *unless the user holds the network id for a few seconds the hidden menu will not appear
Windows Phone 7.5	Settings (General) -> Scroll down to Wi-Fi Option -> Toggle Wi-Fi networking -> tap the desired WiFi Network from the list -> toggle Proxy option -> specify any additional proxy options needed

Heatmap data. Tables A.4-5 provide raw data for the heatmaps which were presented in §5.3, rega**r**ding the number of security controls in each browser that are: (a) available and default enabled and (b) manageable by the user.

Table A. 4: Number of security controls that are preconfigured and enabled in each browser

Preconfigured & enabled controls	**# of controls**	**GC**	**MF**	**IE**	**OP**	**AS**	**AB**	**CM**	**FM**	**IM**	**OM**	**Om**	**SM**
Annoyance	9	2	2	2	2	2	3	2	3	2	2	1	2
Browser fingerprinting	4	0	0	0	0	0	0	0	0	0	0	0	0
Exploits / Malware	17	4	4	4	4	2	3	2	5	2	3	2	2
Identity theft	8	1	1	1	1	1	0	0	1	0	1	0	1
Interception of network data	4	1	1	1	1	1	1	1	1	1	1	0	1
Phishing	9	3	3	3	3	3	2	2	3	2	3	1	3
Privacy breach	22	4	3	4	3	5	2	1	4	1	3	0	4
Resource abuse	10	1	1	1	1	1	1	0	2	0	1	0	0
Rogue certificates	3	1	1	1	1	1	1	1	1	1	1	0	1
Spam advertisements	5	1	1	1	1	1	1	1	1	1	1	1	1
Technical failure of browser	12	3	3	3	3	1	2	1	4	1	2	1	1

Table A. 5: Number of controls that are manageable from users in each browser

Manageability controls	**# of controls**	**GC**	**MF**	**IE**	**OP**	**AS**	**AB**	**CM**	**FM**	**IM**	**OM**	**Om**	**SM**
Annoyance	7	7	7	8	5	6	3	3	1	2	1	2	7
Browser fingerprinting	3	3	3	4	2	2	2	1	1	1	0	1	3
Exploits / Malware	10	13	11	12	8	4	3	4	1	2	0	2	10
Identity theft	5	7	7	8	3	1	3	3	0	1	0	3	5
Interception of network data	2	2	3	3	0	2	1	0	0	1	0	0	2
Phishing	6	7	8	8	4	4	3	0	0	2	0	3	6
Privacy breach	16	17	18	21	11	7	7	9	3	3	2	5	16
Resource abuse	7	7	8	8	4	2	1	2	0	0	0	1	7
Rogue certificates	2	2	2	2	0	2	1	0	0	1	0	0	2
Spam advertisements	5	5	5	5	3	2	1	0	0	1	1	1	5
Technical failure of browser	8	9	6	6	6	2	1	3	0	0	0	1	8

B. Security awareness of smartphone users

Instrument model. The following table summarizes the categorical variables, which were used in the statistical analysis of the survey described in Chapter XX.

Table B. 1: Variables in the instrument model

Variable	Variable label	Values	Question
AgreementMsgs	Agreement messages	Never, Sometimes, Always	Q12
Android	Android OS device	No, Yes	Q6
AppTesting	Application testing in OAR	Don't know, No, Yes	Q10
BlackBerry	BlackBerry OS device	No, Yes	Q6
Business	Store business data	No, Yes	Q23
Cost	Cost	No, Yes	Q8
Developer	Developer	No, Yes	Q8
Efficiency	Efficiency	No, Yes	Q8
Encryption	Use encryption	No, Yes	Q20
FreeSoft	Searched for free smartphone security software	No, Yes	Q21
iPhone	iOS device	No, Yes	Q6
IsAware	Security savvy user	No, Yes	Q4
IT	IT expertise	Moderate, Good, Excellent	Q3
Jailbreaking	iPhone jailbreaking	No, Yes	Q14
MalwareExistence	Smartphone malware existence	No, Yes	Q16
Misplaced	Device misplaced	No, Yes	Q24
NoUse	No use of security software	No, Yes	Q17
PassLock	Use device password lock	No, Yes	Q20
Pc	Use security software in PC	No, Yes	Q17
Personal	Store personal data	No, Yes	Q22
Pirated	Prefer Pirated Apps	No, Yes	Q15
Privacy	Privacy Concern	No, Yes	Q13
RemoteLoc	Use remote locator	No, Yes	Q20
RemoteWipe	Use remote data wipe	No, Yes	Q20
Reputation	Reputation	No, Yes	Q8
Reviews	Reviews	No, Yes	Q8
SecMessages	Security messages	Never, Sometimes, Always	Q11
SecSoftExistence	Smartphone security software existence	No, Yes	Q18
Security /Privacy	Security or Privacy	No, Yes	Q8
Smartphone	Use security software in smartphone	No, Yes	Q17
SoftEssential	Smartphone security software essential	No, Yes	Q19
Symbian	Symbian OS device	No, Yes	Q6
TrustRep	Trust app repository	No, Yes	Q9
Usability	Usability	No, Yes	Q8
Usefulness	Usefulness	No, Yes	Q8
Windows	Windows OS device	No, Yes	Q6

Instrument. The measurement instrument (i.e. the questionnaire) of the survey is given herein.

Athens University of Economics and Business
Information Security and Critical Infrastructure Protection
Research Laboratory

This is a voluntary and anonymous questionnaire. **Please read the following questions and answer as honestly and responsibly as possible**.

Research coordinators: **Alexios Mylonas**, Ph.D. Researcher (amylonas@aueb.gr)
Dimitris Gritzalis, Professor (dgrit@aueb.gr)

1. Sex Male □ Female □

2. Age: _____

3. IT expertise: Excellent □ Good □ Moderate□

4. Have you completed an information security course (undergraduate or postgraduate) or relative certification (e.g. CISSP, CISA)? Yes □ No □

5. I am an owner of a: Smartphone □ PC/Laptop/Netbook □ Other:

6. Which is the operating system of your smartphone?
Android □ BlackBerry □ Symbian □ Windows Mobile□ iPhone iOS □ Other:

7. Do you install new applications on your smartphone? Yes □ No □

8. What are your main criteria in choosing which application to install on your smartphone?
_____

9. Do you consider the applications in the official application repository to be secure for installation on your device? Yes □ No □

10. Have the applications in the official application repository undergone security application testing before you download them on your smartphone?
Yes, they have □ No, they have not □ I do not know □

11. Do you pay attention to security messages appearing during the installation of a new application on your smartphone? Always □ Sometimes □ Never □

12. Do you pay attention to licensing agreements appearing during the installation of a new application on your smartphone? Always □ Sometimes □ Never □

13. Are you concerned about the privacy and protection of your personal data?
Yes □ No □

14. Do you know what the term iPhone jailbreaking means? Yes □ No □

15. Do you prefer the installation of a pirated application instead of buying the original application version? Yes □ No □

16. Are you aware of the existence of smartphone malicious software (virus, worm, trojan horse, etc.)? Yes □ No □

17. In which devices do you use security software (e.g. antivirus, firewall, etc.)?

Smartphone □ PC/Laptop/Netbook □ None□ Other:

18. Are you aware of the existence of smartphone security software (e.g. antivirus, firewall, etc.)? Yes □ No □

19. Do you consider smartphone security software essential? Yes □ No □

20. Which of the following smartphone protection mechanisms do you use?

File Encryption □ Device Password Lock □ Remote file deletion□ Device Locator □ None □ Other:

21. Have you ever searched the application repository for free smartphone security software?

Yes □ No □

22. Do you store personal data on your smartphone? Yes □ No □

23. Do you store business data on your smartphone? Yes □ No □

24. Have you ever misplaced your smartphone? Yes □ No □

Thank you for your time and effort.

Sample size estimation. To conduct our statistical analysis tests with the x^2 distribution with power level 95% and significance level 5%, we a priori estimated the required sample size (Cohen, 1998). Table B. 2 summarizes the results of this test.

Table B. 2: Test results for a priori sample size estimation

Effect size w	error probability (a)	Power (1-β error probability)	df	Required sample size
0.3	0.05	0.95	1	145

Sample's age distribution. Table B. 3 summarizes the age distribution of the survey respondents. The majority of them is aged [15-30].

Table B. 3: Distribution of ages in the sample

Range	% of users
[15-22]	43
[23,30]	38
[31-52]	19

C. Sensor data as digital evidence

The following tables include the description of JSON attributes, which are used by Themis.

Table A. 6: Description of JSON attributes that are used in the ETP in the CONFIGURE message

Attribute	Description
sensor_type	Sensor's identifier (Android API specific)
location_provider	Location provider identifier (e.g. GPS, network)
media_provider	Multimedia sensor's identifier (i.e. front/back camera, microphone)
capture_duration	Capturing time, measured in seconds
capture_interval	Time between two capturing actions, measured in seconds

Table A. 7: Description of JSON attributes that are used in the ETP in the ACQUIRE message

Attribute	Description
reference_timestamp	Unix timestamp of collection start (in nanoseconds)
event_ timestamp	Time passed since smartphone's uptime (in nanoseconds)
timestamp	Unix timestamp (in seconds)
is_last_known	Used for cached locations

Table B.1 summarizes the details pertinent to the sensor hardware in the two test devices, i.e. Nexus S and LG Optimus L3. These details must be taken into consideration when data acquisition is being configured (e.g. for the selection of transport channel), as well as in the manual analysis of the recovered data.

Table B. 4. Sensor data details pertinent to the two test devices

Sensor	Values	Range (Nexus S)	Range (LG)	Unit
Accelerometer	X, Y, Z	19.6 (abs)	39.22(abs)	m/s^2
Camera	raw	N/A	N/A	-
GPS	X,Y	[0.0-90.0], [-180.0-180.0]	[0.0-90.0], [-180.0-180.0]	rad
Gravity	X, Y, Z	19.6 (abs)	39.22 (abs)	m/s^2
Gyroscope	X, Y, Z	34.9 (abs)	-	rad/s
Humidity	Single	-	-	%
Light	Single	$[0.0 – 30.0x10^5]$	-	Lx
Linear Acceleration	X, Y, Z	19.6	39.22	m/s^2
Magnetometer	X, Y, Z	2000.0	1600.0	μT
Microphone	raw	N/A	N/A	-
Orientation	X, Y, Z	[0.0-360.0]	[0.0-360.0]	rad
Pressure	Single	-	-	hPa/mbar
Proximity	Single	[0.0 – 5.0]	[0.0 – 1.0]	Cm
Rotation Vector	X, Y, Z	1.0 (abs)	1.0	rad
Temperature	Single	-	-	°C

An example of a vector containing sensor's info in a name-value pair is the following:

1. RESOLUTION: 0.038300782
2. POWER: 0.03
3. NAME":"Linear Acceleration Sensor"
4. "VERSION":1
5. "MINIMUM_DELAY":0
6. "MAXIMUM_RANGE":39.22
7. "TYPE":10
8. "VENDOR":"Google Inc."

Tables D.1-D.6 summarize the results of our tests regarding bandwidth consumption for each sensor in the test devices.

Table D.1: Bandwidth requirements for Nexus S **(**excluding multimedia sensors and GPS**)**

Sensor/Rate (B/s)	NORMAL	UI	GAME	FASTEST
Accelerometer	287,2	288,4	288,8	289,2
Corrected gyroscope	1274,8	1274,8	1274,4	1307,34
Gravity sensor	280,8	280	282	281,6
Gyroscope	636,8	636,8	636,8	729,94
Light sensor	0,14	0,27	1,74	6,54
Linear acceleration	35,6	281,6	282,4	282,4
Magnetometer	48	60	296,4	295,6
Orientation sensor	35,6	282,4	281,6	282,4
Proximity sensor	0,14	0,14	0,14	0,14
Rotation vector	35,6	281,6	280,4	280,8

Table D.2: Bandwidth requirements for Nexus S (only multimedia sensors)

Multimedia sensors on Nexus S	Raw Data	JSON	JSON GZipped
Microphone	1699,2	4713,47	1668,7
Video low (h264)	29478,54	82345,87	30602,07
Video high (720p, h264)	158121,1	427204,4	161070,54

Table D.3: Bandwidth requirements for LG Optimus L3 (excluding multimedia sensors and GPS)

Sensor/Rate (B/s)	NORMAL	UI	GAME	FASTEST
Accelerometer	30	99,2	297,6	587,2
Gravity sensor	30	100	297,6	586,4
Linear acceleration	30	99,6	296,8	588,8
Magnetometer	30	100	297,6	297,2
Orientation sensor	30,4	98	298	297,2
Proximity sensor	0,14	0,14	0,14	0,14
Rotation vector	30	99,6	297,2	589,2

Table D.4: Bandwidth requirements for LG Optimus L3 (only multimedia sensors)

Multimedia sensors on LG L3	Raw Data	JSON	JSON GZipped
Microphone on LG L3	755,64	2095,74	675,6
Video low on LG L3	7387,57	20659,27	7636,1

Table D.5: Bandwidth requirements for GPS (both devices)

Location providers	Raw data	JSON	JSON GZipped
GPS/Network Provider	53,34	441,6	197,34

Table D.6: Bandwidth requirements for accelerometer **(**LG Optimus L3**)**

Accelerometer on LG L3	Raw Data	JSON	JSON GZipped
NORMAL	30	237,47	3,94
UI	99,2	1110,2	5,37
GAME	297,6	3275,67	7,94
FASTEST	587,2	6363,8	11,57

Tables D.7-D.8 summarize the results of our tests with battery consumption for each sensor in the test devices.

Table D.7: Power requirements Nexus S

Sensors	Power (mA)
Accelerometer	0,23
Magnetometer	6,80
Light sensor	0,75
Proximity sensor	0,75
Gyroscope	6,10
Rotation vector	13,13
Gravity sensor	13,13
Linear acceleration	13,13
Orientation sensor	13,13
Corrected gyroscope	13,13

Table D.8: Power requirements LG L3

Sensors	Power (mA)
Accelerometer	0,03
Magnetometer	0,50
Proximity sensor	0,02
Gravity sensor	0,03
Linear acceleration	0,03
Orientation sensor	0,53

Snippet D.1 includes the acquisition configuration entry and the collected data from the scenario of section 5.3.3.

Acquisition configuration entry:

[{"location_provider":"gps","capture_interval":"3600", "capture_duration":"30"}]ed0a3ecf6e81c3d6b9ab50c31c84e53c7e71eb92

Collected sensor data:

[{"LOCATION":{"ACCURACY":10,"PROVIDER":"gps","SPEED":0,"ALTITUDE":48,"TIMESTAMP":1360156432,"IS_LAST_KNOWN":false,"BEARING":0,"LONGTITUDE":23.631649307180822,"LATITUDE":38.51618037248121}},{"LOCATION":{"ACCURACY":10,"PROVIDER":"gps","SPEED":0,"ALTITUDE":48,"TIMESTAMP":1360160032,"IS_LAST_KNOWN":false,"BEARING":0,"LONGTITUDE":23.631649307180822,"LATITUDE":38.516180372481 21}},{"LOCATION":{"ACCURACY":10,"PROVIDER":"gps","SPEED":0,"ALTITUDE":48,"TIMESTAMP":1360163632,"IS_LAST_KNOWN":false,"BEARING":0,"LONGTITUDE":23.631649307180822,"LATITUDE":38.5161803724 8121}}]829028f9722bdc716e6f73a47451d17b9bbfea06

Snippet D.1: GPS location data

Figure D.1 presents the timeline of events of section 5.3.4.

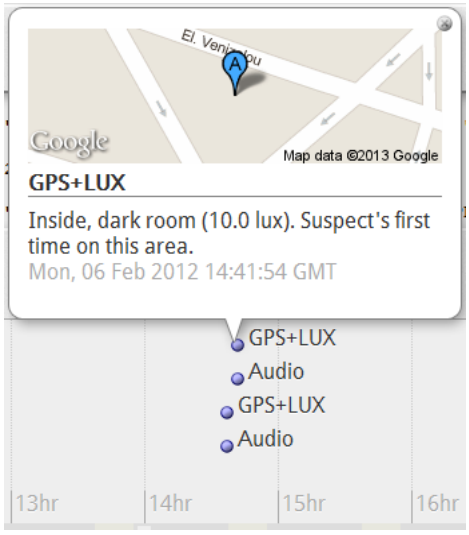


Figure D.1: Timeline of events of extended context awareness scenario

Table E.1 summarizes the results of our experiments with the light sensor. Daylight fluorescent tubes give an illumination value approx. 10^4 when the distance between the tube and the

sensor is kept to a minimum. In our experiments, we noticed that illumination values are greater than 10^4 when the device is outdoors on a sunny day. While these tests are only indicative, this sensor may be useful for a forensic investigation process, in the sense that if it is combined with location data it may give specific information regarding the context of the device (i.e. device is indoors or outdoors).

Table E. 1: Summary of light sensor experiments

Scenario tested	**Illumination value**
Device in a pocket	4.68
Room without light tubes on	4.72
Room without light tubes on, near 4 PC monitors	21.16
Room with fluorescent light tubes	246.66
Right under fluorescent light tube	688.48
20cm from 4 fluorescent light tubes	6483.15
Cloudy noon daylight	8834.43
5cm from 4 fluorescent light tubes	9891.19
Sunny morning 08:30 under thin shadow	10874.4
Sunny morning 09:30 facing sun	>>11000 (spikes from $11x10^3$ to $6x10^4$)

D. Risk assessment of privacy threats

This subsection includes supplementary material for Chapter X.

Level of Identification. Table C. 1 presents our proposed static mapping of the ability of a permission to identify an individual. Permissions for assets that provide a weak correlation with the identity of the Android user (e.g. under specific assumptions the location of the device may be used to infer the user's identify) are assigned to score 1. For the assets where these assumptions are more likely to occur in certain cases (for instance a camera or audio snapshot identifies the user, the user refers to his identity in a text message), their permissions are mapped to level 2. Finally, the permissions that provide access to data that relatively easy identify users (e.g. emails, Facebook accounts, etc.) are assigned to level 3 and the ones that provide strong correlations with the user identity (e.g. users name or phone number) are assigned to level 4.

Table C. 1: Level of Identification per Permission

Permission	Identification
ACCESS_COARSE_LOCATION	1
ACCESS_FINE_LOCATION	1
ACCESS_NETWORK_STATE	1
ACCESS_WIFI_STATE	1
AUTHENTICATE_ACCOUNTS	3
BLUETOOTH_ADMIN	1
CAMERA	2
GET_ACCOUNTS	3
GET_TASKS	1
PROCESS_OUTGOING_CALLS	2
READ_CALENDAR	1
READ_CALL_LOG	2
READ_CONTACTS	2
READ_EXTERNAL_STORAGE	2
READ_HISTORY_BOOKMARKS	1
READ_LOGS	1
READ_PHONE_STATE	4
READ_PROFILE	4
READ_SMS	2
READ_SOCIAL_STREAM	2
READ_USER_DICTIONARY	1
RECEIVE_MMS	2
RECEIVE_SMS	2
RECEIVE_WAP_PUSH	1
RECORD_AUDIO	2
SUBSCRIBED_FEEDS_READ	1
USE_CREDENTIALS	3
WRITE_EXTERNAL_STORAGE	2

Severity Assessment. This is an indicative **questionnaire** used to address the severity of impact of applicable threat scenarios.

- *Q1*. What will happen if *someone tracks* your *location*? (**T1**,**T3**)
- *Q2*. What will happen is *someone listens* to your *phone calls*? (**T2**)
- *Q3*.What will happen if *someone takes* a *photo* or a *video* of you *without* your *knowledge*? (**T2**)
- *Q4*. What will happen if *someone reads* your *SMS* or *emails*? (**T2**,**T5**)
- *Q5*. What will happen if *someone reads your contacts*? (**T2**,**T5**)
- *Q6*. What will happen if *someone reads your calendar*? (**T1**,**T5**)
- *Q7*. What will happen if your *bookmarks* and *browsing history* is sent to *your friends*? (**T5**)
- *Q8*. What will happen if your *calling history* is *sent* to *advertisers*?(**T3**)
- *Q9*. What will happen if *someone accesses* your *credentials*? (**T4**)
- *Q10*. What will happen if *someone reads without your permission* your *documents* or *media* from the *external storage*? (**T5**)
- *Q11*. What will happen if *someone reads* your *identification info*? (**T4**)

The predefined answers that the user can select are the following:

1. *Negligible*:
    - *N1*. Nothing.
    - *N2*. This would be annoying.
    - *N3*. I would be irritated.
    - *N4*. I would have to reenter or modify it/them.
2. *Limited*:
    - *L1*. This would cost me money.
    - *L2*. I would be a bit afraid or confused.
    - *L3*. I would be stressed.
    - *L4*. I would be embarrassed.
3. *Significant*:
    - *S1*. I may lose my job.
    - *S2*. This may affect my health.
    - *S3*. This would cost me lots of money.
    - *S4*. I may face legal problems.
    - *S5*. I would be humiliated.
4. *Maximum*:
    - *M1*. I would not be able to work again.
    - *M2*. I would get ruined financially.

- *M3*. My health would be damaged.
- *M4*. I could lose my life.
- *M5*. Noone would speak to me again.

The following table includes the frequency of Android permission combinations that include pairs of permissions that protect an asset and a transmission channel. The permission combinations were created from 27673 applications that were crawled from Google Play in May to June 2013.

Table C. 2: Frequency of permission combinations in Google Play (June–May 2013)

Combination		Frequency (%)
Permission1	**Permission2**	
INTERNET	ACCESS_NETWORK_STATE	81,13255
INTERNET	WRITE_EXTERNAL_STORAGE	55,142384
INTERNET	READ_EXTERNAL_STORAGE	55,08095
INTERNET	READ_PHONE_STATE	48,919483
INTERNET	ACCESS_WIFI_STATE	31,06389
INTERNET	ACCESS_COARSE_LOCATION	28,675196
INTERNET	ACCESS_FINE_LOCATION	27,753687
INTERNET	GET_ACCOUNTS	18,852993
INTERNET	CAMERA	8,188783
INTERNET	GET_TASKS	7,169702
INTERNET	READ_CONTACTS	6,942035
INTERNET	READ_HISTORY_BOOKMARKS	6,75412
INTERNET	READ_CALL_LOG	5,6663775
INTERNET	RECORD_AUDIO	4,629228
CALL_PHONE	ACCESS_NETWORK_STATE	3,798063
CALL_PHONE	WRITE_EXTERNAL_STORAGE	3,6535125
CALL_PHONE	READ_EXTERNAL_STORAGE	3,6535125
INTERNET	READ_LOGS	3,3571844
CALL_PHONE	READ_PHONE_STATE	3,3210464
CALL_PHONE	ACCESS_FINE_LOCATION	3,057242
CALL_PHONE	ACCESS_COARSE_LOCATION	2,4176064
CALL_PHONE	ACCESS_WIFI_STATE	1,9405898
CALL_PHONE	READ_CONTACTS	2,1176565
INTERNET	USE_CREDENTIALS	1,9189073
SEND_SMS	READ_PHONE_STATE	1,7526742
CALL_PHONE	READ_CALL_LOG	1,9082612
BLUETOOTH	ACCESS_NETWORK_STATE	1,7746657
INTERNET	RECEIVE_SMS	1,7237641
SEND_SMS	ACCESS_NETWORK_STATE	1,7382675
SEND_SMS	WRITE_EXTERNAL_STORAGE	1,7088678
SEND_SMS	READ_EXTERNAL_STORAGE	1,7088678
BLUETOOTH	WRITE_EXTERNAL_STORAGE	1,6486976

BLUETOOTH	READ_EXTERNAL_STORAGE	1,6486976
CALL_PHONE	GET_ACCOUNTS	1,5394622
BLUETOOTH	BLUETOOTH_ADMIN	1,586188
INTERNET	READ_SMS	1,8199103
BLUETOOTH	READ_PHONE_STATE	1,4813418
CALL_PHONE	CAMERA	1,4903599
INTERNET	BLUETOOTH_ADMIN	1,4490994
SEND_SMS	ACCESS_FINE_LOCATION	1,4295661
BLUETOOTH	ACCESS_WIFI_STATE	1,3117226
SEND_SMS	ACCESS_COARSE_LOCATION	1,2164198
SEND_SMS	READ_CONTACTS	1,4113165
INTERNET	READ_CALENDAR	1,1564035
SEND_SMS	READ_CALL_LOG	1,1775402
SEND_SMS	ACCESS_WIFI_STATE	0,94996595
SEND_SMS	RECEIVE_SMS	1,0333192
BLUETOOTH	ACCESS_FINE_LOCATION	1,0056934
CALL_PHONE	RECORD_AUDIO	0,91331154
SEND_SMS	GET_ACCOUNTS	0,8203238
CALL_PHONE	GET_TASKS	0,9312413
CALL_PHONE	RECEIVE_SMS	1,2714634
INTERNET	PROCESS_OUTGOING_CALLS	0,92181927
CALL_PHONE	READ_SMS	1,1551245
SEND_SMS	READ_SMS	0,8636029
BLUETOOTH	READ_CONTACTS	0,949887
BLUETOOTH	GET_ACCOUNTS	0,79877114
BLUETOOTH	ACCESS_COARSE_LOCATION	0,73634106
SEND_SMS	CAMERA	0,77129734
CALL_PHONE	PROCESS_OUTGOING_CALLS	0,9774302
CALL_PHONE	READ_LOGS	0,8983032
BLUETOOTH	READ_CALL_LOG	0,86114097
BLUETOOTH	RECORD_AUDIO	0,8407313
SEND_SMS	RECORD_AUDIO	0,63342315
BLUETOOTH	GET_TASKS	0,6729081
BLUETOOTH	CAMERA	0,8600728
CALL_PHONE	READ_HISTORY_BOOKMARKS	0,61784565
BLUETOOTH	READ_LOGS	0,5450894
CALL_PHONE	READ_CALENDAR	0,4841336
SEND_SMS	GET_TASKS	0,46495152
SEND_SMS	READ_LOGS	0,5532139
BLUETOOTH	RECEIVE_SMS	0,66001415
BLUETOOTH	READ_SMS	0,86868227
CALL_PHONE	BLUETOOTH_ADMIN	0,7895357
INTERNET	RECEIVE_MMS	0,67807555
CALL_PHONE	USE_CREDENTIALS	0,4548408
BLUETOOTH	PROCESS_OUTGOING_CALLS	0,7599902
SEND_SMS	PROCESS_OUTGOING_CALLS	0,67009693

SEND_SMS	RECEIVE_MMS	0,59197074
SEND_SMS	USE_CREDENTIALS	0,4151653
CALL_PHONE	RECEIVE_MMS	0,80698293
SEND_SMS	READ_CALENDAR	0,24852265
BLUETOOTH	USE_CREDENTIALS	0,3952924
BLUETOOTH	READ_HISTORY_BOOKMARKS	0,41212836
SEND_SMS	BLUETOOTH_ADMIN	0,35395098
SEND_SMS	READ_HISTORY_BOOKMARKS	0,43092525
INTERNET	READ_USER_DICTIONARY	0,30873132
BLUETOOTH	READ_CALENDAR	0,6458333
BLUETOOTH	RECEIVE_MMS	0,27013752

References

Abel W. Agents, trojans and tags: The next generation of investigators. International Review of Law, Computers & Technology 2009;23(1-2):99–108.

AlFardan NJ, Paterson KG. Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. Technical Report; 2013.

Allan A., Warden P., Got an iPhone or 3G iPad? Apple is recording your moves, http://radar.oreilly.com/2011/ 04/apple-location-tracking.html (accessed November 2012).

Amini S, Lin J, Hong J, Lindqvist J, Zhang J. Towards Scalable Evaluation of Mobile Applications through Crowdsourcing and Automation. Technical Report CMU-CyLab-12-006; Carnegie Mellon University; 2012.

Amrutkar C, Traynor P. Short paper: rethinking permissions for mobile web apps: barriers and the road ahead. In: Proc. of the second ACM workshop on security and privacy in smartphones and mobile devices. USA: ACM; 2012. p. 15–20.

Amrutkar C, Traynor P, van Oorschot P. Measuring ssl indicators on mobile browsers: Extended life, or end of the road? In: Gollmann D, Freiling F, editors. Information Security. Springer; LNCS-7483; 2012. p. 86–103.

Anderson J, Bonneau J, Stajano F. Inglorious installers: Security in the application marketplace. In: Proc. of the 9th Workshop on the Economics of Information Security (WEIS‘10); 2010.

Androulidakis I, Kandus G. A Survey on Saving Personal Data in the Mobile Phone. In Pernul G, et al., editors. In: Proc. of 6th International Conference on Availability, Reliability and Security (ARES-2011). Austria; 2011. p. 633-638.

Androulidakis I, Kandus G. Feeling Secure vs. Being Secure, The Mobile Phone User Case, In: Proc. of 7th International Conference in Global Security, Safety and Sustainability (ICGS3-2011). Greece: Springer; LNCS-99; 2012. p. 212-219.

Apple, About the security content of Safari 6, http://support.apple.com/kb/HT5400;

Aviv A, Gibson K, Mossop E, Blaze M, Smith J. Smudge attacks on smartphone touch screens. In: Proc. of the 4th USENIX conference on Offensive technologies. USENIX Association; 2010. p. 1–7.

Bäcker M. Das IT-Grundrecht: Funktion, Schutzgehalt, Auswirkungeb auf staatliche Ermittlungen. In: Uerpmann-Wittzack, editor. Das neue Computergrundrecht, Recht der Informationsgesellschaft Bd 15. Berlin: 2009. p. 53–60.

Bader M, Baggili I. iPhone 3GS forensics: Logical analysis using apple iTunes backup utility. Small Scale Digital Device Forensics Journal 2010; 4(1).

Baghdassarian S, Milanesi C. Forecast: Mobile Application Stores, World-wide, 2008-14. Technical Report G00209676; Gartner; 2010.

Bao L, Intille S. Activity recognition from user-annotated acceleration data. In: Ferscha A, Mattern F, editors. Pervasive Computing. Springer; LNCS-3001; 2004. p. 1-17.

Barrera D, Kayacik HG, van Oorschot PC, Somayaji A. A methodology for empirical analysis of permission-based security models and its application to Android. In: Proc. of the 17th ACM conference on Computer and Communications Security. ACM; 2010. p. 73–84.

Barrera D, Van Oorschot P. Secure software installation on smartphones. Security and Privacy, IEEE 2011;9(3):42-8.

Barth A, Felt AP, Saxena P, Boodman A. Protecting browsers from extension vulnerabilities. In: 17th Network and Distributed System Security Symposium. 2010.

Basagiannis S, Katsaros P, Pombortsis A. An intruder model with message inspection for model checking security protocols. Computers & Security 2010;29(1):16–34.

Becher M, Freiling FC, Hoffmann J, Holz T, Uellenbeck S, Wolf C. Mobile security catching up? revealing the nuts and bolts of the security of mobile devices. In: Proc. of the 2011 IEEE Symposium on Security and Privacy. USA: IEEE Computer Society; 2011. p. 96–111.

Beebe N, Clark J. A hierarchical, objectives-based framework for the digital investigations process. Digital Investigation 2005;2(2):147-67.

Bennett D. The Challenges Facing Computer Forensics Investigators in Obtaining Information from Mobile Devices for Use in Criminal Investigations. Information Security Journal: A Global Perspective 2012; 21(3):159–168.

Beresford A, Rice A, Skehin N, Sohan R. Mockdroid: Trading privacy for application functionality on smartphones. In: Proc. of the 12th Workshop on Mobile Computing Systems and Applications (HotMobile 2011). USA: ACM; 2011. p. 49–54.

BlackBerry, BlackBerry Security – Manuals and Guides, http://docs.blackberry.com/en/smartphone_users/subcategories/?userType=1&category=Security (accessed September 2013).

Botha R, Furnell S, Clarke N. From desktop to mobile: Examining the security experience. Computers & Security 2009; 28(3-4): 130 - 137.

Böhme R., Köpsell S. Trained to accept?: a field experiment on consent dialogs. In: Proc. of the 28th international conference on Human factors in computing systems (CHI '10). USA: ACM; 2010. p. 2403-2406.

Brown I. Regulation of converged communications surveillance. New Directions in Surveillance Privacy 2009.

BSA. 2011 BSA Global Software Piracy Study: Shadow Market. Technical Report (9th ed.); Business Software Alliance; 2012.

Bujari A, Licar B, Palazzi C. Movement pattern recognition through smartphone's accelerometer. In: Proc. of the IEEE Consumer Communications and Networking Conference (CCNC 2012). 2012. p.502-6.

Caldwell T. Smart security. Network Security 2011;2011(4):5 – 9.

Canali D, Cova M, Vigna G, Kruegel C. Prophiler: a fast filter for the large-scale detection of malicious web pages. In: Proc. of the 20th international conference on World Wide Web. ACM; 2011. p.197–206.

Carlini N, Felt AP, Wagner D. An evaluation of the Google chrome extension security architecture. In: Proc. of the 21st USENIX Conference on Security. 2012.

Carrier B, Spafford E. Getting physical with the digital investigation process. International Journal of Digital Evidence 2003;2(2):1–20.

Carrier B, Spafford E. An event-based digital forensic investigation framework. in Proc. of the Digital Forensic Research Workshop. 2004.

Casey E. Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet (2nd ed.) USA: Academic Press, Inc., 2004.

Casey E, Bann M, Doyle J. Introduction to windows mobile forensics. Digital Investigation 2010; 6(34):136-46.

CERT, Securing Your Web Browser, http://www.cert.org/tech_tips/securing_browser/ (accessed July 2013).

CERT-US, Browsing Safely: Understanding active content and cookies, http://www.us-cert.gov/ncas/tips/st04-012 (accessed July 2013).

Chen EY, Bau J, Reis C, Barth A, Jackson C. App isolation: get the security of multiple browsers with just one. In: Proc. of the 18th ACM conference on Computer and Communications security. ACM; 2011. p. 227–38.

Chia P, Yamamoto Y, Asokan N. Is this app safe? a large scale study on application permissions and risk signals. In: Proc. of the 21st International World Wide Web Conference. 2012.

Chin E, Felt AP, Greenwood K, Wagner D. Analyzing inter-application communication in Android. In: Proc. of the 9th international conference on Mobile Systems, Applications, and Services. USA: ACM; 2011. p. 239–52.

Chin E, Felt A, Sekar V, Wagner D.Measuring User Confidence in Smartphone Security and Privacy. In: Proc. of the 8th Symposium on Usable Privacy and Security. USA; ACM; 2012. p. 1-16

Choe B, Min J, Cho S. Online gesture recognition for user interface on accelerometer built -in mobile phones. In: Wong K, et al., editors. Neural Information Processing. Models and Applications. Springer; LNCS-6444; 2010. p. 650-7.

Choudhury T, Consolvo S, Harrison B, Hightower J, LaMarca A, LeGrand L, Rahimi A, Rea A, Bordello G, Hemingway B, Klasnja P, Koscher K, Landay J, Lester J, Wyatt D, Haehnel D. The mobile sensing platform: An embedded activity recognition system. IEEE Pervasive Computing 2008; 7(2):32-41.

Ciardhuáin S. An extended model of cybercrime investigations. International Journal of Digital Evidence 2004;3(1):1-22.

CISCO. 2013 Cisco Annual Security Report. Technical Report; 2013a.

CISCO. Visual Networking Index: Global Mobile Data Traffic Forecast Update, 20122017. Technical Report FLGD 10855; 2013b.

Cluley G., First iPhone worm discovered - ikee changes wallpaper to Rick Astley photo, http://nakedsecurity.sophos.com/2009/11/08/iphone-worm-discovered-wallpaper-rick-astley-photo/ (accessed November 2012).

Cohen F. Computational aspects of computer viruses. Computers & Security 1989;8(4):297–8.

Cohen J. Statistical power analysis for the behavioral sciences. Lawrence Erlbaum, 1988. Communications Committee. Broadband access in the EU: Situation at 1 July 2011. Technical Report COCOM11-24; 2011.

Coursen S. The future of mobile malware. Network Security 2007;2007(8):7-11.

Commission of the European Communities. Green paper on detection technologies in the work of law enforcement, customs and other security authorities. Technical report, COM (2006)

Community framework for electronic signatures. Technical Report; Directive 1999/93/EC; 1999.

Computer Misuse Act 1990, http://www.legislation.gov.uk/ukpga/1990/18/contents (accessed July 2013).

Cova M, Kruegel C, Vigna G. Detection and analysis of drive-by-download attacks and malicious JavaScript code. In: Proc. of the 19th international conference on world wide web. ACM; 2010. p. 281–90.

Coudert F, Gemo M, Beslay L, Andritsos F. Pervasive Monitoring: Appreciating Citizen's Surveillance as Digital Evidence in Legal Proceedings. In: Proc. of the 4th International Conference on Imaging for Crime Detection and Prevention (ICDP 2011); 2011. p. 1–6.

Cronbach L. Coefficient alpha and the internal structure of tests. Psychometrika 1951;16:297-334. Curtsinger C, Livshits B, Zorn B, Seifert C. Zozzle: Fast and precise in browser JavaScript malware detection. In: USENIX Security Symposium. 2011. p. 33–48.

Dancer F, Dampier D, Jackson J, Meghanathan N. A theoretical process model for smartphones. In: Meghanathan N, et al., editors. Springer; AISC-178; 2013. p. 279-90.

Dang Q. Recommendation for Applications Using Approved Hash Algorithms. Technical Report; 2009.

De Groef W, Devriese D, Nikiforakis N, Piessens F. Flowfox: a web browser with flexible and precise information flow control. In: Proc. of the 19th ACM conference on Computer and Communications Security. ACM; 2012. p. 748–59.

De La Vergne H, Milanesi C, Zimmermann A, Cozza R, Nguyen T H, Gupta A, Lu A. Competitive Landscape: Mobile Devices, Worldwide, 4Q09 and 2009. Technical Report G00174716; Gartner; 2010.

DFRWS . A Road Map for Digital Forensic Research. Technical Report DTR - T001-01 FINAL; First Digital Forensic Research Workshop (DFRWS); 2001.

Dietz M, Shekhar S, Pisetsky Y, Shu A, Wallach D. Quire: Lightweight provenance for smart phone operating systems. In Proc. of the 20th USENIX Security Symposium. 2011.

Distefano A, Me G. An overall assessment of mobile internal acquisition tool. Digital Investigation 2008;5, Supplement(0):p. 121-7.

Dlamini M, Eloff J, Eloff M. Information security: The moving target. Computers & Security 2009;28(3-4):189 –98.

Doumas A, Mavroudakis K, Gritzalis D, Katsikas S. Design of a neural network for recognition and classification of computer viruses. Computers & Security 1995;14(5):435–48.

Dritsas S, Dritsou V, Tsoumas B, Constantopoulos P, Gritzalis D. Ontospit: Spit management through ontologies. Computer Communications 2009;32(1):203–212.

Dritsas S, Mallios J, Theoharidou M, Marias G, Gritzalis D. Threat analysis of the session initiation protocol regarding spam. In: International Performance, Computing, and Communications Conference, 2007. IEEE. 2007. p. 426–233.

Dritsas S, Soupionis Y, Theoharidou M, Mallios Y, Gritzalis D. SPIT identification criteria implementation: Effectiveness and lessons learned. In: 23rd International Information Security Conference. Springer; 2008. p. 381–395.

Eckersley P. How unique is your web browser? In: Proc. of the 10th International Conference on Privacy Privacy Enhancing Technologies. Springer; 2010. p. 1–18.

Egele M, Kruegel C, Kirda E, Vigna G. Pios: Detecting privacy leaks in iOS applications. In: Proc. of the Network and Distributed System Security Symposium (NDSS-2011). 2011.

Egelman S, Cranor L, Hong J. You've been warned: An empirical study of the effectiveness of web browser phishing warnings. In: Proc. of the 26th Annual SIGCHI Conference on Human Factors in Computing Systems (CHI '08). USA: ACM; 2008. p. 1065–74.

Enck W, Ongtang M, McDaniel P. On lightweight mobile phone application certification. In: Proc. of the 16th ACM conference on Computer and Communications Security (CCS '09). USA: ACM; 2009. p. 235-45.

Enck W, Gilbert P, Chun B, Cox L, Jung J, McDaniel P, Sheth A. Taintdroid: An information -flow tracking system for realtime privacy monitoring on smartphones. In: Proc. of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI'10). USENIX Association; 2010. p. 1-6.

Enck W, Octeau D, McDaniel P, Chaudhuri S. A study of Android application security. In: Proc. of the 20th USENIX Conference on Security. USA: USENIX Association; 2011.

European Telecommunications Standards Institute (ETSI). Lawful Interception; Requirements of Law Enforcement Agencies. Technical Report Technical Specification 101: 331; 2009.

Fahl S, Harbach M, Muders T, Smith M, Baumgartner L, Freisleben B. Why eve and mallory love Android: an analysis of Android ssl (in)security. In: Proc. of the 2012 ACM conference on Computer and Communications Security. USA: ACM; 2012. p. 50–61.

Felt A, Chin E, Hanna S, Song D, Wagner D. Android permissions demystified. In: Proc. of the 18th ACM Conference on Computer and Communications Security (CCS '11). USA: ACM; 2011a. p. 627–38.

Felt A, Egelman S, Wagner D. Ive got 99 problems, but vibration ain't one: A survey of smartphone users concerns. In: ACM Workshop on Security and Privacy in Mobile Devices (SPSM 2012). 2012a.

Felt A, Greenwood K, Wagner D. The effectiveness of application permissions. In: 2nd USENIX Conference on Web Application Development. 2011c. p. 75–86.

Felt A, Finifter M, Chin E, Hanna S, Wagner D. A survey of mobile malware in the wild. In: Proc . of the 1st ACM Workshop on security and privacy in smartphones and mobile devices (SPSM '11). ACM; 2011b. p. 3–14.

Felt A, Ha E, Egelman S, Haney A, Chin E, Wagner D. Android permissions: User attention, comprehension, and behavior. In Proc. of the Symposium on Usable Privacy and Security (SOUPS 2012). 2012b.

Felt A, Wang H, Moshchuk A, Hanna S, Chin E. Permission re-delegation: Attacks and defenses. In: Proc. of the 20th USENIX Security Symposium, USA. 2011b.

Felt A, Wang H, Moshchuk A, Hanna S, Chin E, Greenwood K, Wagner D, Song D, Finifter M, Weinberger J, et al. Permission re-delegation: Attacks and defenses. In: Proc. of the 20th USENIX Security Symposium, USA. 2011c .

Fielding R., Gettys J., Mogul J., Frystyk H., Masinter L., Leach P., Berners-Lee T., Hypertext Transfer Protocol–HTTP/1.1, Technical Report RFC 2616, 1999.

Flick U. An introduction to qualitative research. Sage Publications, 1998.

Furnell S. Why users cannot use security. Computers & Security 2005;24(4):274-9.

Furnell S, Jusoh A, Katsabas D. The challenges of understanding and using security: A survey of end-users. Computers & Security 2006;25(1):27-35.

Furnell S. Making security usable: Are things improving? Computers & Security 2007;26(6):434-43.

Gafurov D, Helkala K, Soendrol T. Gait recognition using acceleration from MEMS. In: Proc. of the 1st International Conference on Availability, Reliability and Security (ARES 2006). 2006.

Georgiev M, Iyengar S, Jana S, Anubhai R, Boneh D, Shmatikov V. The most dangerous code in the world: validating SSL certificates in non-browser software. In: Proc. of the 2012 ACM conference on Computer and Communications Security. USA: ACM; 2012. p. 38–49.

Gershowitz AM. The iPhone meets the fourth amendment 2008

Gilbert P, Chun BG, Cox LP, Jung J. Vision: automated security validation of mobile apps at app mar - kets. In: Proc. of the 2nd International Workshop on Mobile Cloud Computing and Services (MCS'11). USA: ACM; 2011. p. 21-6.

Google, Android Security Overview, http://source.Android.com/tech/security/index.html (accessed July 2013a).

Google, Google Dashboards, http://developer.Android.com/about/dashboards/index.html (accessed July 2013b)

Google . Privacy policies for Android apps developed by third parties, https://support.google.com/googleplay/answer/2666094?hl=en (accessed July 2013c.)

Google . Refs - platform/frameworks/base - git at google, https://Android.googlesource.com/platform/frameworks/base/+refs (accessed July 2013d).

Google Developers. Safe Browsing API, https://developers.google.com/safe-browsing (accessed July 2013).

Grace M, Zhou Y, Wang Z, Jiang X. Detecting Capability Leaks in Android-based Smartphones. Technical Report; North Carolina State University; 2011.

Grace MC, Zhou W, Jiang X, Sadeghi AR. Unsafe exposure analysis of mobile in-app advertisements. In: Proc. of the 5th ACM conference on Security and Privacy in Wireless and Mobile Networks. ACM; 2012. p. 101–12.

Grier C, Tang S, King ST. Secure web browsing with the op web browser. In: IEEE Symposium on Security and Privacy (IEEE SP 2008), 2008.IEEE; 2008. p. 402–16.

Grispos G, Storer T, Glisson WB. A comparison of forensic evidence recovery techniques for a windows mobile smart phone. Digital Investigation 2011;8(1):23-36.

Gritzalis D, Secure Electronic Voting. Kluwer Academic Publishers. 2003.

Gritzalis D., Katsikas S. Towards a formal system to system authentication protocol. Computer Communications 1996;19(12):954-961.

Gritzalis D. A baseline security policy for distributed healthcare information systems. Computers & Security 1997;16(8):709–719.

Gritzalis D. Embedding privacy in it applications development. Information Management & Computer Security 2004;12(1):8–26.

Gritzalis D. Enhancing security and improving interoperability in healthcare information systems. Informatics for Health and Social Care 1998;23(4):309–323.

Gritzalis D., A digital seal solution for deploying trust on commercial transactions. In: Information Management & Computer Security. 9(2), 71-79, March 2001.

Gritzalis D, Theoharidou M, Kalimeri E, Towards an interdisciplinary information security education model. In: Proc. of the 4th World Conference on Information Security Education. Moscow, 2005. p.22-35

Gritzalis D, Mallios Y. A SIP-oriented SPIT management framework. Computers & Security 2008;27(5):136–153.

Gritzalis D, Soupionis Y, Katos V, Psaroudakis I, Katsaros P, Mentis A. The sphinx enigma in critical voip infrastructures: Human or botnet? In: 4th International Conference on Information, Intelligence, Systems and Applications. IEEE Press; 2013. .

Grobler C, Louwrens C, von Solms S. A multi-component view of digital forensics. In: International Conference on Availability, Reliability, and Security, 2010. IEEE; 2010. p. 647–52.

Gupta A, Milanesi C, Cozza R, Lu C. Market Share Analysis: Mobile Phones, Worldwide, 2Q13. Technical Report G00253753; Gartner; 2013.

Harrell F, Lee K, Califf R, Pryor D, Rosati R. Regression modeling strategies for improved prognostic prediction. Statistics in Medicine 1984;3(2):143-52.

Hoffmann-Riem. W. Freiheit und Sicherheit im Angesicht terroristischer Anschläge. Zeitschrift für Rechtspolitik 2002;35(497).

Hogben G, Dekker M. Smartphones: Information security risks, opportunities and recommendations for users. Technical Report; ENISA; 2010.

Hoog A. Android forensic techniques. In: Android Forensics. Boston: Syngress; 2011. p. 195-284.

Hornyack P, Han S, Jung J, Schechter S, Wetherall D. These aren't the droids you're looking for: Retrofitting Android to protect data from imperious applications. In: Proc. of the 18th ACM Conference on Computer and Communications Security (CCS '11). USA: ACM; 2011. p. 639-52.

Hosmer D, Lemeshow S. Applied logistic regression. Volume 354. Wiley-Interscience, 2000.

Howell J, Schechter S. What you see is what they get. In: Proc. of the IEEE Workshop on Web 2.0 Security and Privacy, 2010.

Husain MI, Sridhar R. iForensics: Forensic analysis of instant messaging on smart phones. In: Goel S, et al., editors. Digital Forensics and Cyber Crime. Springer; LNICST-31; 2010. p. 9-18.

Hypponen M. Malware goes mobile. Scientific American 2006;295(5):70–7.

ICO. Privacy impact assessment handbook. V2.0, Information Commissioner's Office, United Kingdom.

Ieong R. Forza A. Digital forensics investigation framework that incorporate legal issues. Digital In vestigation 2006;3, Supplement(0):29-36.

Ikonomopoulos S, Lambrinoudakis C, Gritzalis D, Kokolakis S, Vassiliou K. Functional requirements for a secure electronic voting system. In: 17th International Information Security Conference. Springer; AICT-86; 2002. p. 507–520.

Iliadis J, Spinellis D, Gritzalis D, Preneel B, Katsikas S. Evaluating certificate status information mechanisms. In: 7th ACM Conference on Computer and Communications Security. USA: ACM; 2000. p. 1–8.

ISO. Information technology Security techniques - Information security risk management. Technical Report ISO/IEC 27005:2008.

Jang D, Jhala R, Lerner S, Shacham H. An empirical study of privacy-violating information flows in JavaScript web applications. In: Proc. of the 17th ACM conference on Computer and Communications Security. ACM; 2010. p. 270–83.

Jang D, Tatlock Z, Lerner S. Establishing browser security guarantees through formal shim verification. In: Proc. of the 21st USENIX conference on security symposium. USENIX Association; 2012.

Jansen W, Ayers R. Guidelines on cell phone forensics. Technical Report NIST SP 800-101; National Institute of Standards and Technology; 2007.

Jeon W, Kim J, Lee Y, Won D. A practical analysis of smartphone security. In: Proc. of the 2011 international conference on human interface and the management of information. Springer-Verlag; 2011. p. 311–20.

Jiang X. An evaluation of the application ("app") verification service in Android 4.2, http://www.cs.ncsu.edu/faculty/jiang/appverify/ (accessed July 2013).

Johns M. On JavaScript malware and related threats. Journal in Computer Virology 2008;4(3):161–78.
Jung J, Jeong C, Byun K, Lee S. Sensitive privacy data acquisition in the iPhone for digital forensic analysis. In: Park J, et al., editors. Secure and Trust Computing, Data Management and Applications. Springer; CCIS-186; 2011. p. 172-86.

Kandias M, Galbogini K, Mitrou L, Gritzalis D. Insiders trapped in the mirror reveal themselves in social media. In 7th International Conference on Network and System Security. Springer; LNCS 7873; 2013a. pp. 26-37

Kandias M, Mitrou L, Stavrou V, Gritzalis D. Which side are you on? a new panopticon vs. privacy. In: 10th International Conference on Security and Cryptography. 2013b. p. 98–110.

Kandias M, Stavrou V, Bosovic N, Gritzalis D. Proactive insider threat detection through social media: The youtube case. In: 12th Workshop on Privacy in the Electronic Society, Berlin. 2013c. .

Katsikas S, Gritzalis D. The need for a security policy in health care institutions. International Journal of Biomedical Computing 1994;35:

Kelley P, Consolvo S, Cranor L, Jung J, Sadeh N, Wetherall D. A conundrum of permissions: Installing applications on an Android smartphone. In Proc. of the Workshop on Usable Security (USEC). 2012.

Kern N, Schiele B, Junker H, Lukowicz P, Trster G. Wearable sensing to annotate meeting recordings. Personal and Ubiquitous Computing 2003;7:263-74.

Kerr O.S. Searches and Seizures in a Digital World. Harvard Law Review 2005; 119:531–586.

Khan A, Lee YK, Lee S, Kim TS. A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer. IEEE Transactions on Information Technology in Biomedicine 2010;14(5):1166-72.

Klaver C. Windows mobile advanced forensics. Digital Investigation 2010;6(34):147-67.

Kohn M, Eloff J, Olivier M. Framework for a digital forensic investigation. In Proc. of the ISSA from Insight to Foresight Conference, Sandton, South. 2006.

Kokolakis S, Gritzalis D, Katsikas S. Generic security policies for healthcare information systems. Health Informatics Journal 1998;4(3-4):184–195.

Kolbitsch C, Livshits B, Zorn B, Seifert C. Rozzle: De-cloaking internet malware. In: IEEE Symposium on Security and Privacy, 2012. IEEE; 2012. p. 443–57.

Kotzanikolaou P, Theoharidou M, Gritzalis D. Assessing n–order dependencies between critical infrastructures. International Journal of Critical Infrastructures 2013a;9(1):93–110.

Kotzanikolaou P, Theoharidou M, Gritzalis D. Cascading effects of commoncause failures on critical infrastructures. In: Proc. of the 7th IFIP International Conference on Critical Infrastructure Protection. Springer, USA.2013b.

Kuntze N, Rudolph C, Alva A, Endicott-Popovsky B, Christiansen J, Kemmerich T. On the creation of Reliable Digital Evidence. In: Peterson G, et al. editors, Advances in Digital Forensics. Springer; AICT-383; 2012. p. 3–17.

Kwapisz J, Weiss G, Moore S. Activity recognition using cell phone accelerometers. SIGKDD Explor Newsl 2011;12(2):74-82.

Lambrinoudakis C, Gritzalis D, Tsoumas V, Karyda M, Ikonomopoulos S. Secure electronic voting: The current landscape. In: Gritzalis D, editor. Secure Electronic Voting. Springer; AIS-7; 2003. p.101–122.

Lambrinoudakis C, Kokolakis S, Karyda M, Tsoumas V, Gritzalis D, Katsikas S. Electronic voting systems: Security implications of the administrative workflow. In: 14th International Workshop on Database and Expert Systems Applications, 2003b. p. 467–471.

Lane N, Miluzzo E, Lu H, Peebles D, Choudhury T, Campbell A. A survey of mobile phone sensing. IEEE Communications Magazine 2010;48(9):140–50.

Lederm T, Clarke N. Risk assessment for mobile devices. In: Trust, Privacy and Security in Digital Business. Springer; LNCS-6863; 2011. p. 210–21.

Lekkas D, Gritzalis D. Cumulative notarization for long-term preservation of digital signatures. Computers & Security 2004;23(5):413–424.

Lekkas D, Gritzalis D. Long-term verifiability of the electronic healthcare records authenticity. International Journal of Medical Informatics; 2007;76(56):442–448.

Lekkas D, Gritzalis D. e-passports as a means towards the first world-wide Public Key Infrastructure. In: 4th European PKI Workshop. Springer; LNCS-4582; 2007. p. 34–48.

Lessard J, Kessler G. Android Forensics: Simplifying Cell Phone Examinations. Technical Report; Purdue University; 2010.

Lester J, Choudhury T, Borriello G. A practical approach to recognizing physical activities. In: Fishkin K, et al., editors. Pervasive Computing. Springer; LNCS-3968; 2006. p. 1-16.

Lin J, Sadeh N, Amini S, Lindqvist J, Hong JI, Zhang J. Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing. In: Proc. of the 2012 ACM Conference on Ubiquitous Computing. ACM; 2012. p. 501–10.

Liu J, Zhong L, Wickramasuriya J, Vasudevan V. Uwave: Accelerometer-based personalized gesture recognition and its applications. Pervasive and Mobile Computing 2009;5(6):657-75.

Long X, Yin B, Aarts R. Single-accelerometer-based daily physical activity classification. In: Proc. of the Annual Engineering International Conference of the IEEE in Medicine and Biology Society (EMBC-2009). 2009. p. 6107-10.

Lookout. Security alert: Droiddream malware found in official Android market, http://blog.mylookout.com/blog/2011/03/01/security-alert-malware-found-in-official-Android-market-droiddream/ (assessed April 2012)

Madrigal, I'm Being Followed: How Google—and 104 Other Companies—Are Tracking Me on the Web, http://www.theatlantic.com/technology/archive/2012/02/im-being-followed-how-google-151-and-104-other-companies-151-are-tracking-me-on-the-web/253758/ (accessed July 2013)

Mantyjarvi J, Lindholm M, Vildjiounaite E, Makela SM, Ailisto H. Identifying users of portable devices from gait pattern with accelerometers. In Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '05), 2005. p. 973-6.

Marias G, Dritsas S, Theoharidou M, Mallios J, Gritzalis D. SIP vulnerabilities and anti-spit mechanisms assessment. In: 16th IEEE International Conference on Computer Communications and Networks, 2007.IEEE; 2007. p. 597–604.

Marinos L., Sfakianakis A., ENISA Threat Landscape - Responding to the Evolving Threat Environment, Technical Report, 2012.

Marmol F, Perez G, Security threats scenarios in trust and reputation models for distributed systems. Computers & Security 2009; 28(7):545-556.

Marquardt P, Verma A, Carter H, Traynor P. (sp)iphone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In Proc. of the 18th ACM Conference on Computer and Communications Security (CCS '11). USA: ACM; 2011. p. 551–62.

McClure S, Scambray J, Kurtz G, Kurtz. Hacking exposed: network security secrets & solutions. McGraw-Hill/Osborne New York, 2005.

McDaniel P, Enck W. Not so great expectations: Why application markets haven't failed security. IEEE Security and Privacy (SP) 2010;8(5):76-8.

McKnight D.H., Chervany N.L. The meanings of trust. Technical Report WP9604; University of Minnesota Information Systems Research Center; 1996.

Meyerovich LA, Livshits B. Conscript: Specifying and enforcing fine-grained security policies for JavaScript in the browser. In: IEEE Symposium on Security and Privacy, IEEE; 2010. p. 481–96.

Microsoft. SmartScreen Filter: frequently asked questions, http://windows.microsoft.com/ en-us/windows7/smartscreen-filter-frequently-asked-questions-ie9 (accessed July 2013)

Microsoft. Windows Phone 7 Application Certification Requirements, http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh184843%28v=vs.105%29.aspx (accessed July 2013)

Mislan RP, Casey E, Kessler GC. The growing need for on-scene triage of mobile devices. Digital Investigation 2010;6(3):112–24.

Mitrou L, Gritzalis D, Katsikas S. Revisiting legal and regulatory requirements for secure e-voting. In: 17th International Information Security Conference. Springer; AICT-86; 2002. p. 469–480.

Mokhonoana P, Olivier M. Acquisition of a Symbian smart phones content with an on-phone forensic tool. Technical Report; 2007.

Morrissey S. iOS forensic analysis for iPhone, iPad and iPod touch. New York: Apress; 2010

Motiee S, Hawkey K, Beznosov K. Do windows users follow the principle of least privilege?: Investigating user account control practices. In: Proc. of the 6th Symposium on Usable Privacy and Security. USA: ACM; 2010. p. 1-13.

Mozilla. Plugin Check & Updates, https://www.mozilla.org/en-US/plugincheck/ (accessed July 2013)

MSDN, Windows Mobile Device Security Model, http://msdn.microsoft.com/en-us/library/bb416353.aspx (accessed September 2013).

MSDN, Security for Windows Phone, http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff402533%28v=vs.105%29.aspx (accessed September 2013).

Mukasey M, Sedgwick J, Hagy D. Electronic Crime Scene Investigation: A Guide for First Responders. Technical Report NCJ 219941; National Institute of Justice; 2008.

Mutawa N, Baggili I, Marrington A. Forensic analysis of social networking applications on mobile devices. Digital Investigation 2012;9, Supplement (0):S24-33.

Nachenberg C. A Window Into Mobile Device Security. Technical Report; Symantec; 2011.

Nadji Y, Giffin J, Traynor P. Automated remote repair for mobile malware. In: Proc. of the 27th Annual Computer Security Applications Conference (ACSAC '11). USA: ACM; 2011. p. 413-22.

Nagelkerke N. A note on a general definition of the coefficient of determination. Biometrika 1991;78(3):691–2.

Nauman M, Khan S, Zhang X. Apex: extending Android permission model and enforcement with user-defined runtime constraints. In: Proc. of the 5th ACM Symposium on Information, Computer and Communications Security (CCS '10). USA: ACM; 2010. p. 328–32.

National Cyber Security Alliance - StaySafeOnline.org, http://www.staysafeonline.org/ (accessed July 2013)

Network Computing. Certificate Authority Compromises Are Global In Reach, http://www.networkcomputing.com/security/certificate-authority-compromises-are gl/231601123 (accessed July 2013)

Niemela J. Just because it's signed doesn't mean it isn't spying on you, www.f-secure.com/weblog/archives/00001190.html (accessed April 2012).

Nokia, Platform Security, http://developer.nokia.com/Community/Wiki/Platform_Security (accessed September 2013)

Opera. Opera's Fraud and Malware Protection, http://www.opera.com/help/tutorials/security/fraud/(accessed July 2013)

Oppliger R. Security and privacy in an online world. Computer 2011;44(9):21–22.

OWASP. Top 10 mobile risks. V1.0, http://www.owasp.org/index.php/OWASP_Mobile_Security_Project (accessed July 2013)

Panday K. Taming the Beast (Browser Exploit Against SSL/TLS). Technical Report; Microsoft; 2011.
Park J, Chung H, Lee S. Forensic analysis techniques for fragmented flash memory pages in smart phones. Digital Investigation 2012.

Paterson. On the Security of RC4 in TLS, http://www.isg.rhul.ac.uk/tls/ (accessed July 2013)

Paterson KG, Ristenpart T, Shrimpton T. Tag size does matter: Attacks and proofs for the TLS record protocol. In: Proc. of the 17th international conference on the theory and application of cryptology and information security. Springer; 2011. p. 372–89.

Polemi D, Ntouskas T, Georgakakis E, Douligeris C, Theoharidou M, Gritzalis D. S-port: Collaborative security management of port information systems. In: 4th International Conference on Information, Intelligence, Systems and Applications. IEEE Press; 2013. .

Pearce P, Felt AP, Nunez G, Wagner D. Addroid: Privilege separation for applications and advertisers in Android. In: Proc. of the 7th ACM Symposium on Information, Computer and Communications Security. ACM;2012. p. 71–2.

Pooters I. Full user data acquisition from Symbian smart phones. Digital Investigation 2010;6(34):125 - 35.

Qualys. Qualys BrowserCheck, https://browsercheck.qualys.com (accessed July 2013)

Quirolgico S, Voas J, Kuhn R. Vetting mobile apps. IT Professional 2011;13:9–11.

Ramabhadran A. Forensic investigation process model for Windows Mobile devices. Technical Report; Security Group - Tata Elxsi; 2009.

Ravi N, Dandekar N, Mysore P, Littman M. Activity recognition from accelerometer data. In: Proc. of the National Conference on Artificial Intelligence. MIT Press; 2005.

Redman P, Girard J, Wallin L. Magic Quadrant for Mobile Device Management Software. Technical Report G00211101; Gartner; 2011.

Rehault F. Windows Mobile advanced forensics: An alternative to existing tools. Digital Investigation 2010;7(1-2):38-47.

Reid M. United States v. Jones: Big Brother and the "Common Good" Versus the Fourth Amendment and your Right to Privacy. 2012. Available from: http://works.bepress.com/melanie_reid/9.

Reith M, Carr C, Gunsch G. An examination of digital forensic models. International Journal of Digital Evidence 2002;1(3):1-12.

Rogers M, Goldman J, Mislan R, Wedge T, Debrota S. Computer forensics field triage process model. In: Proc. of the Conference on Digital Forensics Security and Law. 2006. p. 27-40.

Rose C. Ubiquitous Smartphones, Zero Privacy. Review of Business Information Systems 2012; 16(4):187–192.

Rushin S. The Judicial Response to Mass Police Surveillance. Journal of Law, Technology & Policy 2011; 2011(2):282–328.

Salgado R.P. Fourth Amendment Search and the Power of the Hash. Harvard Law Review 2005; 119:38–46.

Sarma BP, Li N, Gates C, Potharaju R, Nita-Rotaru C, Molloy I. Android permissions: a perspective combining risks and benefits. In: Proc. of the 17th ACM symposium on access control models and technologies. ACM; 2012. p. 13–22.

Saxena P, Akhawe D, Hanna S, Mao F, McCamant S, Song D. A symbolic execution framework for JavaScript. In: IEEE symposium on security and privacy (SP), 2010. IEEE; 2010. p. 513–28.

Schechter S, Dhamija R, Ozment A, Fischer I. The emperor's new security indicators. In: IEEE symposium on security and privacy, IEEE; 2007. p. 51 –65.

Schlegel R, Zhang K, Zhou X, Intwala M, Kapadia A, Wang X. Soundcomber: A stealthy and context - aware sound trojan for smartphones. In: Proc. of the 18th Annual Network and Distributed System Security Symposium (NDSS 2011). 2011. p. 17-33.

SERT. Quarterly Threat Intelligence Report Q4 2012. Technical Report; 2013

Shabtai A, Fledel Y, Kanonov U, Elovici Y, Dolev S, Glezer C. Google Android: A comprehensive security assessment. IEEE Security Privacy 2010;8(2):35–44.

Shekhar S, Dietz M, Wallach D. AdSplit: Separating smartphone advertising from applications. Technical Report; 2012.

Shin D, Lopes R. An empirical study of visual security cues to prevent the sslstripping attack. In: Proc. of the 27th annual computer security applications conference. ACM; 2011. p. 287–96.

Soupionis Y, Dritsas S, Gritzalis D. An adaptive policy-based approach to SPIT management. In: 13th European Symposium on Research in Computer Security. Springer; LNCS-5283; 2008. p. 446–460.

Soupionis Y, Gritzalis D. Audio CAPTCHA: Existing solutions assessment and a new implementation for VoIP telephony. Computers & Security 2010;29(5):603–618.

Spinellis D., Gritzalis S., Iliadis J., Gritzalis D., Katsikas S. Trusted Third Party services for deploying secure telemedical applications over the WWW. Computers & Security. 18(7), 627-639, 1999.

Souppaya M, Scarfone K. Guidelines for Managing the Security of Mobile Devices in the Enterprise. NIST; 2013. NIST Special Publication 800-124, revision 1.

Spirakis P, Katsikas S, Gritzalis D, Allegre F, Darzentas J, Gigante C, Karagiannis D, Kess P, Putkonen H, Spyrou T. Securenet: A network-oriented intelligent intrusion prevention and detection system. Network Security Journal 1994;1(1).

StatCounter. StatCounter Global Stats, http://gs.statcounter.com (accessed July 2013)

Stergiopoulos G., Tsoumas B., Gritzalis D. On Business Logic Vulnerabilities Hunting: The APP_LogGIC Framework. In: Proc. of the Network and System Security, pp. 236--249, Springer 2013.

Stevens R, Gibler C, Crussell J, Erickson J, Chen H. Investigating user privacy in Android ad libraries. In: Workshop on Mobile Security Technologies. 2012.

Sunshine J, Egelman S, Almuhimedi H, Atri N, Cranor LF. Crying wolf: an empirical study of SSL warning effectiveness. In: Proc. of the 18th conference on USENIX security symposium., USA: USENIX Association; 2009. p. 399–416.

Sutherland I, Evans J, Tryfonas T, Blyth A. Acquiring volatile operating system data tools and techniques. ACM SIGOPS Operating Systems Review 2008;42(3):65–73.

Symbian Freak, HOT: Nokia's S60 3rd Ed security has been HACKED?, http://www.symbian-freak.com//news/008/03/s60_3rd_ed_has_been_hacked.htm (accessed September 2013)

Sylve J, Case A, Marziale L, Richard G. Acquisition and analysis of volatile memory from Android devices. Digital Investigation 2012;8(34):175-84.

Tan J. Forensic readiness. Cambridge, USA:@ Stake 2001;

Theoharidou M, Gritzalis D. A Common Body of Knowledge for Information Security. IEEE Security & Privacy 2007;5(2):64-67.

Theoharidou M, Kotzanikolaou P, Gritzalis D. Risk-based criticality analysis. In: Critical Infrastructure Protection III. Springer; AICT-311; 2009. p. 35–49.

Theoharidou M, Kotzanikolaou P, Gritzalis D. A multi-layer criticality assessment methodology based on interdependencies. Computers & Security 2010;29(6):643–658.

Theoharidou M, Kotzanikolaou P, Gritzalis D. Risk assessment methodology for interdependent Critical Infrastructures. International Journal of Risk Assessment and Management. 2011;15(2-3):128-148.

Theoharidou M, Papanikolaou N, Pearson S, Gritzalis D. Privacy risks, security and accountability in the cloud. In: 5th IEEE Conference on Cloud Computing Technology and Science. IEEE Press; 2013a.

Theoharidou M, Tsalis N, Gritzalis D. In cloud we trust: Risk-assessment as-a-service. In: Trust Management VII. Springer; 2013b. p. 100–10.

Thing V, Chua TW. Symbian smartphone forensics: Linear bitwise data acquisition and fragmentation analysis. In: Kim T, et al., editors. Computer Applications for Security, Control and System Engineering. Springer; CCIS; 2012. p. 62–9.

Thing VL, Ng KY, Chang EC. Live memory forensics of mobile phones. Digital Investigation 2010;7, Supplement(0):S74 –82.

Thing V, Tan D. Symbian smartphone forensics and security: Recovery of privacy-protected deleted data. In: Chim T, Yuen T, editors. Information and Communications Security. Springer; LNCS-7618; 2012. p. 240-51.

Thomson M, von Solms R. Information security awareness: Educating your users effectively. Information Management & Computer Security 1998;6(4):167 -73.

Togias S. The right in confidentiality and integrity of information technology systems according to the German Federal Constitutional Court: old wine in new bottles? In: Bottis M, editor. An Information Law for the 21st Century. Athens: 2010. p. 530–540.

Tryfonas T, Gritzalis D, Kokolakis S. A qualitative approach to information availability. In: 15th Annual Working Conference on Information Security for Global Information Infrastructures. Springer; 2000. p. 37–47.

Tsalis N, Theoharidou M, Gritzalis D. Return on security investment for cloud platforms. In: Economics of Security in the Cloud Workshop. IEEE Press; 2013. .

Tsoumas B, Dritsas S, Gritzalis D. An ontology-based approach to information systems security management. In: 3rd International Conference on Mathematical Models, Methods and Architectures for Computer Network Security. Springer; 2005. p. 151–164.

Veracode, Browser Security Settings for Chrome, Firefox and Internet Explorer: Cybersecurity 101, http://www.veracode.com/blog/2013/03/browser-security-settings-for-chrome-firefox-and-internet-explorer (accessed July 2013).

Vidas T, Zhang C, Christin N. Toward a general collection methodology for Android devices. Digital Investigation 2011;8 (Supplement 1):S14–24.

Virvilis N, Gritzalis D. The big four–what we did wrong in advanced persistent threat detection? In: 8th International Conference on Availability, Reliability and Security. IEEE; 2013. p. 248–54.

Walls R, Learned-Miller E, Levine B. Forensic triage for mobile phones with DEC0DE. In: Proc. of USENIX Security Symposium. 2011.

Wang HJ, Grier C, Moshchuk A, King ST, Choudhury P, Venter H. The multi-principal OS construction of the gazelle web browser. In: USENIX security symposium. 2009. p. 417–32.

Wang Y, Zheng J, Sun C, Mukkamala S. Quantitative security risk assessment of Android permissions and applications. In: Data and Applications Security and Privacy XXVII. Springer; LNCS-7964; 2013. p. 226–41.

Warren A, Bayley R, Bennett C, Charlesworth A, Clarke R, Oppenheim C. Privacy Impact Assessments: International experience as a basis for UK Guidance. Computer Law & Security Review 2008;24(3):233 –42.

Weise J, Powell B. Using computer forensics when investigating system attacks. Technical Report 819-2262-10; Sun Client Solutions Security Expertise Center; 2005.

Whitten A, Tygar J. Why johnny can't encrypt: A usability evaluation of PGP 5.0. In: Proc. of the 8th USENIX Security Symposium. USA; 1999.

Wiebe A. The new Fundamental Right to IT Security – First evaluation and comparative view at the U.S. Datenschutz und Datensicherheit 2008; 32(11):713–716.

Wilkinson S, Haagman D. Good Practice Guide for Computer-Based Electronic Evidence. Technical Report v. 4; UK Association of Chief Police Officers; 2007.

Wired, IE 10′s 'Do-Not-Track' Default Dies Quick Death, http://www.wired.com/threatlevel/2012/06/default-do-not-track/ (accessed July 2013).

Xu N, Zhang F, Luo Y, Jia W, Xuan D, Teng J. Stealthy video capturer: A new video-based spyware in 3G smartphones. In: Proc. of the 2nd ACM Conference on Wireless Network Security. USA: ACM; 2009. p. 69-78.

Yu X, Jiang LH, Shu H, Yin Q, Liu TM. A process model for forensic analysis of Symbian smart phones. In: Advances in Software Engineering. Springer; CCIS-59; 2009. p. 86-93.

Zachman J. A framework for information systems architecture. IBM Systems Journal 1987; 26(3): 276–92.

Zdziarski J. iPhone forensics: Recovering evidence, personal data, and corporate assets. O'Reilly Media, 2008.

Zeigler A., Bateman A., Graff E., Web tracking protection, http://www.w3.org/Submission/web - tracking-protection/ (accessed July 2013)

Zhou Y, Jiang X. Dissecting Android malware: Characterization and evolution. In: Proc. of the IEEE Symposium on Security and Privacy (SP). IEEE, 2012, pp. 95-109.

Zhou Y, Zhang X, Jiang X, Freeh V. Taming information-stealing smartphone applications (on Android). In: McCune J, et al., editors. Trust and Trustworthy Computing. Springer; LNCS-6740; 2011. p. 93–107.

Zhou W, Zhou Y, Jiang X, Ning P. Detecting repackaged smartphone applications in third-party Android marketplaces. In: Proc. of 2nd ACM Conference on Data and Application Security and Privacy (CODASPY-2012). ACM, 2012a, pp. 317-326.

Zhou Y, Wang Z, Zhou W, Jiang X. Hey, you, get off of my market: Detecting malicious apps in official and alternative Android markets. In: Proc. of the 19th Network and Distributed System Security Symposium (NDSS '12). USA; 2012b.