Universidade Estadual de Campinas
Instituto de Computação

Daniel Avila Vecchiato

Benchmarking User-Defined Security Configurations of Mobile Devices

Testes padronizados para configurações de segurança em dispositivos móveis

CAMPINAS
2016

Daniel Avila Vecchiato

Benchmarking User-Defined Security Configurations of Mobile Devices

Testes padronizados para configurações de segurança em dispositivos móveis

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Dissertation presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Supervisor/Orientadora: Profa. Dra. Eliane Martins
Co-supervisor/Coorientador: Prof. Dr. Marco Vieira (Universidade de Coimbra)

Este exemplar corresponde à versão da Tese entregue à banca antes da defesa.

CAMPINAS
2016

Na versão final, esta página será substituída pela ficha catalográfica.

De acordo com o padrão da CCPG: "Quando se tratar de Teses e Dissertações financiadas por agências de fomento, os beneficiados deverão fazer referência ao apoio recebido e inserir esta informação na ficha catalográfica, além do nome da agência, o número do processo pelo qual recebeu o auxílio."
e
"caso a tese de doutorado seja feita em Cotutela, será necessário informar na ficha catalográfica o fato, a Universidade convenente, o país e o nome do orientador."

Na versão final, esta página será substituída por outra informando a composição da banca e que a ata de defesa está arquivada pela Unicamp.

Dedicatória

Aos meus pais. À Patrycia.

We can only see a short distance ahead, but we can see plenty there that needs to be done.

(Alan Turing)

Agradecimentos

Agradeço à Deus, à minha família e à Patrycia, que foram fontes de inspiração e força para a dedicação empregada neste trabalho.

Aos meus orientadores, Professora Doutora Eliane Martins e Professor Doutor Marco Vieira, gostaria de expressar meus sinceros agradecimentos pelo apoio empregado durante todo o Doutorado.

Ao Instituto de Computação da Unicamp e da UFMT, à Universidade de Coimbra e à Capes quero agradecer por todo o apoio institucional e financeiro prestado a este projeto.

Aos meus amigos Maria Angélica, Thiago Genez e Ítalo agradeço por todo o apoio e pelas frutíferas troca de ideias sobre o tema do meu trabalho.

Aos membros do Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra por todo apoio prestado durante minha estada na cidade de Coimbra durante o período de doutorado sanduíche.

Resumo

A ampla disseminação de dispositivos móveis, como smartphones e tablets, e a sua vasta capacidade de uso, que vai desde tirar fotos ao acesso a contas bancárias, torna-os um alvo atraente para os atacantes. Isso, juntamente com o fato de que os usuários frequentemente armazenam informações pessoais em tais dispositivos, e que muitas organizações atualmente implementam a política "*Bring Your Own Device*" (BYOD), que permite aos funcionários usarem seus dispositivos pessoais para acessarem a infra-estrutura de informação corporativa e aplicações, tornando a avaliação da segurança de dispositivos móveis uma questão fundamental.

Muitos problemas de segurança dependem de como o sistema está configurado e como ele é utilizado, além das características do ambiente em que o sistema está inserido. Em particular, as configurações de um sistema são conhecidas por serem uma importante fonte de vulnerabilidades de segurança. Neste trabalho, *configuração de segurança definida pelo usuário* é entendida como qualquer configuração ou valor que os usuários podem definir e que têm algum impacto na segurança global do dispositivo, por exemplo, definir uma senha de desbloqueio, definir um tempo para bloqueio após um tempo de inatividade. Aumentado, dessa forma, a segurança, impedindo que invasores facilmente obtenham acesso ao dispositivo.

Este trabalho apresenta uma abordagem de testes padronizados para avaliar as configurações de segurança definidas pelos usuários, considerando o risco, que cada configuração possui, de prejudicar o dono do dispositvo. Esta abordagem fornece informações valiosas para aqueles que precisam avaliar, comparar ou controlar as configurações de segurança em dispositivos móveis. Na prática, para cada configuração definida pelo usuário, uma intensidade de risco é calculada com base na análise da percepção de um conjunto de especialistas de segurança. A intensidade do risco das diferentes configurações são agregadas para avaliar a segurança geral, com base na análise das configurações de um determinado dispositivo.

Em suma, as principais contribuições deste trabalho são: 1) uma ferramenta que permite avaliar a segurança de dispositivos Android com base nas configurações definidas pelo usuário; 2) uma análise das configurações de segurança definidas pelo usuário de dispositivos Android, a fim de entender os problemas mais comuns relacionados às configurações de segurança; 3) uma abordagem de análise de risco para qualificar / quantificar a segurança de dispositivos móveis tomando como base as configurações de segurança definidas pelo usuário; e 4) a definição de testes padronizados *benchmark* de configuração de segurança para dispositivos móveis, usando o Android como estudo de caso.

A plataforma Android é amplamente usada e representativa com relação ao estado da arte em computação móvel. Na verdade, o Android tem uma quota de mercado de sistemas operacionais móveis de mais de 80%. Desta forma, embora a maioria do nosso trabalho pode ser adaptado para diferentes plataformas, o Android foi usado como referência nos estudos de caso apresentados nesta tese.

Os resultados, com base na análise de dados coletados de 561 dispositivos, mostram que os usuários do sistema Android negligenciam importantes recomendações de segurança ao configurarem seus dispositivos e que o processo de *benchmarking* é realmente uma boa maneira de identificar a configuração mais segura definida pelo usuário.

Abstract

The wide spreading of mobile devices, such as smartphones and tablets, and their always-advancing capabilities, ranging from taking photos to accessing banking accounts, makes them an attractive target for attackers. This, together with the fact that users frequently store critical personal information in such devices and that many organizations currently implement a "*bring your own device (BYOD)*" policy that allows employees to use their personal devices to access the enterprise information infrastructure and applications, makes the assessment of the security of mobile devices a key issue.

Many security issues depend on the way the system is configured and used, and on the characteristics of the surrounding environment. In particular, system configurations are known to be a key source of security vulnerabilities. In this work, a *user-defined security configuration* is understood as any setting, configuration or value that users can define or set and that have some impact in the overall security of the device, for instance, setting a password to unlock the device or defining a timeout to automatically lock the device after a period of user inactivity. Thus, enhancing the device security by preventing attackers from easily getting access to the device.

This work presents an approach to benchmark the user-defined security configurations of mobile devices considering the risk that each configuration has to harm or cause any type of loss to the device owner. This approach provides valuable information for those who need to evaluate, assess, compare or control the security configurations of mobile devices. In practice, for each user-defined configuration, an intensity of severity is calculated based on the analysis of the perception of a set of security experts. The intensity of severity of the different configurations are aggregated to assess overall security, based on the analysis of the concrete settings of a given device.

In short, the main contributions of this work are: 1) a tool that allows assessing the security of Android devices based on the user-defined configurations; 2) a security analysis of the user-defined security configurations of Android devices in order to understand the common misconfiguration problems; 3) a risk analysis approach to qualify/quantify the security of mobile devices concerning the user-defined security configurations; and 4) a security configuration benchmark for mobile devices, using Android as case study.

The Android platform is widely use and representative with respect to the state-of-the-art in mobile computing. In fact, Android has a global mobile OS market share of more than 80%. This way, although most of our work can be adapted to different platforms, Android has been used as reference in the case studies presented in this thesis.

The results presented, based on the analysis of data collected from 561 devices, show that Android users neglect important security recommendations while configuring their devices and that benchmarking is indeed a practical way to assess and compare the security of mobile devices with regard to user-defined configurations. In fact, the results can be used by both manufacturers and users to enhance the security level of their devices.

List of Figures

3.1 High Level Architecture of uSEA. 43
3.2 Manufacturers share. 46
3.3 Android platform versions. 47
3.4 Analysis of users safe security configurations number. 47
3.5 Analysis of the configurations of all categories. 48
3.6 Analysis of the configuration in the application category. 49
3.7 Analysis of the configuration in the browser category. 49
3.8 Analysis of the configuration in the content category. 50
3.9 Analysis of the configuration in the network category. 51
3.10 Analysis of the configuration in the password category. 52
3.11 Analysis of the configuration in the permission category. 53
3.12 Analysis of the configuration in the system category. 54
3.13 Analysis of devices on the field that are vulnerable to known vulnerabilities and exposures. 61

4.1 Overview of the process to define the security configurations severity level. 66
4.2 Process to support the analysis by the experts. 68
4.3 Ranking graph of the four exemplified configurations. 70
4.4 High level view of the risk analysis process. 74
4.5 Evaluation of the 41 security recommendations evaluated in a universe of 561 devices. 78

5.1 High level vision of the benchmarking process 82
5.2 Main elements of the experimental setup considering three approaches: A) Manual; B) Automatic; and C) Hybrid. 88
5.3 Analysis of the five security configurations that are part of the security qualification step. 94
5.4 Devices ranking showing the precedence of them. 96

List of Tables

2.1 Comparison between this work and the 7 most relevant related work. . . . 34

3.1 Consolidation of security concerns from CIS security benchmarks. 39
3.2 Presentation of new security concerns. 40

4.1 Semantic of each value in the risk scale used. 68
4.2 Example of the experts perception for some security recommendations. . . 69
4.3 Security concerns and their respective weights. 71
4.4 Severity classification of the security recommendations. 72
4.5 Security configurations for five possible cases. 76
4.6 Security assessment for five possible cases. 78
4.7 Presentation of the security configuration set of three installations. 79

5.1 Set of Android security configurations and their respective risks. 91
5.2 Presentation of the security configuration set of the best and worst configuration of the dataset, as well the most common misconfigurations. 95
5.3 Risk exposure and risk percentage of 10 devices chosen from the dataset. . 95

A.1 Template values of security concerns from CIS security benchmarks. 114

Contents

1 Introduction **16**
1.1 Towards benchmarking the security of mobile devices configurations 18
1.2 Main contributions of the work 19
1.3 Structure of the thesis 20

2 Background and Related Works **22**
2.1 Background on mobile devices 22
2.1.1 Mobile devices vs PCs 23
2.1.2 Security in mobile devices 23
2.1.3 The Android OS from a security perspective 26
2.2 Risk analysis 27
2.3 Benchmarking computer systems 29
2.3.1 Performance benchmarking 29
2.3.2 Dependability benchmarking 30
2.3.3 Security benchmarking 31
2.3.4 Related Work 31
2.4 Conclusion 34

3 On the relevance of security configurations **36**
3.1 Identification of user-defined security configurations 36
3.1.1 Concepts, definitions and procedure 37
3.1.2 CIS security settings 38
3.1.3 Additional security settings 40
3.2 The uSEA tool 42
3.2.1 Tool architecture 42
3.2.2 Collecting the security configurations 44
3.3 Case Study: Analysis of user-defined security configurations of real devices 46
3.3.1 Data gathering and sample space 46
3.3.2 Overall results 47
3.3.3 Application category 48
3.3.4 Browser category 49
3.3.5 Content category 50
3.3.6 Network category 51
3.3.7 Password category 51
3.3.8 Permission category 52
3.3.9 System category 53
3.4 Mapping configurations with known vulnerabilities and exploits 54
3.4.1 Browser Exploit 55

3.4.2 Data Interception 56
3.4.3 Malware 57
3.4.4 Unauthorized Location Tracking 57
3.4.5 Network exploits 58
3.4.6 Abuse of costly services and functions 59
3.4.7 Phishing and Spoofing 59
3.4.8 Denial of Service 60
3.4.9 Privilege Escalation 60
3.5 Countermeasures recommended 61
3.6 Conclusion 63

4 Risk assessment of security configurations 65
4.1 Experts analysis 67
4.1.1 Experts categorization 67
4.1.2 Gathering likelihood and impact information from experts 67
4.2 Analysis of the configurations using GRIP 69
4.3 The groups of security recommendations 72
4.3.1 Grouping the security configurations 72
4.3.2 Describing the groups 73
4.3.3 Using the groups to assess risk and identify problems 74
4.4 Case Study: Risk Analysis of real Android Devices 75
4.4.1 Analysis of typical installations 76
4.4.2 Analysis of real configurations of Android devices 78
4.5 Conclusion 80

5 Benchmarking security configurations 81
5.1 A risk-based approach for benchmarking security configurations 82
5.1.1 Security qualification 82
5.1.2 Risk quantification 83
5.2 Specifying benchmarks for security configurations 84
5.2.1 Benchmark elements 84
5.2.2 Benchmark components 85
5.2.3 Benchmark properties 90
5.3 Case study: A benchmark for Android devices 91
5.3.1 Defining the set of security configurations 91
5.3.2 Defining the risk of each security configuration 91
5.3.3 Defining the subsets for security qualification and risk quantification 92
5.3.4 Preparing the experimental setup 92
5.3.5 Gathering the device configurations 93
5.3.6 Calculating the metrics 93
5.4 Experimental evaluation and results 94
5.5 Validating the proposed security benchmark 96
5.5.1 Representativeness 96
5.5.2 Portability 97
5.5.3 Repeatability 97
5.5.4 Scalability 98
5.5.5 Non-intrusiveness 98
5.5.6 Simplicity of use 98
5.6 Conclusion 99

6 Conclusions and future work **101**
6.1 Main achievements 102
6.2 Threats to validity 103
6.3 Future work 104

Bibliography **105**

A Security configuration values **114**

Chapter 1

Introduction

Mobile devices are an increasingly attractive target for cyber attacks [86][1]. In fact, the popularity of mobile devices (tablets and smart phones) and their current use for, among other activities, online banking, social networking, entertainment, taking photos, exchanging text and multimedia messages, and buying applications, movies or books, makes them a key storage device for sensitive information, raising new security and privacy issues [63]. In practice, mobile devices provide many of the traditional features of personal computers, but they also include a variety of connectivity technologies, such as GSM (Global System for Mobile Communications), GPRS (General Packet Radio Service), NFC (Near Field Communication), Bluetooth and WiFi, among other sensors and hardware, like gyroscopes, GPSs (Global Positioning System), and built-in cameras and microphones, all in a smaller and lighter device. As such characteristics facilitate and increase the chances of having the device attacked, stolen[2] or lost, users of mobile devices should be aware of data disclosure or leakage problems, which brings the need for novel security mechanisms and tools, including techniques for assessing how secure a given system/configuration is.

Security is an integrative concept that includes the following properties [97]: *confidentiality* (protection against unauthorized disclosure of a service or piece of information), *authenticity* (assurance that a service or piece of information is authentic), *integrity* (protection of a service or piece of information against unauthorized modification), and *availability* (protection against possible denials of service caused maliciously). In practice, the goal of information security is to protect systems, resources, data and other assets from intrusion (unauthorized accesses) reducing the risk of security incidents. The risk of intrusion is related to the system vulnerabilities and the potential security attacks. The system vulnerabilities are an internal factor related to the set of security mechanisms available (or not available) in the system, the correct configuration of the system, and the hidden flaws on the system implementation (in other words, a vulnerability is any weakness that can be exploit and compromise the security of any information system). Vulnerability prevention consists of guarantying that the software has the minimum vulnerabilities possible (e.g. by using security testing). On the other hand, vulnerability removal is the process of mitigating the vulnerabilities found in the system (e.g. by applying new secu-

[1]The total mobile phone market is forecast to reach 1.9 billion units in 2016 [32].

[2]A survey conducted by the Consumer Reports National Research Center stated that 2.1 million Americans had phones stolen in 2014 [73].

rity patches released by software vendors) [99]. It is well-known that vulnerabilities range from configuration problems to programming bugs and project defects [55]. This way, correctly setting all available configurations do not prevent all possible attacks against mobile devices, but for sure increases the bar of effort necessary for an attacker to be successful in his intent.

A security attack is any action that compromises the security of an individual or organization. It mainly depends on the intentionality and capability of humans to maliciously break into the system taking advantage of vulnerabilities [99]. The success of a security attack is related with the existence of vulnerabilities in the system and attacks can be harmless in a system without vulnerabilities. On the other hand, vulnerabilities are harmless if the system is not subjected to security attacks. The prevention against security attacks includes all the measures needed to minimize or eliminate the potential attacks against the system (e.g. by reducing the potential attack surface). A system's attack surface is the set of ways in which an adversary can enter the system and potentially cause damage [54]. Attack removal is related to the adoption of measures to stop or mitigate types of attacks that have occurred before (e.g. by using intrusion detection system and intrusion prevention system).

As the purpose and usage of mobiles devices increases, the assessment of security characteristics becomes a major issue to understand the effectiveness of existing security mechanisms and configurations. A key aspect is that security is not only related to the hardware and software used, but it also depends on the way the system is used and on the characteristics of the surrounding environment, where the user-defined settings play a particularly important role [16]. In this work, we define security assessment as a process designed to evaluate identifiable security aspects (e.g. user-defined security configurations) of systems in an application domain (e.g. mobile devices), dividing the evaluated systems into acceptable or unacceptable, given the specific security requirements and security knowledge in that domain. In practice, as proposed in [16], security assessment must consider two perspectives: 1) the active search for vulnerabilities and security problems, and 2) the characterization of the proneness for other hidden or unidentified problems to exist.

The security of mobile devices is nowadays a major concern for organizations. Due to their small size and memory capability, and to the ease with which information can be disclosed from a facility, mobile devices pose a considerable risk when used and transported outside the physical boundaries of an organization [37]. Mobile devices can include Personal Digital Assistants (PDAs), mobile phones, laptops, tablets, smart phones, removable media and others [37]. However, in this thesis, we delimit the scope of mobile devices to smart phone and tablets. Some existing security solutions for mobile devices, such as Intrusion Detection Systems (IDS), Antivirus, Firewalls, Anti-theft [50], and static and dynamic analysis of the behavior of applications [23,24], can be used to detect and prevent some vulnerabilities and attacks intended against mobile devices. However, as stated by Mylonas et al. [62], none of the existing platforms is able to fully prevent attackers from using mobile devices as privacy attack vectors, harvesting data from the device without the user's knowledge and consent. Some countermeasures that can be applied include (a) user awareness, i.e. informing the user about security and privacy risks and limitations of smart phone platforms, and (b) providing secure application distribution in smart phone platforms. In this work we are particularly concerned with the **awareness that users and manufacturers should have towards the perils behind user-defined configurations**.

1.1 Towards benchmarking the security of mobile devices configurations

Many security issues depend on the way the system is configured and used, and on the characteristics of the surrounding environment, where the user-defined settings play an important role [16]. In this work, we define *security configuration assessment* as a process designed to evaluate identifiable security aspects (e.g. user-defined settings) of the systems in an application domain (e.g. mobile devices), given the latest specific security requirements and knowledge in the domain. A *user-defined security configuration* is understood as any setting, configuration or value that users can define or set and that have some impact in the overall security of the device (e.g. setting a password to unlock the device or defining a timeout to automatically lock the device after a period of inactivity). Finally, a *security configuration recommendation* is a suggestion for a specific user-defined configuration or a user action that impacts the security of the device (e.g. install anti-malware).

A device with configurations improperly set can lead to an unsafe environment, as some configurations have directly impact in the information security, widening or reducing the attack surface. Take as an example the Bluetooth configuration, if the user is not using this feature and let it enabled, it unnecessarily opens possibilities for an attacker to explore vulnerabilities regarding the Bluetooth. The device unlock password is another well known case, as it is the user responsibility to set a password to avoid unwanted people from easily getting access to the device.

Assessing the security level of user-defined configurations is an open problem in different areas. In fact, the security of mobile devices, as other computer systems, depends on different aspects, such as the hardware, operating system, software and also the way that users configure and use their devices.

The common sense suggests that users will, probably, never know information and systems security deeply enough to safely manage their configurations. This way, this work proposes techniques and tools for the creation of a security benchmark that aims at assessing and comparing the security level of mobile devices regarding their configurations. A special focus is put on Android devices, but mainly in the case studies, as most of the proposed approaches can be easily adapted to other mobile platforms. The focus on Android is supported by its wide use (a market share of 82.2 percent for smart phones was reported in the second quarter of 2015 [30] and a market share of 61.9 percent for tablets was reported in 2013 [31]), and its representativeness with respect to the state-of-the-art in mobile computing.

Towards the goal of benchmarking user-defined security configurations of mobile devices, a set of research problems have to be overcome. The first is the need for identifying the user-defined configurations that have a direct influence in the device security. The literature is vast addressing this subject, but there is no consensus or standardization of the security configurations of mobile devices.

Another important issue is collecting field data to support the analysis of the most common security issues regarding user configurations, and their real impact in terms of security flaws. In fact, there is a lack of research in this subject as the focus is mainly on the users' perception of security and not on effectiveness of their real configurations. Although such analysis may be supported by a risk-based approach (to understand the impact and likelihood that each user misconfiguration may have in the device or the user if attacked), the literature is scarce in this subject, addressing neither qualitative nor quantitative analysis of user-defined security configurations of mobile devices.

The definition of a benchmark to compare and rank a set of mobile devices considering the security aspect of user-defined security configurations is the ultimate research challenge. Security configuration benchmarks have already been used in the databases domain, but supported by a manual analysis applied over a small set of databases.

1.2 Main contributions of the work

The main contribution of this work is the **definition of an approach to benchmark the security level of mobile devices from the perspective of the user-defined security configurations**. Such approach can be use, in practice, in several contexts for improving the security. For example, by companies that allow employees to access their information system through smartphones and by manufacturers that want to improve the security of their mobile devices before release. More specifically, the main contributions of this work are:

1. The identification of a list of best configuration practices that can be used to enhance the security of Android devices (see Chapter 3), this list was gathered from the literature and can be used to evaluate device's security level. A great part of these configurations was extracted from the CIS [27] security benchmarks;

2. A tool, available at Google Play, that automates the collection and analysis of the values of the security settings of Android devices (see Chapter 3). This tool presents to the user the misconfigurations that exist in the device and has been used to gather the dataset for the case studies in this work. The tool automatically extracts 41 configurations from the mobile device under testing, 14 of which defined and proposed in this work and the remaining adapted from the well-known CIS benchmarks [27], providing information that can be used for security assessment and improvement;

3. An analysis of the configurations from 561 Android devices on the field. Preliminary results show that the security of user-defined configurations in Android mobile devices is a major concern. A key observation is that security configurations are closely linked to the initial factory settings, which suggests that users need to better understand and change the settings to become more secure.

4. A risk-based approach for assessing the security risk posed by user-defined configurations in Mobile devices. The impact and likelihood values are defined based on a Multiple-Criteria Decision Analysis (MCDA) performed on the inputs provided by a set of security experts. A case study considering the user-defined configurations of 561 Android devices is presented, showing that the majority of the users neglect important and basic security configurations and that the proposed approach can be used in practice to characterize the security risk level of such devices;

5. A security benchmarking methodology that aims at assessing and comparing different mobile devices or different configurations of a given device concerning the user-defined security configurations. The benchmark is built on top of the risk analysis approach mentioned above, and specifies all the components and steps needed to rank the devices under benchmarking. This may stimulate users and vendors to improve the security level of such devices.

A key aspect is that, although the proposed tool focus specifically on Android devices, most of the settings assessed also apply to other mobile platforms. This will facilitate the future development of similar tools targeting other systems, for which a large part of our architecture can be reused. On the other hand, the risk analysis approach, the security benchmarking process, and the security certification process, are platform independent (although they are demonstrated to the specific case of Android).

As this project get, manipulate, analyze and disclose sensitive data from humans beings, it is necessary to note that this work has been approved by the University of Campinas ethics committee through the report 1.145.809.

1.3 Structure of the thesis

This first chapter introduced the problem addressed and the main contributions of this thesis.

Chapter 2 presents background concepts and related work, making a comparative analysis between this work and the known literature. In practice, we discuss the main aspects of security of mobile devices and benchmarking.

Chapter 3 presents a set of user-defined security configurations, as well the process followed to gather them. The chapter also presents an automated tool to extract the security configurations from Android devices on the field. Finally, it presents an analysis of the most common security configurations and proposes countermeasures that can be applied to reduce the risks raised by a incorrectly configured device.

Chapter 4 presents an approach to determine the risk level of security configurations using the perception of experts selected from both academia and industry. The risks are categorized into five groups of severity and a case study showing the risk analysis for 561 devices is presented.

Chapter 5 presents a specification of a configuration security benchmark for mobile devices. The components, properties and configuration settings are presented in detail. A case study is also provided, showing how the results of such benchmark can be analysed and used in practice.

Chapter 6 concludes the paper and puts forth ideas for future work. This chapter also summarizes the main results and shows the list of publications achieved during this work.

Chapter 2

Background and Related Works

The widespread of mobile devices brings challenges to the security of computer systems, raising new perspectives and opportunities of research, including the assessment of user-defined security configurations. Nowadays, the networks of organizations should consider new issues, including [84]: **an increasing use of new devices**, as it is common for employees to access the company systems from their devices, a trend called BYOD (Bring Your Own Device); **the emergence of cloud-based applications**, in which the data are available everywhere, as far as an Internet connection is available; and **a dynamic perimeter**, as the increasing use of new devices and cloud-based applications together changed the notion of statical perimeter that existed in the past. In practice, due to its widespread and *sui generis* characteristics, the security of mobile devices is an key challenge that should be addressed by manufacturers and researches.

This chapter presents background and related works on mobile devices, focusing particularly on the security, risk analysis and benchmarking aspects. We will show that there is a gap in the literature with respect to the security assessment of mobile devices, especially in what regards the user-defined security configurations. This opens many possibilities of research and is an key area to be addressed.

The outline of this chapter is as follows. Section 2.1 presents background on mobile devices. Section 2.2 presents existing risk analysis approaches. Section 2.3 discusses background and related on benchmarking of computer systems. Finally, section 2.4 concludes the chapter.

2.1 Background on mobile devices

The availability of inexpensive mobile devices such as smartphones and tablets has accelerated their popularization. In fact, smartphones are ever-present in our daily lives, allowing us to take photos, access bank accounts, share information, and perform many other diverse tasks. Moreover, as more organizations implement BYOD (bring-your-own-device) policies, employees increasingly use personal mobile devices to access enterprise systems and information, which raises new security issues, especially regarding user-defined security configurations. In the following sections we discuss the specific security characteristics of mobile devices that differentiate them from personal computers.

2.1.1 Mobile devices vs PCs

Smartphones and tablets have much in common with PCs, but five key aspects distinguish mobile devices security from conventional computer security [61], which have to be taken into account when developing new techniques and tools for assessing and improving the security of mobile devices:

1. *Mobility*: as mobile devices are not kept in a location that can be physically secured, they can be easily stolen or tampered;

2. *Strong personalization*: mobile devices are normally not shared between multiple users, having a strong emphasis on user-defined configurations;

3. *Strong connectivity*: most devices support multiple ways to connect to a network or the Internet;

4. *Technology convergence*: a single device combines different technologies (music player, mobile phone, digital camera, etc.), which may enable an attacker to exploit different routes to perform attacks; and

5. *Reduced capabilities*: although smartphones are like pocket PCs, they lack some characteristic features; for example, they have incomplete keyboards and reduced battery autonomy.

As people increasingly use smartphones for privacy-sensitive transactions, such as online banking and e-commerce, the likelihood of threats designed to generate profit for the attackers also greatly increases. In fact, the reported number of new mobile vulnerabilities rose from 127 in 2013 to 168 in 2014 - a 32 percent increase [86]. Also reported in the first quarter of 2015 was the fact Symantec blocked approximately 550 malware attacks against Android each day [86]. This confirms that attackers are moving their efforts to mobile platforms. The scientific community and manufacturers should follow this trend and focus more on researching new mobile device security solutions [50].

2.1.2 Security in mobile devices

The security of mobile devices is a major concern for end-users and also for companies that allow employees to access their information systems from mobile devices. The identification of the security issues of these devices is a relevant task to mitigate the risks raised by the use of mobile devices to perform sensitive tasks.

Goal of attacks

Attacks to computational systems are a common problem for companies and also for end users. Attackers have different focus when focusing on mobile devices, but the most relevant are [50]:

- Privacy: this type of attacks concern situations in which confidentiality is compromised. Attackers amy develop applications that use APIs to retrieve the list of contacts, media, location, and others [25]. In fact, mobile devices are easy to be stolen, which can lead to sensitive data disclosure if the data are not protected (e.g. by encrypting the device, or enabling password to unlock the device);

- Sniffing: consists in the capture of valuable information based on the use of sensors, such as the camera, microphone and GPS. These sensors can be used to record sensitive users actions;
- Denial-of-Service (DoS): this type of attacks affects the availability of the device. DoS against mobile devices can be performed with a small effort, for instance by draining its battery or performing CPU intensive tasks;
- Over-billing: this can be performed in several ways; for example, the attacker may make premium-rate calls or send SMS generating extra charges in the phone bill of the victim, which can easily go unnoticed until the user gets the next bill [25]. Another way to perform this attack is by exploiting the pay-per-use contracts in wireless services performing downloads in the victim device that generate extras fees;
- Ransom: this type of attack is performed by encrypting the victim device and demanding a value to decrypt it [25].

There are other goals that are related to the psychological characteristics of the atackers, such as *personal satisfaction*, in which the attacker breaks systems to show and/or challenge their skills [65], and *personal revenge*, where not satisfied employees use their privileged knowledge of the company systems to cause security incidents [65];

Attacks channels

Attacks against mobile devices generally occur through four different channels [28]:

- Software downloads (e.g. downloading a malware from the black market);
- Visiting a malicious website (e.g. phishing attacks);
- Direct attack through the communication network (e.g. man-in-the-middle attacks in a WiFi network);
- Physical attacks (e.g. attacks performed in lost/stolen devices).

The *software downloads* channel can be mitigated or even avoided by the installation of an anti-malware and by not installing malicious or suspicious applications. Disabling the installation of applications from unknown sources is another practice that helps closing this channel, as the user has more security guarantees when installing applications from the official app stores (e.g. Google Play, AppStore).

The *visiting a malicious website* can be mitigated or even avoided by correctly configurating the browser application. For example, some attacks, such as Cross Site Scripting

(XSS) and Cross Site Request Forgery (CSRF), can be mitigated by disabling JavaScript, as both attacks need to execute a fragment of JavaScript prepared by the attacker.

The *communication network attacks* can be mitigate or even avoided by correctly configuring the network interface to disable unused communication channels (e.g Bluetooth and WiFi).

Physical attacks can be mitigated or avoided by the correct configuration of the device password, requesting strong passwords to unlock the device. Others configurations that are also related to avoiding physical attacks are updating the firmware to the latest version, disabling developer options and encrypting the device storage.

Correctly setting all configurations does not avoid all possible attacks against mobile devices, but it increases the bar of effort necessary for an attacker to be successful in his attempt. Also, user-defined configurations are not enough to avoid attacks that exploit vulnerabilities in the operating system (e.g. allow unlock the device due to an error in the home application) or in the hardware (e.g. allow to enter in flash mode to change the operating system).

Security mechanisms

Mobile devices have security mechanisms to prevent different types of threats, including:

- Intrusion Detection Systems (IDS): are capable of detecting several attacks and intrusions, assisting in device protection [65]. There are two main types of intrusion detection techniques: *anomaly-based* that compare the real behaviour with the normal one; and *signature-based* that are based in signatures of well-known attacks [50].
- Trusted Mobile: the Trusted Computing Group (TCG) created a set of specifications to measure, store, and report hardware and software integrity through a hardware root-of-trust. The integrity measurements are performed during the load-time and at runtime (dynamic measurements).
- User-defined Security Configurations: operating systems have a set of configurations that can be set by the users or systems administrators to enhance the device security. For example, enabling a password and encrypting the device storage can be seen as actions that should be taken to decrease the attack surface of the device.

Vulnerabilities

There are many types of vulnerabilities in mobile devices platforms. Symantec reported 528 new mobile vulnerabilities in 2015, a growth of 214% if compared with 2014 (168 new vulnerabilities) [86] (in fact, the number of mobile vulnerabilities has increased every year over the past three years). In the same report, Symantec stated that the greatest number of the mobile vulnerabilities discovered in recent years are on the iOS platform, where the greater attacker goal is to jail-break (gain root privilege) devices or gain unauthorized access to install malware.

Vulnerabilities can be exploited by malware in order to gain root privilege access on compromised mobile devices. For instance, some vulnerabilities that allow attackers to compromise Android devices by simply sending them a malicious multimedia message (MMS) were reported in 2015 [86]. Other similar vulnerabilities (CVE-2015-6602 and CVE-2015-3876) allow an attacker to gain control of a device when the targeted victim opens an .mp3 or .mp4 file. These vulnerabilities were already patched by Google, but this shows that easy-to-exploit vulnerabilities are indeed present in mobile devices platforms.

2.1.3 The Android OS from a security perspective

Android is the most popular operating system for mobile devices. In fact, it dominated the market with an 82.8% share in the second quarter of 2015 [41]. A great advantage, if compared with its main competitors (iOS and Windows Phone), is the fact that Android is an open source operating system based on the Linux kernel. This turned the Android OS the main target of researchers focusing in different aspects, such as security, malware detection and others.

There are different types of **security mechanisms** present in the Android platform, but some are more representative than others, being thus explored in more researches, namely:

- Permissions: applications must declare the permissions they need for additional capabilities, including access to device features such as the camera or the Internet. The permissions must be accepted by the user before installing an application (before the Android Marshmallow version) or when the application accesses the resource protected by the permission (after the Android Marshmallow version). Some works use the Android permission system as a parameter for malware detection [8, 106];

- Application sandbox: isolates the app data and code execution from other apps. If needed fact, applications must explicitly share resources and data, because Android sandboxes the applications from each other;

- Linux Kernel: as the Android framework is built on top of the Linux Kernel, it share its security mechanisms, such as file permissions and others;

- Application Review [25]: Apple and Google review all applications that are available at their stores (respectively, App Store and Google Play). Some are even reviewed manually by experts. This process reduces, but does not avoid, the presence of malware in the official application markets.

There are some research works [23, 79] that propose the extension of the Android platform to improve the identification of malware/intrusion by behavior-based intrusion detection systems. Andromaly [79] is a framework that implements a Host-based Malware Detection System that continuously monitors various features and events obtained from the mobile device and applies Machine Learning anomaly detectors to classify the collected data as normal (benign) or abnormal (malicious). Another example is TaintDroid [23], which is an extension of the Android platform that tracks the flow of privacy sensitive data through third-party applications.

2.2 Risk analysis

The works of Rosenberg et al. [74], Amland [5], Halkidis et al. [36] and Hogben and Dekker [39] are key references on the concepts of software risk analysis. The risk of a software program can be characterized by the combination of two factors: the severity of a potential failure event and the probability of its occurrence, as shown in equation (1) extracted from the work of Rosenberg et al. [74]:

$$risk = \sum (p(Ei) \times c(Ei)) \tag{2.1}$$

where:

i is the number of unique failure events;

Ei are the possible failure events;

p is probability;

c is cost.

Although this definition is almost a standard among authors, there is no consensus on how to obtain the probability and the cost of a failure, and the subject is approached differently by various authors. Risk-based testing focuses on the parts of a software that are most likely to experience a problem that would have the highest impact [82]. It seems to be a discouraging task, but once it is decomposed into small tasks, a systematic approach can be employed to make it manageable [74].

For Amland [5], the consequences of a failure event in the operation environment should be analyzed from the point of view of the supplier and of the customer. These two views are considerably different. For example, for the customer the software may lead to the violation of service level agreements, disobedience of government rules and loss of market share in its segment. For the supplier, the consequences of negative publicity and high maintenance costs may be relevant. This way, Amland expanded the original formula including the two views, as shown in equation (2).

$$risk = \sum (p(Ei) \times (\frac{(sc(Ei) + cc(Ei))}{2}) \tag{2.2}$$

where:

i is the number of unique failure events;

Ei are the possible failure events;

p is probability;

sc is the supplier's cost if a failure occurs in Ei;

cc is the customer's cost if a failure occurs in Ei.

In this work we consider the risks regarding the user-defined configurations of mobile devices concerning both the user side and an attacker side, so we need to analyze the risks regarding both gain (of the attacker) and loss (of the target). The most important for us to consider, are those risks that may have a higher impact in the target (user) and a higher gain to the attacker. Note that, according to Pfleger [70], it is not possible to estimate the probability as an exact measure, but experimental data may be used to assess the risk.

Zhang et al. [49] proposed a qualitative and quantitative risk assessment method or software security combining attack tree model analysis and Bayesian Network analysis. Modeling with attack tree, which solves the problem of how to quantify the risk possibilities, takes the vulnerabilities of the system and the capability of the attackers into account to estimate risk probabilities (although this approach can not accurately determine the probability of events). The risk assessment method proposed in this work is accomplished in two steps: 1) identification of the possible risks with the attack tree model; 2) quantifying the probabilities of attack occurrences, the impact and the levels of security risks through Bayesian Network analysis. Although complete, expressing attacks as trees might not be the most flexible approach, meaning that a complex attack tree may be very difficult to analyze [16].

Halkidis et al. [35,36] proposed an architectural risk analysis for software systems based on security patterns in order to determine to what extent specific security patterns can be used to protect against known attacks. This information is fed into a mathematical model based on the fuzzy-set theory and fuzzy fault trees in order to compute the risk for each category of attacks, namely Spoofing, Tampering-with-Data, Repudiation, Information Disclosure, Denial-of-Service, and Elevation-of-Privilege (STRIDE) [40]. Similarly to the aforementioned studies [5, 74], the computed risk considers the likelihood of occurrence of a risky event, the exposure of the system to that event and its consequences. However, when considering mobile devices, STRIDE is not the best threat model as it is too generic, and the actual semantics involved in the application of each of these threats are too open for disagreements. For example, the *Information Disclosure* threat can happen through different means such as *physical access*, *interception of traffic data*, and *insider threats* (e.g. malware) [16].

The works cited above are important to provide a theoretical basis and served as a starting point for our research, although none of then focus on mobile devices. This means that new research in this direction is needed. Next we discuss some key works on risk analysis of mobile devices, with the exception of the works by Milligan and Hutcheson [58] and of Shabtai et al. [77, 78], which are discussed in Section 2.3.

Hogben and Dekker [39] focused their work on information security risks, opportunities and recommendations for smartphone users, with inputs from a group of experts from across government and business organizations. For them, risks vary depending on how the smartphone is used and the usage scenario can be divided into three categories, as follows:

- Consumer: the device is used for personal tasks, e.g. phone calls, social networking, gaming, online banking, among others.

- Employee: the device is used by an employee in a business or government organization to access information or systems.
- High official: the device is used by a high position employee within the organization, so it deals with sensitive information and/or tasks.

The risks, for each category of user, are computed considering both the likelihood and the impact. The *likelihood* of a threat is determined by the number of underlying vulnerabilities, the relative ease with which they can be exploited and the attractiveness for an attacker. The *impact* of a threat can be determined by the value of the assets affected by the threat. In practice, the risks were determined by consulting a group of experts that were asked to indicate the likelihood (from very low to very high) and impact (from very low to very high) of each risk in each of the three usage scenarios. This work is very interesting, complete and similar to our intended risk analysis, although our focus is on the analyzes of the risks risen by the user's misconfigurations.

Theoharidou et al. [87] presented a risk assessment method specific for smartphones. It identifies smartphone assets and provides a detailed list of specific applicable threats, which are also present in other several works [24,25,39,44,50,58,77] (although there is no standard in the known literature). The method is based on user input, with respect to the impact assessment, coupled with statistics for threat likelihood calculation. However, similarly to the aforementioned works, they do not take into account the risk of user misconfigurations.

The works introduced above are important to understand the strengths and weaknesses of the known literature regarding risk assessment and risk analysis of mobile devices. Some of the related work [10,13,64] analyze the importance and/or the perception of the user on the security of their mobile devices, but none of then actually does the assessment of real mobile devices or take the user-defined configurations into account for the risk analysis, which is a clear contribution of our work.

2.3 Benchmarking computer systems

A *benchmark* can be defined as a standard procedure that allows evaluating and comparing different systems or components according to specific characteristics such as performance, dependability, and security [102]. In the next subsections we discuss performance, dependability and security benchmarks, with more emphasis in the last ones due to their relevance to our work.

2.3.1 Performance benchmarking

Performance benchmarks are the most common type of benchmark and can be defined as a standard procedure and tools to compare the performance of systems in a given domain. There are several research works in the literature and tools in the market to evaluate the performance of databases [33,89], hard disks and solid-state drives [71], web servers [83], hardware [83], and many others.

There are two main organizations in the performance benchmarking business [97]: TPC (Transaction Processing Performance Consil) [89], which is a non-profit organization founded to define transaction processing and database benchmarks; and SPEC (Standard Performance Evaluation Corporation) [83], which is a non-profit organization created to establish, maintain and endorse a standardized set of relevant benchmarks in several domains, but with a strong emphasis on high-performance computers.

A performance benchmark normally includes three main components [97]:

- Workload: defines a set of tasks that are representative of the ones normally executed in real environments, thus being dependent on the benchmark domain. This way, some workloads include io-bound (great usage of input/output operations) tasks, while others focus on cpu-bound (great usage of processing operations) tasks. Mix loads are also used in some cases.
- Measures: defines a set of representative measures that are used to portray and compare the performance of the System Under Benchmarking (SUB). Some examples of performance measures are: transactions per second (in the database domain), megabits per second (in the networks domain), response time, etc.
- Rules: set of criteria that must be followed during the benchmark implementation and execution in order to make the results valid and comparable [97].

2.3.2 Dependability benchmarking

Dependability is defined as "*a property of a computer system such that reliance can justifiably be placed on the service it delivers*" [51]. In other words, dependability is an integrative concept that includes the following properties: availability, safety, reliability, confidentiality, integrity, and maintainability [52].

Dependability benchmarks have been proposed for different types of systems ranging from databases [97] to virtualization infrastructures [14]. The goal of benchmarking the dependability of computer systems is to provide generic and reproducible ways for characterizing their behaviour in the presence of faults [47]. This way, such benchmarks include an additional component: the *faultload*; and additional dependability benchmark measures.

A *faultload* in a key component of dependability benchmarks and should include a representative, realistic and repeatable set of faults that allows the characterizations of the behavior of the system in the presence of faults. There are different kinds of faults that can be considered: *hardware* (low-level faults affecting the values in memory, registers, etc.), *software* (defects in the code that escaped the verification and validation processed), and *operator* (e.g. administrator mistakes in the administration tasks).

In addition to performance measures, a dependability benchmark includes dependability-related measures [100]. Some examples found in existing benchmarks are the **number of data errors detected** (measures the impact of faults on the data integrity), and the **availability of the system under testing** during the execution of the workload in the presence of the faultload (measures the amount of time the system is available during the experiments).

2.3.3 Security benchmarking

A *security benchmark* provides a metric (or small set of metrics) to characterize the level to which security goals are met [102] in the system under testing, allowing developers, administrators and device owners to compare alternatives and make informed decisions [67]. A key aspect is that security is, usually, more dependent on what is unknown about the system (e.g. unknown bugs, hidden vulnerabilities) than on what is known (e.g., known features, existing security mechanisms) [102], which makes *security assessment* a challenging task.

A security configuration benchmark is a process designed to evaluate identifiable security aspects (e.g. user-defined settings) of systems in an application domain (e.g. mobile devices), dividing the evaluated systems into acceptable and unacceptable, given the specific security requirements and security knowledge in that domain. In practice, the benchmark provides a metric (or a small set of metrics) able to compare and rank the security level of a system accordingly to its configurations.

The general idea is to compare the devices security level according to the configurations that users make in their devices. This can be used, for instance, by companies who want to allow their employers to access the enterprise systems through their personal mobile devices. With such benchmark, the company is able to qualify whether the device is secure or not to access the systems without a major risk of disclosing sensitive or confidential information.

Unlike performance and dependability benchmarks, a security configuration benchmark does not need a *workload* nor a *faultload*. On the other hand, there is the need for a *set of user-defined security configurations*, which includes the set of configurations that can be modified by the user (device owner) and have some influence in the device security (e.g. enable a password to unlock the device and/or set a inactivity timeout before automatically lock the device). A set of security configurations for the mobile devices domain is discussed in detail in Chapter 3 and our security configuration benchmarking approach is discussed in Chapter 5.

2.3.4 Related Work

Security assessment methodologies and techniques have been proposed in several forms. The Common Criteria standard is one of the security assessment methodologies available and supported by the ISO/IEC standards organization [42]. The Common Criteria (CC) is an international standard (ISO/IEC 15408) for computer security. Its purpose is to allow users to specify their security requirements, developers to specify the security attributes of their products, and evaluators to determine if products actually meet their claims. Additionally, it presents requirements for the Information Technology security of a product or system under the distinct categories of functional requirements and assurance requirements.

The Common Criteria functional requirements are able to define the desired security behavior, while assurance requirements are the basis for gaining confidence about the fact that the claimed security measures are effective and correctly implemented [56]. However, the Common Criteria received a lot of critics from researchers due to several reasons, including its high complexity, time consuming and emphasis in the analysis of specification documents instead of real implementations. Another drawback of the Common Criteria is that the security requirements must be defined before the project starts [1, 43], which is quite difficult in such an uncertain Information Technology environment.

Another important methodology for security evaluation is the OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation) approach [4] from the Software Engineering Institute (SEI) of the Carnegie Mellon University. The OCTAVE is a risk-based strategic assessment and planning technique for security. OCTAVE is self-directed, meaning that people from an organization assume responsibility for setting the organization's security strategy (i.e. they support the process of self-evaluation within an organization) [3]. However, this method is quite difficult to use to compare different environments, to understand the impact of security decisions, or to evaluate the security of software alternatives, as it is designed for the organization as a whole [16].

The security benchmarks from the Center for Internet Security (CIS) intend to provide best practices to help organizations assessing and improving their cyber security [27]. CIS is a nonprofit organization focused on enhancing cyber security by providing resources that help organizations achieving their security goals through prescriptive guidance solutions. They created and maintain a series of security configuration documents (called benchmarks) for different commercial and open source systems such as mobile devices (iOS6, iOS7, Android 2.3, and Android 4.0), network devices, operating systems, and server software.

The recommendations prescribed by CIS are available in the form of documents that focus on the practical aspects of the configuration of the target systems and assert the correct value that each configuration option should have in order to enhance the overall security during the system usage. In practice, each document contains, among other information, the relevant security settings, including a short description and discussion on the relevance of each setting, the audit steps needed to collect the configured values, and the proposed value (and rationale) for each setting. The set of security configuration benchmarks created by the CIS is an extremely relevant initiative and the general idea is that the proposed guidelines are meant to be used by system and application administrators, security specialists, auditors, help desk, and end users to enhance the security of their systems. The key advantage of the CIS approach is that the configuration settings are based on field experience. However, although the documents are referred as benchmarks, they do not provide any solution for automated security characterization.

Neto and Vieira [66] proposed an approach to assess the effectiveness of database management systems (DBMS) configurations concerning security. The approach consists in the use of a set of security best practices obtained through a detailed analysis of the CIS series of security configuration documents for DBMS. The security best practices are used to define configuration tests that can be executed in a semi-automated manner (some tests can be performed by simple tools, but most require the human analysis of the DBMS configuration), thus helping on characterizing (from a qualitative point-of-view) the security of the target system. The main limitation of the work is the complexity of the intended target systems and the need for manual analysis, which may affect the results obtained.

Paranoid Android (PA) [72] proposed a scalable smartphone security architecture that is able to apply security checks on remote security servers that host replicas of phones in a virtual environment. In practice, a recorded execution trace is transmitted to the cloud over an encrypted channel, where a replica of the phone is running on an emulator. Using runtime analysis in the emulator, PA detects certain types of attacks, such as buffer overflow and code injection. This work presents a security model that performs attack detection on a remote server, being in this regard similar to the architecture of our tool. However, it focus on identifying attacks via the analysis of the phone system calls usage, and not on the identification of vulnerabilities on the configuration of devices.

Chin et al. [13] conducted a user study involving 60 smartphone users to gain insight into the user perception on smartphone security and application installation habits. They found that, in general, users are (a) more concerned about privacy on their smartphones than on their laptops, and (b) more apprehensive about performing privacy-sensitive and financial tasks on their smartphones than on their laptops. The work states that security and/or user confidence on smartphones may be improved by: (1) better data backup, lock, and wipe services; (2) new security indicators in smartphone application markets to increase user trust; and (3) user education and improved user interfaces to address common misconceptions about wireless network communication. Although the focus of the work is the analysis of the main concerns of users about security and privacy, it cannot be used to characterize the real security of mobile devices configurations. However, this work emphasizes the need for security indicators that increase user trust, which is precisely one of the goals of our work.

Ben-Asher et al. [10] surveyed 465 smartphone users regarding their security needs, awareness and concerns in the context of mobile phones usage. The work also evaluates acceptance and perceived protection regarding existing and novel authentication methods. Findings are that people consider some information sensitive (e.g. photos and contacts) and worry about physical attacks on their phones. Responses also show that users are interested in increased security and data protection. Although the work aims at proposing a two-level security model for mobile phones grounded in the users needs, it does not provide a concrete assessment of the current security of mobile devices.

The work from Shabtai et al. [77, 78] presents a comprehensive security assessment of the Google Android framework. They survey the security mechanisms of the Android platform (e.g. Linux mechanisms, application permissions and others) and analyze the existing security mechanisms, the effort to implement them and, finally, they present a qualitative risk analysis (Likelihood vs Impact) of the threats. This is a very interesting work that proposes several security countermeasures, but it does not consider the configuration of the device and does not propose an extensive taxonomy for the known Android configuration threats.

Mylonas et al. [63] surveyed the security awareness of smartphone users that install applications from official app repositories (e.g. Apple's App Store, Google Play, etc.). One of the goals is to provide an answer to the following question: "*Do smartphone users enable security controls on their devices?*". The findings suggest that the majority of users trust the app repository, security controls are not enabled, and users disregard security during application selection and installation.

The works presented above include several important concepts and give an insight into the strengths and weaknesses of existing security assessment studies of Android devices. However, these works (except PA [72]) are based on a manual process (mainly through interviews) that is prone to user error and is influenced by the experience of the user, which may distort the final result (the user may not understand all the security settings or even do not know how to assess them). Our study uses an automatic tool (Android application) for gathering and analyzing the configuration settings, providing key information for security assessment and improvement of mobile devices. We also highlight the most common vulnerabilities raised by user's security misconfiguration and propose countermeasures for the manufacturers, which is a clear contribution of our work (first row of the table).

2.4 Conclusion

This chapter presented relevant background and related works on the security of computer systems to help the reader following the ideas presented in the next Chapters. The works presented include several important concepts and give an insight into the strengths and weaknesses of existing security assessment proposals and benchmarks. A key observation is that most works use a manual or semi-automated process, which is prone to user error or influenced by the experience of the user. This may distort the final result, as the user may not understand all the security settings or even do not know how to assess them.

The works discussed include several concepts that are important to understand and follow the rest of this manuscript. Such works focus different aspects, but in general they are related with security assessment, security benchmarking, risk analysis, risk assessment and security configurations. Table 2.1 presents the relation among the main related works and the key subjects addressed in this thesis.

Table 2.1: Comparison between this work and the 7 most relevant related work.

	Security Tool	Analysis of field data	Users survey	Risk analysis	Security benchmark	Security configuration	Mobile Devices
Thesis	✓	✓		✓	✓	✓	✓
CIS benchmarks [27]					✓	✓	✓
Neto and Vieira [66]				✓	✓	✓	
Chin et al. [13]			✓				✓
Hogben and Dekker [39]				✓		✓	✓
Theoharidou [87]			✓	✓			✓
Shabtai [77]				✓			✓
Mylonas [64]			✓			✓	✓

As can be seen, the main differences and contributions of our work with regard to the state-of-the-art are the presence of a security tool (uSEA) and the analysis of field data. Comparing with the CIS benchmarks [27], our work provides a security configurations benchmark that is able to rank devices based on the risk analysis of each security configuration (instead of only providing guidelines for setting the configurations, as is the case of the CIS documents.

There is a clear gap in the literature with respect to methodologies to evaluate the security of mobile devices, mainly with respect to the user-defined security configurations. Just as the individuals share responsibility for their personal safety, the security of the information managed by a mobile device strongly depends on its user. End users do not appear to understand cybersecurity sufficiently to manage their security configurations. Therefore, researches and manufacturers should strive to improve the current scenario to avoid or reduce the perils of mobile device configurations.

Chapter 3

On user-defined security configurations

Vulnerabilities in the mobile devices environment rise from programming bugs, project defects and configurations issues [55]. This way, the identification and the assessment of security characteristics becomes a major issue to understand the effectiveness of existing security mechanisms and configurations. Moreover, it is important to know how the users configure their devices on the field, offering a better perception of the most common configurations issues.

This chapter presents an approach supported by a tool that allows assessing the security of Android devices based on the user-defined security configurations, which are known to be a key source of vulnerabilities. The tool automatically extracts 41 settings from the mobile devices under testing, 14 of which defined and proposed in this work and the remaining adapted from the well-known CIS benchmarks. The chapter discusses the settings that are analyzed, describes the high level architecture of the tool, and presents a preliminary evaluation that demonstrates the importance of this type of tools as a foundation towards the assessment of the security of mobile devices.

The outline of this chapter is as follows. Section 3.1 discusses the process for the identification of the user-defined security configurations that serve as basis for this work. Section 3.2 presents a tool for gathering the values for such configurations from devices. Section 3.3 presents an analysis of user-defined security configurations of real devices and Section 3.4 maps the configurations with known vulnerabilities and exploits. Section 3.5 proposes some countermeasures that can be applied to mitigate some vulnerabilities. Finally, Section 3.6 concludes the Chapter.

3.1 Identification of user-defined security configurations

In this work, a *user-defined security configuration* is defined as any setting, configuration or value that users can define or set and that have some impact in the overall security of the device. Examples of such configurations include: enabling a password to unlock the device; encrypt the device storage; and perform a data reset after a number of incorrect password tentatives. In practice, these configurations increase the effort that an attacker must make to retrieve information from a targeted device.

This work is based on a detailed analysis of the security recommendations proposed by the CIS security benchmarks for mobile devices, complemented by a set of settings identified as relevant. The two sets have been integrated in a list of user-defined configurations that can be applied to different mobile operating systems, and that has been used to support the implementation of uSEA (user-defined SEcurity configuration Assessment), our security assessment tool for the specific case of Android devices.

3.1.1 Concepts, definitions and procedure

Configurations are a great part of computer systems operations and are related with performing a customization according to the users, or generically, the environmental needs. With a customized system, it is possible to adapt to different purposes, for instance, an Internet banking can be adjusted by the user to present the stocks and investment funds that he wants to follow. Another example are modern operating systems, which have a great number of possible configurations and can be customized from setting a wallpaper to enabling cryptography in the device storage. In fact, configurations are related with the personalization of the system, thus enhancing the user experience.

Some configurations are irrelevant from a security perspective, for instance, changing the color of the system task bar or changing the wallpaper. On the other hand, there are a lot of configurations that have a direct impact in the system security, such as the definition of a strong password to log into the system, the encryption of the device storage and the setting of the right permissions to access files in the system.

Every system has a default set of configurations, which is typically referred as *default factory settings*. However, the user is able to change some of those configurations to better fulfill his requirements and environment. When these *user-defined configurations* impact the device security, then they become *user-defined security configurations*.

The process for gathering security configurations can be manual or automatic. The automatic analysis of user-defined configurations has several advantages when compared with the manual analysis. First, it is much simpler and faster. Second, automatic analysis is not prone to users errors, as is the case of manual analysis where user experience and misinterpretations may influence the security assessment results. Finally, using an automatic analysis allows assessing some configurations that cannot (or is impractical to) be accessed by manual means using the standard user interfaces. For example, to verify the presence of malware we need to verify the installed application signature on the device with regard to a database of known malware, anti-malware and rootkit applications (e.g. if the device has 15 installed applications and the database 700 malware signatures, 15x700 comparisons will be made only to verify the existence of malware), which is impractical to do by manual means. Nonetheless, manual analysis has also some advantages over the use of a tool. In fact, there are configurations that can only be accessed via the user interface and for which there is no API available. This is mainly due to the fact that some systems limit the access to sensitive data from user-level applications.

The literature regarding security configurations of mobile devices is vast and spread, as there is no standard or consensus on naming configurations or identifying them. Of particular interest are the CIS Security benchmarks [27], which list a great number of configurations, even if the provided documents are still incomplete and need to be complemented by other works [10, 13, 20, 50]. It is important to note that the set of available security configurations are always changing as new versions of mobile operating systems are released. This way, we are aware of the fact that the list of security configurations identified may become outdated for new versions of the operating systems. Nevertheless, a key aspect is that adding new settings to the list (or revising existing ones) is a task that would basically consist of repeating the same steps performed in this work.

3.1.2 CIS security settings

Four CIS security benchmarks [27], two designed for iOS devices (iOS 6 and iOS 7), and another two for Android devices (2.3 and 4.0), were analyzed in order to understand and extract all the relevant user-defined security configurations that should be considered when assessing the security of Android devices. As mentioned before, the CIS benchmarks provide a set of best practices, in the form of PDF documents, to help organizations assessing and improving their cyber security. The recommendations prescribed by CIS focus on the practical aspects of the configuration of the target systems and assert the correct value that each configuration option should have in order to enhance the overall security during the system usage.

Table 3.1 summarizes the 44 security settings collected from the four documents, dividing them into the following seven distinct categories (created to allow better understanding the specific characteristics and relevance of the settings):

- Application: this category is related to the applications installed on a mobile device, focusing on their type (normal, malware, antivirus, root) and on being updated;
- Browser: this category is related to the configuration of the user's default browser;
- Content: this category concerns about the data/information stored on a device;
- Network: this category is related to general communication settings;
- Password: this category concerns about the strength of passwords;
- Permission: this category is related to access permissions and encryption of the device storage;
- System: this category is related to the general system settings.

Note that this categorization is extremely useful when one wants to focus in a given subset of settings related to a specific configuration aspect and can also be used as a practical way to summarize the results of a given security assessment effort. For each setting the table shows in which document or documents it was found, which provides some insights on its potential generalization.

The specific configuration values for each setting are not shown in the Table 3.1 due to lack of space, but they can be found in the Appendix A. In practice, some values can also be inferred from the name of the recommendation (e.g. setting *enable password* means that a password must be set (enabled) on the device), while for others it is indeed

Table 3.1: Consolidation of security concerns from CIS security benchmarks.

ID	SECURITY CONCERNS	iOS6	iOS7	A2.3	A4.0
	APPLICATION				
A1*	Enable automatic app updates		✓		
	BROWSER				
B1	Disable JavaScript	✓	✓	✓	✓
B2*	Disable form auto-fill	✓	✓	✓	✓
B3	Disable cookies				✓
B4	Enable block pop-ups				✓
B5*	Disable plug-ins			✓	✓
B6*	Disable remember passwords	✓	✓	✓	✓
B7*	Enable basic SSL checks for websites	✓	✓	✓	✓
B8*	Turn on private browsing when needed	✓	✓		
	CONTENT				
C1*	Disable SMS preview when the device is locked	✓	✓	✓	✓
C2*	Disable apps views when the device is locked	✓	✓		
C3	Limit the number of text messages				✓
C4	Limit the number of multimedia messages				✓
	NETWORK				
N1	Remove WiFi entries	✓	✓	✓	✓
N2	Disable network notification	✓	✓	✓	✓
N3	Disable WiFi when not needed	✓	✓	✓	✓
N4	Disable bluetooth when not needed	✓	✓	✓	✓
N5*	Disable auto-join for all WiFi networks	✓	✓	✓	
N6	Enable airplane mode	✓	✓	✓	✓
N7	Turn off personal hotspot when not needed	✓	✓		
N8*	Turn off VPN when not needed	✓	✓	✓	
N9*	Turn off AirDrop discoverability		✓		
	PASSWORD				
P1	Enable password	✓	✓	✓	✓
P2	Require alphanumeric value	✓	✓	✓	✓
P3*	Disable passcode unlock for fingerprints		✓		
P4	Enable SIM card lock			✓	✓
P5	Do not make passwords visible			✓	✓
P6	Erase data upon excessive password failures	✓	✓	✓	✓
P7	Set a maximum password age				✓
P8	Set number of password history				✓
P9	Insert minimum passcode length	✓	✓	✓	✓
P10	Set minimum number of complex characters	✓	✓	✓	✓
P11	Set timeout minutes for sleep	✓	✓	✓	✓
	PERMISSION				
PE1	Enable encrypt phone				✓
PE2*	Encrypt credentials storage			✓	
	SYSTEM				
S1	Update firmware to latest version	✓	✓	✓	✓
S2*	Disable access to control center on lock screen		✓		
S3*	Erase all data before discard the device	✓	✓	✓	✓
S4	Disable developer options			✓	✓
S5	Disable unknown sources			✓	✓
S6	Disable location services when not needed	✓	✓	✓	✓
S7*	Enable “find my device” feature		✓		
S8*	Set security to disallow profile removal	✓	✓		
S9	Disable mock location			✓	✓
	TOTAL				
Total number of security concerns in each document		25	30	28	32

necessary to refer to the Appendix (e.g. the configuration *set timeout minutes for sleep* recommends that the device should lock after less than 90 seconds of inactivity). These values are used by the security assessment tool to assess how good (in security terms) the configuration of a mobile device under testing is. The user-defined settings gathered

Table 3.2: Presentation of new security concerns.

ID	SECURITY CONCERNS
APPLICATION	
A2	Do not install root applications
A3	Install anti-malware
A4	Do not install malware
CONTENT	
C5	Limit the number of photos
C6	Limit the number of audios
C7	Limit the number of videos
C8	Limit the number of documents
NETWORK	
N10	Disable ICMP Ping
N11	Disable bluetooth discoverability
PASSWORD	
P12	Enable pattern password
P13	Disable visible lock pattern
PERMISSION	
PE3	Disable write permission on /system
PE4	Disable write permission on /data
PE5	Do not root ("JailBreak")

by the tool are compared with the ones defined in the context of the CIS documents to identify potential security vulnerabilities.

As some configurations are only applicable to the iOS platform and due to the security characteristics of the Android architecture, the assessment of some settings (marked with an asterisk on Table 3.1) cannot be automated by a third party application. For instance, B2 (*disable form auto-fill*) is set through the Browser application and due to the Android sandbox, another application cannot read this information. As this may affect the final results, solutions should be researched in the future to allow gathering those settings (e.g. closer integration with the operating system design).

3.1.3 Additional security settings

Due to their potentially limited scope, the security assessment of the user-defined settings stated in the CIS benchmarks needs to be complemented to include additional key concerns to tackle more security issues. Table 3.2 presents the 14 security settings proposed in order to improve the CIS benchmarks, considering the same seven categories described in the previous section.

The reasoning behind each of the security settings proposed is as follows:

- *A2*: some applications can root (jailbreak) the device, causing loss of warranty and security breaches, as an application on a root device can access any information stored;
- *A3*: antivirus applications can detect and remove malware reducing the attack

vectors and attack surface of the device;

- *A4*: malware may disclose private information, causing financial loss and others;
- *C5*, *C6*, *C7* and *C8*: some photos, audios, videos and documents with sensitive or important information could be stored on the device;
- *N10*: answering to ping requests can give to an attacker the possibility of knowing the IP of the device, thus increasing the attack surface;
- *N11*: if Bluetooth access is invisible, the device becomes more difficult to find, reducing the attack surface;
- *P12*: although the use of a password pattern is less secure than use an alphanumeric password, it is better than not using a password mechanism;
- *P13*: a visible password pattern tracks the user motion while unlocking the device, thus making visible the track motion may disclose the password;
- *PE3* and *PE4*: due to its sensitivity, the system and data folder must have read only permission, having write permissions facilitates rooting the device and information corruption and disclosure;
- *PE5*: if the device is rooted (jailbreak) there is no security assurance as any application with root permission can access any information on the device.

In the particular case of *C5*, *C6*, *C7*, and *C8*, contents may be disclosed on a possible device theft or attack. The maximum number of items of each type is obviously open to discussion, but we considered 20 items. This value was retrieved from the CIS security benchmarks documents [27] that originally defined it as the limit for text and multimedia messages. Although the CIS documents do not contemplate other content types, we considered the same value for photos, videos, audio and documents. Anyway, this is the type of configuration that can be easily tuned in our tool.

The settings of the application and content categories were derived from the needs raised by the works [10, 13]; N10 and N11 were added confronting the current existing Android APIs for the network interfaces and the CIS security benchmarks; P12 and P13 were added after comparison between the CIS security benchmarks configurations related to password and the login methods of the Android devices; and PE3, PE4 and PE5 were added after analysis of mobile device's known threats [50].

We are aware of the fact that the list of security settings identified is not complete and may become outdated for new versions of the operating systems. Nevertheless, the proposed set is based on the well-known benchmarks from CIS, complemented with several settings defined by the authors. A key aspect is that adding new settings to the list (or revising existing ones) is a task that would basically consist of repeating the same steps performed in this work. The major issue at this point is automation, as already referred for a considerable set of settings defined in the CIS documents, for which advanced solutions need to be researched in the future (e.g. the integration of security assessment in the operating system itself).

3.2 The uSEA tool

The *User-defined Security Configuration Assessment* tool (uSEA) aims at evaluating Android devices based on the user-defined security configurations described above. The tool supports the active search for configuration issues that may lead to security vulnerabilities in the devices under testing, providing information that can be used for improving the security of the assessed targets and for assessing the proneness of those targets to security attacks. The tool is currently available on Google Play [92]) and was initially introduced in [93].

3.2.1 Tool architecture

To support the implementation of the security assessment tool, the two sets of recommendations presented in Tables 3.1 and 3.2. As discussed in Section 3.1.2, from the 44 security recommendations extracted from the CIS security benchmarks only 27 can be automatically assessed due to specific Android architectural concerns. These, on top of the 14 recommendations proposed in Section 3.1.3, lead to a total number of 58 relevant user-defined settings, from which 41 have been included in our security assessment tool for Android devices.

The uSEA is based on a client-server model in which the client runs in an Android device and the server in a cloud environment. The client extracts the user's configurations and presents the results of the security analysis to the user. The data collected from the device is sent to the server by invoking a web service that replies with the results of the analysis. The decision on using a cloud server is based on two aspects: 1) improved maintenance, as it is easier to deploy an web server update than the mobile application; and 2) storage of the data collected in a centralized repository that can be used for further research analysis. In terms of security, the collected data is only available through a database connection using a *ssh* login to the remote server and some configurations of network ports. Additionally, the data collected are anonymous as the unique information that allows correlating the misconfigurations with the source device is the AndroidID (a device unique identifier), but no information in the dataset is enough to identify the user that owns a device with a given AndroidID (i.e. the AndroidID is not sufficient to identify the user). Figure 3.1 depicts the uSEA's high level architecture.

The process consists of 6 steps:

1. The *Security Assessment App* extracts the security configurations from the device under testing, except the ones in the browser category, as listed in Tables 3.1 and 3.2;

2. The information collected is encapsulated into a SOAP message that is sent to the *Web Service* (WS);

3. The analysis of the extracted settings is performed by comparing the user-defined configuration values with the ones recommended for each setting from a security point of view;

4. The outcome of the analysis is stored in the *Database* for future analysis and also sent to the client application for being later displayed to the user;

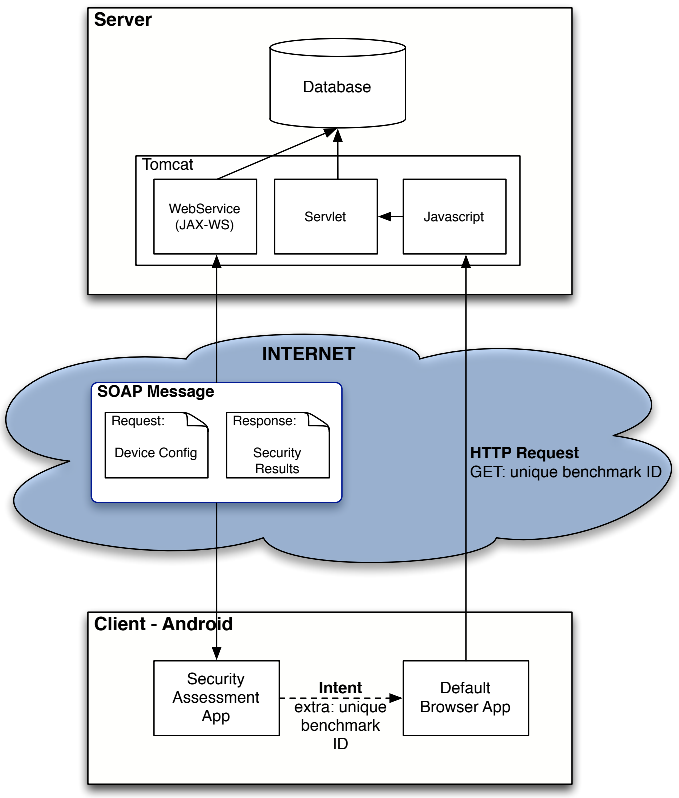

Figure 3.1: High Level Architecture of uSEA.

5. The *Default Browser App* is requested to open in the mobile device and to access the *JavaScript* web page in the server, in order to do a security analysis of the browser configuration;

6. The information collected is analyzed by the *Servlet*, stored in the *Databas*, and sent back to the mobile device for being displayed together with the analysis of the other settings (sent to the client in step 4).

An unique benchmark ID, also depicted in the HLA, is generated by the *WebService* when a new security analysis is started and returned to the *Security Assessment App* together with the WS response. The *Default Browser App* is opened via Intent (one way to do interprocess communication in Android) with this ID parameter, which is later sent via the GET method to the JavaScript web page, allowing linking the browser configurations with the device's configurations previously obtained. Although this unique ID changes when the security assessment tool is executed more than once in the same device (it is used to identify a single and unique security assessment result on the dataset), we are able to correlate the results from multiple executions, which allows understanding the evolution in terms of the security of configurations in the same device. Note, however, that for privacy reasons the information we have is not enough for tracking the data back to the users of the mobile devices tested.

The architecture of the tool was defined and implemented aiming at achieving an acceptable level of maintenance and extensibility, supporting the addition of new settings with little impact. In practice, it enables gathering new security concerns by performing simple modifications in some components, which is an important aspect as the tool should be extended when new needs emerge. For example, to extend the tool for another platform (e.g. iOS), the client app needs to be reimplemented (implementing the adequate mechanisms to gather the relevant settings). However, on the server side only minor tunings are required, mainly regarding the analysis functionality: the set of configurations may change (e.g. disable developer options does not apply for iPhone users), and the signatures for the malware, antivirus and root applications need to be updated as they are currently designed for the Android (nevertheless, extending to another platform requires only gathering the new signatures and updating a configuration file).

3.2.2 Collecting the security configurations

The uSEA application is available on Google Play, which means that any Android user can install the application and contribute to our dataset. When executing the application for the first time, the user must agree with our *terms of use agreement*, so we can collect the user data for research purpose. After the user agreement, they are encouraged to answer an optional survey that intends to outline his profile (personal information), including aspects like gender, age, highest education degree earned, field of action, and the level of experience on using smartphones and tablets. The age is divided into 6 categories: young (less than 25 years), young adult (25 to 34 years), adult (35 to 44 years), middle age (45 to 54 years), mature adult (55 to 64) and old-aged (65 years or more), following the recommendations presented in [81]. The highest education degree is divided into non graduate, graduate, master and PhD. The acting area is divided into 5 categories: agricultural sciences, biological and health sciences, exact and technological sciences, humanities and social sciences, and others. Finally, the experience on mobile devices usage is divided into: less than one year, between 1 and 2 years, between 2 and 5 years, and more than 5 years.

The security analysis is performed at the server side using the information received from the client application running in the mobile device. The security concerns in the *application* category are compared against a pre-defined package list of known malware, root and security applications. The list of malware was extracted from the *malgenome* project [105] that provides a large collection of 1260 Android malware samples. This dataset was analyzed and the packages were extracted in order to generate a list of known malware signatures, resulting in a set of 734 distinct packages. The list of security applications consists of applications such as mobile device management systems (MDM), password managers, malware detectors and removal, and any other application whose purpose is to bring security and/or privacy to the user. This list was created based on a thorough search for security applications at Google Play and several developer forums (e.g. xdadevelopers [104] and stackoverflow [68]), resulting in a total of 87 security applications (at the time of writing this work). For the root applications, we used a similar methodology and considered any application that intends to root an Android device or provide any information about how to do it, which resulted in the identification of 16 root applications. We are aware that these lists may be incomplete thus leading to the possibility of false negatives (e.g. not reporting the existence of a malware because it is not included in our list), but extending them consists of simply adding entries to a configuration file. Also, this approach may lead to some false positives (e.g. reporting a legitimate application as being a malware) due to the existence of malware in our list which was implemented using the same name of a legitimate application.

In the *browser* category, several security settings cannot be gathered on Android, because all applications (including the web browser) run inside the Application Sandbox. This means that the application data and code execution are isolated from other applications. In other words, the browser application runs in a secure environment, so other processes on the system, including our security assessment tool, cannot access its code or private data. The solution found to extract and validate some of the security settings of the default web browser is to run a web page that executes client-side code written in JavaScript. This allows assessing if the JavaScript (B1), cookies (B3), and pop-ups (B4) are enabled (or not) on the browser configuration.

For the recommendations in the *content* category, we used an Android feature called content provider for Honeycomb and above versions and implemented a method to count the number of items for the other versions. Content providers manage access to a structured repository of data, but are available only in the latest versions of Android. They encapsulate the data, and offer mechanisms for defining data security aspects for others applications, and are the standard interface that connects data in one process with code running in another process [25]. The Android system holds all the text and multimedia messages, videos, photos and audios on distinct content providers making it easy and fast to retrieve the number of items of each type. The method developed for the older versions identifies and counts as documents pdf files, presentations, spreadsheets and text documents (except ".txt" files that are mainly used for storing application and system configurations, and not for user data).

For the *password* security concerns, we used the Device Administration API that includes a vast number of methods to access the Android password policies. Unfortunately, to have access to this API the security assessment tool needs to request administration permission during its first execution. For the other categories (*network*, *permission* and *system*), the data was collected using APIs like *WifiManager* (to gather WiFi configurations), *BluetoothAdapter* (in charge of get the Bluetooth settings), *Settings.System* (containing miscellaneous system preferences), and others.

In summary, from the 44 security recommendations extracted from the CIS security benchmarks only 27 can be automatically assessed due to specific Android architectural concerns. These, on top of the 14 settings proposed by the authors, lead to a total number of 58 relevant user-defined configurations, from which 41 could be included in the uSEA. However, because the security configurations on the browser category are stored in the context of the default web browser application, the solution implemented by our tool requires running a web page that executes client-side code, thus assessing these configurations requires the user authorization to open the web browser.

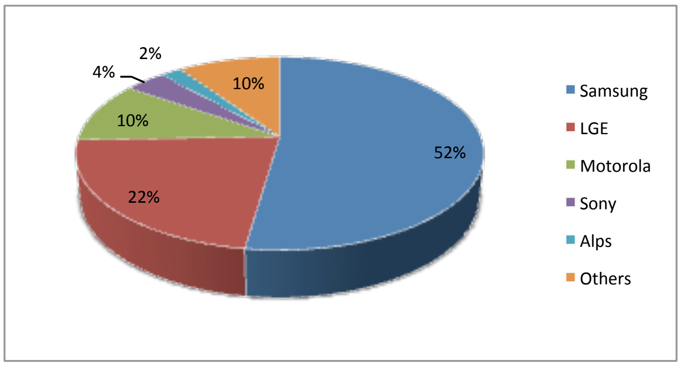

Figure 3.2: Manufacturers share.

3.3 Case Study: Analysis of user-defined security configurations of real devices

Delegating to the end-users the configuration of security settings of mobile devices leads to a large number of vulnerabilities. In this context, it is extremely important to understand the most common misconfigurations and, specially, propose ways to mitigate the corresponding vulnerabilities and threats. This section discusses in detail our observations regarding the security configurations of 561 Android devices.

3.3.1 Data gathering and sample space

Approximately one third of users in our dataset of 561 devices responded to the personal information survey. Among those, the most common profiles are male (66%), have less than 25 years (51%), are graduated (47%), work or study in a area different from the ones listed by the application (57%), and there is no major significance in the category of experience with mobile devices. Although this information is relevant to understand the sample set, the focus of this work is to understand the overall Android security configuration issues. This way, the concrete user profiles are not taken into account, i.e. the profile of the user is not related to the misconfigurations observed in his device, as presenting such an analysis would require a much larger sample space.

Although during the period of our study we gathered data from 561 devices, the sample size for the browser category is only of 419 users (some users did not allow the execution of the default web browser during the security tests). The overall sample space market share of the manufacturers is depicted in Figure 3.2. The numbers indicate a predominance of Samsung devices (52%), but there are also other manufacturers with a great portion of the sample space: LGE (22%), Motorola (10%), Sony (4%) and Alps (2%). The relative number of devices running a given version of the Android platform is depicted in Figure 3.3, Jelly Bean is the most used platform (46%), followed by Gingerbread (26%), KitKat (15%) and Ice Cream Sandwich (12%).

Related works that evaluated the user awareness on security of mobile devices used a sample space smaller than our's. In fact, Chin et al. [13] surveyed 60 smartphone users, Ben-Asher et al. [10] surveyed 465 users and Mylonas et al. [63] sample population's

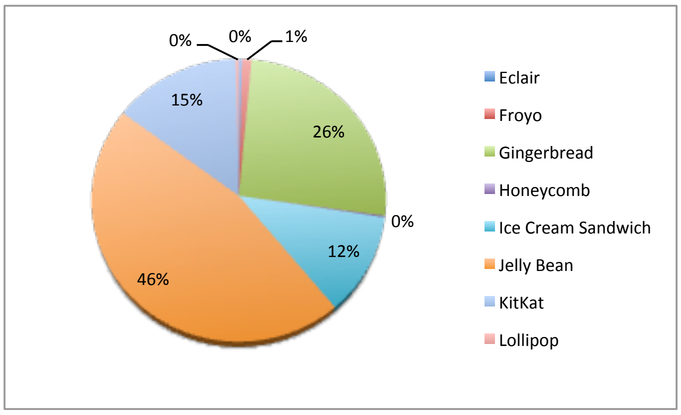

Figure 3.3: Android platform versions.

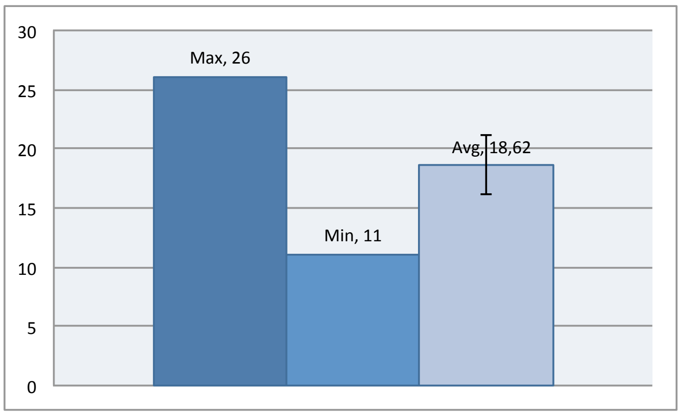

Figure 3.4: Analysis of users safe security configurations number.

includes 458 smartphone users. This gives us some confidence that the observations presented in this study may hold in a generic manner.

3.3.2 Overall results

An overall analysis of the number of correctly set configurations (from a security point of view) in the 561 devices analyzed is shown in Figure 3.4. As we can see, in average, only 18 out of 38 security configurations are correctly set (the browser configuration was not included here due to the limitations discussed in Section 3.3.1). Also, the highest number of correct configurations in a device is 26, which suggests that even the users most concerned with security aspects fail at setting a large number of configurations on their devices (at least 12 in the sample set analyzed). On the other hand, the worst case has only 11 configurations correctly set. This analysis reinforces our argument that users need to become more aware of the security implications of the configurations they set and the manufacturers should limit the user intervention on the security configurations of mobile devices and provide best ways to help users make informed security decisions.

Figure 3.5 presents the results aggregated per category, the "unsecure" label is related to the security configuration number that is being neglected by the users, in other words, that is misconfigured, while the "secure" label means that the security configuration is respected and well set in the security point of view. As shown, the browser and password

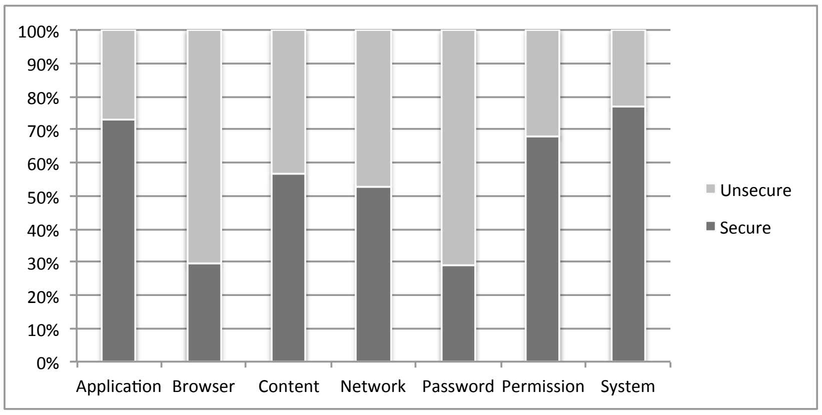

Figure 3.5: Analysis of the configurations of all categories.

category have, in average, less than one third of the configurations correctly set. Also, for the content and network categories, a little more than half of the configurations are correct (the observations for the remaining categories go in the same line, being the system category the one with a large proportion of correctly set configurations). Again, we can argue that a large number of security configurations is being neglected by the users, and that this is largely independent from the category under analysis. Although this generic analysis is important to give a first insight about the security awareness of end-users, in the next sections we provide a fine-grained analysis considering each security configuration individually.

3.3.3 Application category

The configurations in the application category are related to the applications installed on the mobile devices, focusing on malware, root exploits ("jailbreaks") and security applications. This way, the analysis in this category is based on the comparison of the signature of the applications installed in each device with a list of known malware, root exploits and security applications' signatures (already discussed in Section 3.2).

Figure 3.6 depicts the percentage of users that have malware, root exploit and/or security applications installed (or not) on their devices. A first observation is that only 36% of the users have a security application installed, which is a major concern as in 2013 mobile malware was almost exclusively focused on the Android platform [85] and the volume of Android variants increased by 40 percent in 2015 [86]. The installation of security applications can mitigate the risks raised by these malicious applications, that is the case of malware detectors. Another important aspect is that 11% of the devices analyzed had some kind of malware installed. This is a key problem as the most common goals of malware are to collect private user information (61%) and to send premium-rate SMS messages (52%), leading to substantial financial loss [25]. On the other hand, only a small percentage of the users (5%) have root exploit applications on their devices, which is an important and positive aspect, as root exploits are frequently used by malware authors for bypassing security mechanisms and by smartphone owners for customizing their devices (such customizations frequently affect the security of a device) [25].

Our analysis confirms something that some researches frequently suggest: smartphone

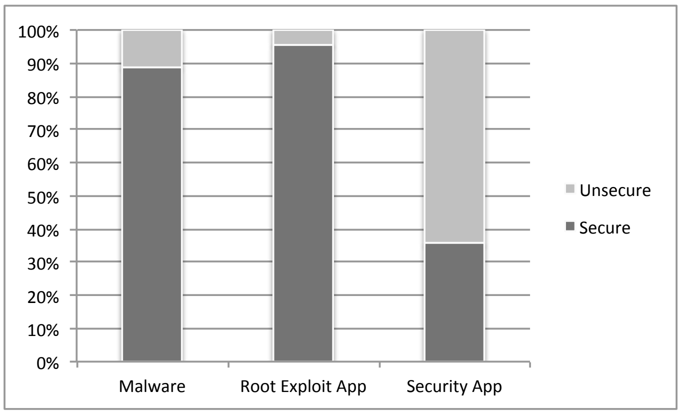

Figure 3.6: Analysis of the configuration in the application category.

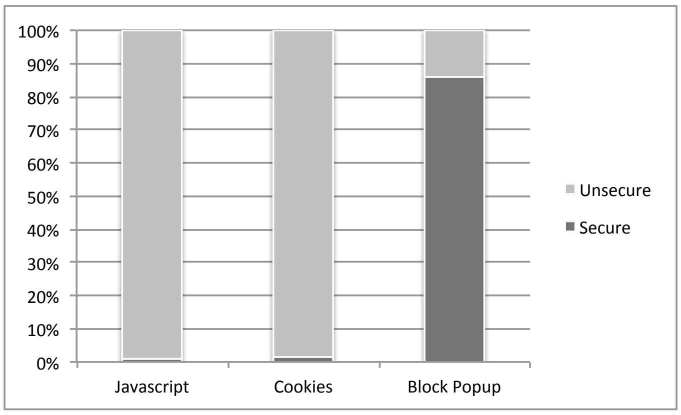

Figure 3.7: Analysis of the configuration in the browser category.

security software is more likely not to be installed, as users tend to be unaware of it [63]. On the other hand, the low percentage of devices with root exploits installed indicates that only a few users are open to customize their devices.

3.3.4 Browser category

The browser category is related to the configuration of the user's default browser. A third party Android application is not able to assess the configurations values of the Browser application due to Android platform restrictions, so it was necessary to assess those configurations through a client-side page.

Figure 3.7 depicts the configurations issues regarding this category. Note that there is a large usage of cookies and JavaScript, which is perfectly understandable because a large number of web sites requires these settings to run properly. The configuration that calls more attention is the one related to blocking pop-ups as 14% of users have this feature disabled. The problem is that pop-ups are frequently used to open untrusted malicious web sites [27] and also for performing phishing attacks [103].

The user's default browser security configurations do not vary much from the ones that come by default on Android devices, which have JavaScript, cookies and block pop-ups enabled. This suggests that most users are unaware or indifferent to the inherent browser security issues and just use the default configurations set by the manufacturer.

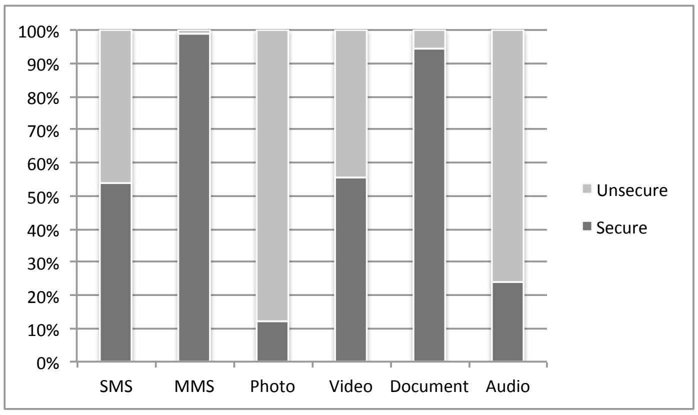

Figure 3.8: Analysis of the configuration in the content category.

3.3.5 Content category

The configurations in the content category are related with the data/information stored on the device, which may be disclosed due to a device theft or cyber attack. The maximum number of items recommended for each type of data/information is obviously open to discussion. In this study we consider 20 items, a value retrieved from the CIS security benchmarks documents [27] that originally defined it as the limit for text and multimedia messages. Although the CIS documents do not contemplate other content types, we consider the same value for photos, videos, audio and documents. Again, this is the type of configuration that can be easily tuned in our tool.

Figure 3.8 presents the results for the content-related security configurations. As we can see, there are several problems, specially in what regards photos and audio files (the average number of photos and audios on the devices are respectively 392 and 150), which increases the chances of having sensitive information on these files. Another concerning issue is the number of text messages and videos (in average, 91 and 50 respectively) suggesting that the users keep an amount of information on their devices that increases the chances of having sensitive data leaked in an eventual cyber attack or physical theft. The multimedia messages (MMS) are less used, probably because it's cheaper and faster to send media by others means, nowadays there are a plenty of application able to supplant MMS capabilities.

The analysis of this category confirms that the majority of the Android users store a large amount of information in different formats (audio, video, text and image), which makes them a storage of sensitive information and, consequently, a valuable target for malicious minds. Note that the analysis is performed from a "hard" security point of view (either it is secure or not), but people may need to have more than 20 photos and audio files in their smartphones (that's reasonable when you use it to take pictures and listen to music), and so on. Thus, if all security parameters are set to the "secure" configuration, smartphones maybe lose their great utility. The important aspect is that most users disregards security aspects carrying in their smartphones a large amount of information.

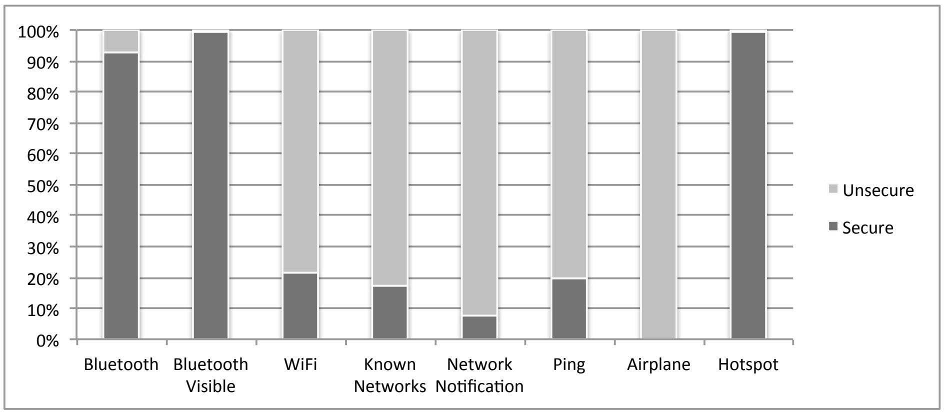

Figure 3.9: Analysis of the configuration in the network category.

3.3.6 Network category

The network category is related to general communication settings. This category is very important as most devices support multiple ways to connect to a network or the Internet, thus increasing the attack surface.

Figure 3.9 summarizes our observations. As shown, most users have pre-configured WiFi networks (see *known networks* configuration) stored on their devices (83%) and also use the WiFi network as the primary mean for connecting to the Internet (78%). Disabling the WiFi interface when not using it reduces the attack surface of the device. Additionally, the cellular data network interface is more difficult to sniff than the WiFi interface [27]. The network notification is another issue. When it is enabled and a new network becomes available, an icon appears on the status bar, which in turn shows a list of available networks from which the user can choose. However, requiring the user to manually configure and join a WiFi network reduces the risk of inadvertently joining a similarly named yet untrusted network (e.g. "linksis" vs. "linksys") [27]. The ICMP echo reply (ping) is activated in some devices (19%), which can lead to the device status discovery (e.g. connected or disconnected). Regarding the airplane mode, the users are not enabling this control (as expected), as it determines whether the devices can handle radio signals, thus being used only in specific cases.

The network category is quite threatening as much information can be extracted using the network interfaces. It is clear that the best practices for the security configurations on this category have a poor adoption by the users, specially in what regards the usage of WiFi networks. This is understandable as the large usage of WiFi is related to its lower cost and better performance when compared to a cellular network data service.

3.3.7 Password category

The password category, as aforementioned in Figure 3.5, are the most concerning category in which less than one third of the configurations correctly set. Figure 3.10 shows the results for the password related configurations. Before analyzing the data, it is important to note that some of these configurations cannot be set by the users via the standard Android interface. For example, the number of password failures allowed before wiping

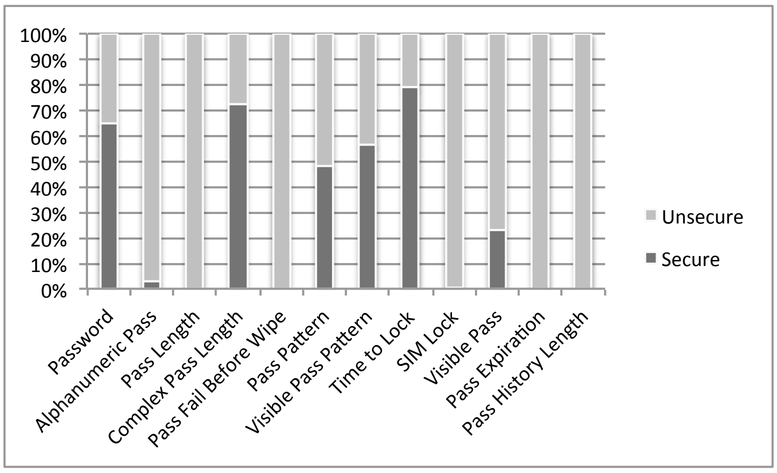

Figure 3.10: Analysis of the configuration in the password category.

the device data, the password history length, and the password expiration date can only be set using third party applications with administrator permission. The results show that the users do not configure those settings because they can not do that by simple means.

Another important security configuration is the use of a password: although most users have a password configured (65%), a big part of them do not use any login mechanism. Additionally, the Android pattern password is more used than the alphanumeric one, but the pattern password is considered more weak because it has a very limited number of combinations (389,112 possible patterns [9]) and can be easily observed by a malicious person or even stressed via brute force methods. The fact that, in America, 1 out of 10 smartphone users are victims of phone theft [53] increases the attention to this configuration as a phone with no password offers no barrier and make it easier for the thief to retrieve any information stored on the device.

The "*SIM Lock*" configuration is another example of a security setting that the user can manage, however as we can see, there is a poor adoption of this mechanism. SIM cards often contain contact and other personal information. This setting will lock the SIM card so that it requires a PIN to access. Parties who do not know the SIM PIN should not be able to view the SIM card's contents, nor use the SIM card in another mobile device.

From this analysis we can state that most users neglect password configurations, which is due to the fact that some configurations cannot be set through the Android user interface. Thus, manufactures should rethink their approach, and provide security tools that facilitate password management and support advanced and easy authentication methods (e.g. finger print authentication) [10].

3.3.8 Permission category

The permission category is related to the encryption of the device storage and also to the access permissions to some Android platform folders. As shown in Figure 3.11, none of the tested devices had storage encryption configured. This is expected as most users are not aware of such feature and also because it is a time consuming process that decreases

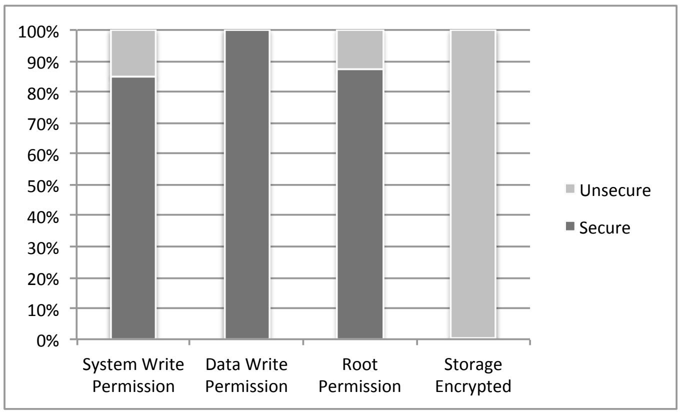

Figure 3.11: Analysis of the configuration in the permission category.

the device performance. However, this also demonstrates the user indifference regarding this setting and suggests that manufacturers should rethink or re-implement this feature since it is not being used in practice.

Another interesting security configuration is the root permission that shows that 13% of our sample space users have their devices rooted, also known as jailbroken: this value is similar to another study that estimates that 15% to 20% of Android phones are rooted [75]. This configuration is of extreme importance and, as aforementioned, it is coveted by two groups of people: malware authors and smartphone users [25]. While the malware authors may try to root a device to gain extra privileges and perform any operation on the phone, users want to install customized versions of operating system (which also opens the door for malicious applications) [25]. The percentage of jailbroken devices is low because the process to jaibreak a device requires an expertise in this subject. It is important to note that many users, regardless of security consequences, root their devices or flash customized Android builds to customize their phones.

It is also important to note that some devices (15%) have write permission to the System folder which contains the Android system, other than the kernel and the ramdisk. This includes the Android user interface as well as all the system applications that come pre-installed on the device. Wiping this partition will remove Android from the device, but it will still be possible to put the phone into recovery or bootloader mode and install a new Android build. The permission of the System folder is very important as with a write permission an user can inadvertently be able to make changes that he/she should not do, such as, install applications with system's privilege.

3.3.9 System category

The system category is related to the general system settings. Results in Figure 3.12 show some important issues as several users have their Android firmware outdated. In fact, we observed that only approximately 21,5% percent of the Android devices have the latest firmware version installed. This result is of great importance, because its calls forth a common and well-known problem of the Android platform: fragmentation. Although the operating system that runs on different devices from different manufacturers is the same, the manufacturers frequently do some customizations putting additional applications,

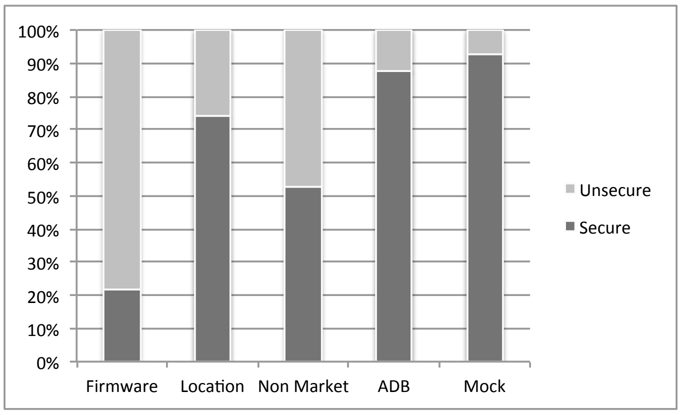

Figure 3.12: Analysis of the configuration in the system category.

carrier specifications, set of wallpapers, ringtones and etc. Consequently, when Google releases a new Android version, the manufacturers should re-implement (or reuse) their own customizations and this demands effort, time and budget, leading some devices to become obsolete when their manufacturers stop supporting them.

Another important configuration is enabling the installation of applications that are not obtained from the Google's official store, which may expose the user to malware [25]: we observed that 47% of the users enable this configuration, which is in our opinion an extremely large number. Although, malware and adware are still found in Google Play, it helps (or tries to) protect the device by blocking potentially harmful apps (before an app is published to Google Play, it is reviewed to make sure it is not harmful).

A key recommendation is to enable the GPS (Global Positioning System), Mock location and ADB (Android Debug Bridge) only when necessary as these allow to, respectively, disclose the user's location and increase the attack surface (e.g. by connecting to the device using an USB cable). The problem is that, although many users have these settings disabled, there is a great number that neglect them suggesting that they are unaware of the security risks faced.

3.4 Mapping configurations with known vulnerabilities and exploits

Understanding real vulnerabilities that are related with user-defined configurations is a key aspect to motivate users to take such configurations seriously. Also, it is of major relevance for leading the manufactures to rethink their policies in terms of the security configurations that can be modified by the users. In this section we discuss several well known vulnerabilities and exposures of Android devices, correlating them with user-defined security configurations that could avoid or mitigate such vulnerabilities.

Based on the most common attacks reported in [28] and the known literature [11,25,50, 78,96], we conducted a search on the Common Vulnerabilities and Exposures (CVE) [60] database, searching for vulnerabilities that can be mitigated or even avoided through the correct configuration of the device. The process followed was as follows:

1. Search for keywords that are similar to the name of known attacks plus the word Android. For example, for the Browser exploit, we searched the following keywords: "*Android browser android*";

2. Exclude the results that are specific for very old Android versions (1.0, 1.1 and 1.5) and the ones raised by specific applications (or specific versions of applications) that are uncommon, thus not possible to find in our data set;

3. Choose the 3 most recent entries. As the CVE contains a huge dataset and some searches retrieved thousands of vulnerabilities, we opted to focus on the 3 most recent ones to make our study feasible.

We are aware that there are other common attacks that are not considered in our analysis. For instance, we do not considered: 1) *zero day attacks*, as they are related to attacks that are performed via unknown/unreported vulnerabilities, thus they are not yet in the CVE database; 2) *spamming attacks*, as the entries in the CVE database are related to very specific versions of Android applications, thus not being considered in our approach; and 3) *device theft or loss*, as this is related to physical attacks (also out of the scope of our work).

The 19 vulnerabilities and exposures presented in this section demonstrate the impact of the user-defined configurations on the device security. Great part of them can be avoided or mitigated by correctly setting 2 or 3 configurations, in special by updating the device firmware to the latest version. This way, this analysis reinforces the idea that manufacturers should provide best ways to help users making informed security decisions.

3.4.1 Browser Exploit

Some exploits are designed to take advantage of vulnerabilities in the software browsers used to access websites [28]. Using the keywords "*Android browser exploit*" we found 46 related entries from which the following three vulnerabilities and exposures were selected for analysis:

> CVE-2015-1275: Cross-site scripting (XSS) vulnerability in `org/chromium/chrome/browser/UrlUtilities.java` in Google Chrome before version 44.0.2403.89 on Android allows remote attackers to inject arbitrary web script or HTML via a crafted intent: URL, as demonstrated by a trailing alert(document.cookie);// substring, aka "Universal XSS (UXSS)."

The CVE-2015-1275 is related to specific versions of the Chrome browser, corresponding to 314 devices in our data set. To reproduce this vulnerability, one needs to execute a block of javascript code, which can be avoided by correctly setting configuration B1 (*Disable JavaScript*). Fig. 3.13 shows the percentage of devices that are exposed to this vulnerability (313 out of 561), i.e. the ones that have Google Chrome before 44.0.2403.89 and javascript enabled. In practice, only one of the 314 devices running the vulnerable Chrome versions was not vulnerable as the other 313 devices have JavaScript enabled.

> CVE-2015-1261: The `android/java/src/org/chromium/chrome/browser/WebsiteSettingsPopup.java` in Google Chrome before 43.0.2357.65 on Android does not properly restrict the use of a URL's fragment identifier during construction of a page-info popup, which allows remote attackers to spoof the URL bar or deliver misleading popup content via crafted text.

As before, also the CVE-2015-1261 is for specific versions of the Chrome browser, corresponding to 314 devices in our data set. Reproducing this vulnerability requires opening pop-ups in the targeted device, which can be avoided by the correct configuration of B4. Fig. 3.13 shows the percentage of devices that are exposed to this vulnerability (32 out of 561): the ones that have Google Chrome before version 43.0.2357.65 and block pop-up disabled.

> CVE-2015-1211: The OriginCanAccessServiceWorkers function in `content/browser/service_worker/service_worker_dispatcher_host.cc` in Google Chrome before `40.0.2214.111` on Windows, OS X, and Linux and before 40.0.2214.109 on Android does not properly restrict the URI scheme during a ServiceWorker registration, which allows remote attackers to gain privileges via a filesystem: URI.

The CVE-2015-1211 versions affected by this vulnerability correspond to 309 devices in our data set. To reproduce it, one needs to execute javascript, open a popup or use cookies. This way, there is no user-defined configuration that can avoid this vulnerability. However, the consequence of this vulnerability is to gain access privileges to the device filesystem, which can be mitigated by correctly setting the configurations in the content category. Fig. 3.13 shows the percentage of devices that are exposed to this vulnerability (309 out of 561): the ones that have Google Chrome version before 40.0.2214.109.

3.4.2 Data Interception

Data interception may occur when an attacker is eavesdropping on communications originating from or being sent to a mobile device [28]. Using the keywords "*Android man in the middle*" we retrieved 1421 entries, from which we select the following vulnerabilities and exposures for a more detailed analysis:

"CVE-2016-1948: Mozilla Firefox before 44.0 on Android does not ensure that HTTPS is used for a lightweight-theme installation, which allows man-in-the-middle attackers to replace a theme's images and colors by modifying the client-server data stream."

The CVE-2016-1948 is for specific versions of Mozilla Firefox, corresponding to 17 devices in our data set. This vulnerability can be exploited by a man-in-the-middle attack, which is easier and more likely to happen in WiFi networks than in the cellular data network. To prevent this exploit, users should correctly set the N3 (*Disable Wi-Fi when not need*) and N6 (*Enable airplane mode*) configurations. Fig. 3.13 shows the percentage of devices that are exposed to this vulnerability (15 out of 561), which are the ones that have Mozilla Firefox before version 44.0 and incorrectly set N3 and N6 configurations.

> CVE-2008-7298: The Android browser in Android cannot properly restrict modifications to cookies established in HTTPS sessions, which allows man-in-the-middle attackers to overwrite or delete arbitrary cookies via a Set-Cookie header in an HTTP response, related to lack of the HTTP Strict Transport Security (HSTS) includeSubDomains feature, aka a "cookie forcing" issue.

The CVE-2008-7298 uses a man-in-the-middle attack to modify cookies in the target device. This way, configurations B3 (*Disable cookies*), N3 (*Disable Wi-Fi when not needed*) and N6 (*Enable airplane mode*) can prevent this exploit. Fig. 3.13 shows the percentage of devices that are exposed to this vulnerability (430 out of 561).

Although there are plenty of vulnerabilities that can be exploited by man-in-the-middle attacks, the great part of them are related to specific applications (therefore they were not considered in our analysis) that do not verify X.509 certificates from SSL servers. This allows man-in-the-middle attackers to spoof servers and obtain sensitive information via a crafted certificate.

3.4.3 Malware

Malware is any malicious software that can initiate a wide range of attacks and spread itself onto other devices [28]. Using the keywords "*Android malware*" we retrieved the following 3 entries:

> CVE-2011-0638, CVE-2011-0639 and CVE-2011-0640: Linux, Mac OS x and Windows do not warn the user before enabling additional Human Interface Device (HID) functionality over USB, which allows user-assisted attackers to execute arbitrary programs via crafted USB data, as demonstrated by keyboard and mouse data sent by malware on a smartphone that the user connected to the computer.

The three vulnerabilities (CVE-2011-0640, CVE-2011-0639 and CVE-2011-0638) are related to the operating systems that are connect to an Android device through a USB cable. This can be avoided by disabling the ADB (S4). Fig. 3.13 shows the percentage of devices that are exposed to this vulnerability (68 out of 561), which correspond to the ones that incorrectly set the S4 (*Disable developer options*) configuration.

3.4.4 Unauthorized Location Tracking

Location tracking allows knowing and monitoring the position of registered mobile devices [28]. The use of the keywords "*Android location*" resulted in 8 entries from which we select the following:

> CVE-2014-0974: The boot_linux_from_mmc function in `app/aboot/aboot.c` in the Little Kernel (LK) bootloader, as distributed with Qualcomm Innovation Center (QuIC) Android contributions for MSM devices and other products, does not properly validate a certain address value, which allows attackers to write data to a controllable memory location by leveraging the ability to initiate an attempted boot of an arbitrary image.

> CVE-2013-4737: The CONFIG_STRICT_MEMORY_RWX implementation for the Linux kernel 3.x, as used in Qualcomm Innovation Center (QuIC) Android contributions for MSM devices and other products, does not properly consider certain memory sections, which makes it easier for attackers to bypass intended access restrictions by leveraging the presence of RWX memory at a fixed location.

For the exploitation of CVE-2014-0974 and CVE-2013-4737 one needs to be able to upload an arbitrary image to the phone and start the boot process. In other words, it is a low-level vulnerability that cannot be avoided by any of the known user-defined security configurations.

> CVE-2012-6334: The Track My Mobile feature in the SamsungDive subsystem for Android on Samsung Galaxy devices does not properly implement Location APIs, which allows physically proximate attackers to provide arbitrary location data via a "commonly available simple GPS location spoofer."

The CVE-2012-6334 is a vulnerability specific for Samsung Galaxy devices, corresponding to 280 devices in our data set. This vulnerability can be exploited by an attacker that has physical control of the device, which can be avoided by the correct configuration of the settings in the password category. Fig. 3.13 shows the percentage of devices that are exposed to this vulnerability (91 out of 561), which corresponds to the Samsung Galaxy devices that have such configurations incorrectly set.

3.4.5 Network exploits

Network exploits take advantage of software flaws in the system that operates on local (e.g. Bluetooth, WiFi) or cellular networks [28]. Using the keywords "*Android bluetooth*" and "*Android WiFi*" we retrieved 15 entries. Next we analyse some in detail:

"CVE-2015-6618: Bluetooth in Android 4.4 and 5.x before 5.1.1 LMY48Z allows user-assisted remote attackers to execute arbitrary code by leveraging access to the local physical environment."

The CVE-2015-6618 is for specific versions of the Android firmware, corresponding to 87 devices in our data set. To avoid this vulnerability, beyond updating the firmware, the user can disable the bluetooth (N4). Fig. 3.13 shows the percentage of devices that are exposed to this vulnerability (6 out of 561), which correspond to the ones that have the vulnerable firmware and bluetooth enabled.

"CVE-2015-5310: WiFi in Android before 5.1.1 LMY49F and 6.0 before 2016-01-01 allows remote attackers to obtain sensitive WiFi information by leveraging access to the local physical environment."

As above, the CVE-2015-5310 is also for specific versions of the Android firmware, but in this case it corresponds to all the 561 devices in our data set. To avoid this vulnerability, beyond updating the firmware, the user can disable the WiFi (N3). Fig. 3.13 shows the percentage of devices that are exposed to this vulnerability (434 out of 561).

3.4.6 Abuse of costly services and functions

The keywords "*Android charges*" retrieved only the following entry:

> CVE-2014-8610: AndroidManifest.xml in Android before 5.0.0 does not require the SEND_SMS permission for the SmsReceiver receiver, which allows attackers to send stored SMS messages, and consequently transmit arbitrary new draft SMS messages or trigger additional per-message charges from a network operator for old messages, via a crafted application that broadcasts an intent with the `com.android.mms.transaction.MESSAGE_SENT` action.

The CVE-2014-8610 allows abusing of costly services and functions and is able to resend SMS messages due to a vulnerability in the MMS application in the Android platform. Although this can not be avoided by a user-defined security configuration, it can be mitigated if the user correctly sets configurations C3 (*Limit the number of text message*) and C4 (*Limit the number of multimedia message*). Fig. 3.13 shows the percentage of devices that are exposed to this vulnerability (256 out of 561).

3.4.7 Phishing and Spoofing

Phishing and spoofing attacks are similar, although phishing is used to deceive people to disclose sensitive information and in spoofing attackers may create fraudulent websites to mimic or "spoof" legitimate sites (and in some cases use those fraudulent sites to distribute malware to mobile devices [28]). Using the keywords "*android phishing*" we retrieved 4 entries from which we selected the following for analysis:

> CVE-2014-8609: The addAccount method in `src/com/android/settings/accounts/AddAccountSettings.java` in the Settings application in Android before 5.0.0 does not properly create a PendingIntent, which allows attackers to use the SYSTEM uid for broadcasting an intent with arbitrary component, action, or category information via a third-party authenticator in a crafted application.

The CVE-2014-8609 enables an attacker to craft an application that can create a phishing SMS in the phone. The correct configuration of S1 (update firmware to latest version) allow avoiding this vulnerability. Fig. 3.13 shows the percentage of devices that are exposed to this vulnerability (552 out of 561), which correspond to the ones that run Android before version 5.0.0.

"CVE-2014-1527: Mozilla Firefox before 29.0 on Android allows remote attackers to spoof the address bar via crafted JavaScript code that uses DOM events to prevent the reemergence of the actual address bar after scrolling has taken it off of the screen."

The CVE-2014-1527 is for specific versions of Mozilla Firefox, corresponding to 3 devices in our data set. This vulnerability can be exploited by means of crafted JavaScript code. To avoid its exploitation, users need to correctly set configuration B1 (*Disable JavaScript*).

3.4.8 Denial of Service

A Denial of Service attack is the attempt to make a computational resource unavailable. Using the keywords "*android denial of service*" we was retrieved 203 entries. The following exemplify the observations and how user-defined configurations may help (or not) mitigating the vulnerabilities:

> CVE-2016-0803: libstagefright in mediaserver in Android 4.x before 4.4.4, 5.x before 5.1.1 LMY49G, and 6.x before 2016-02-01 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted media file that triggers a large memory allocation in the (1) SoftMPEG4Encoder or (2) SoftVPXEncoder component.

The CVE-2016-0803 impacts 374 devices in our data set. There is no user-defined configuration (except updating the firmware) that can avoid this vulnerability. Fig. 3.13 shows the percentage of devices that are exposed to it.

> CVE-2016-0802: The Broadcom WiFi driver in the kernel in Android 4.x before 4.4.4, 5.x before 5.1.1 LMY49G, and 6.x before 2016-02-01 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via crafted wireless control message packets.

The CVE-2016-0802 is for specific versions of the Android firmware, corresponding to 374 devices in our data set. To avoid this vulnerability, beyond updating the firmware, the user can disable the WiFi (N3). Fig. 3.13 shows the percentage of devices that are exposed to this vulnerability (289 out of 561).

3.4.9 Privilege Escalation

Privilege escalation is pursued with the intention of gaining access privileges over more resources than normally allowed. Using the keywords "*android privilege*" we obtained 66 entries, from which we select the following examples:

> CVE-2016-0810: media/libmedia/SoundPool.cpp in mediaserver in Android 4.x before 4.4.4, 5.x before 5.1.1 LMY49G, and 6.x before 2016-02-01 mishandles locking requirements, which allows attackers to gain privileges via a crafted application, as demonstrated by obtaining Signature or SignatureOrSystem access.

The CVE-2016-0810 is for specific versions of the Android firmware, corresponding to 374 devices in our data set. There is no configuration (except updating the firmware) that can avoid the this vulnerability.

"CVE-2016-0806: The Qualcomm WiFi driver in the kernel in Android 4.x before 4.4.4, 5.x before 5.1.1 LMY49G, and 6.x before 2016-02-01 allows attackers to gain privileges via a crafted application."

The CVE-2016-0806 is for specific versions of the Android firmware, corresponding to 374 devices in our data set. To avoid this vulnerability, beyond updating the firmware,

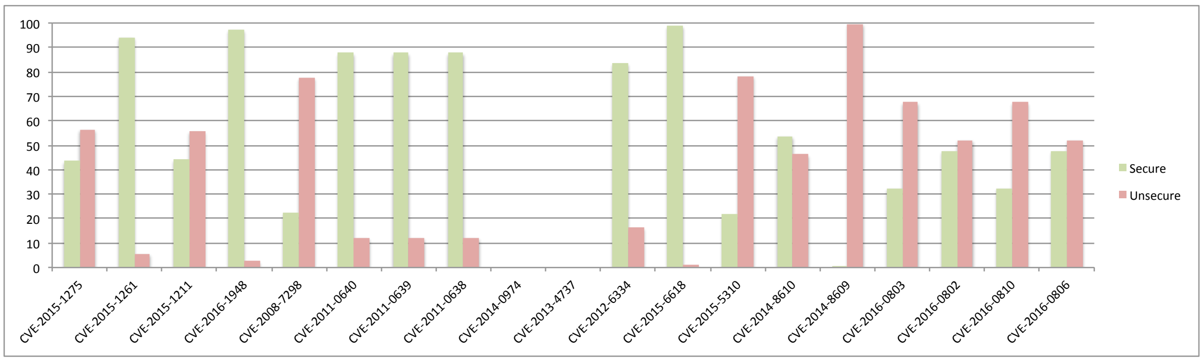

Figure 3.13: Analysis of devices on the field that are vulnerable to known vulnerabilities and exposures.

the user can disable the WiFi (N3). Fig. 3.13 shows the percentage of devices that are exposed to this vulnerability (289 out of 561): the ones that have the vulnerable firmware and WiFi enabled.

3.5 Countermeasures recommended

The misconfigurations presented in Section 3.3 indicate that end-users probably do not know or concern about security deeply enough to manage the settings of their own devices. Additionally, Section 3.4 presented, in practice, the importance of have a well set device.

The CIS security benchmarks for Android devices are defined for two different profiles: **Level 1**: which are intended to be practical and prudent, providing a clear security benefit and not negatively inhibit the utility of the technology beyond acceptable means, and **Level 2** which are intended for environments or use cases where security is paramount, acting as a defense in depth measure, but that may negatively affect the utility or performance of the technology. The first profile makes more sense for the vast majority of users, while the second seems adequate for professionals with sensitive information in their mobile devices. Obviously, if all those security parameters are set to the "secure" configuration, smartphones may lose their great utility, so the countermeasures presented in this section are not meant to achieve a device with no misconfiguration, but to mitigate the most common and concerning ones issues, thus avoiding threats. Also, in this work we argue that Android manufacturers should rethink their security configurations policies and provide best ways to help users make informed security decisions. Therefore we include some countermeasures that should be considered by manufacturers to improve the security in each of the assessed categories.

A concerning issue of the **application category** is that the number of users that have a security application installed is very low (36%). A common practice of the manufacturers is to customize the Android default built with some carriers and other own features. This way, we recommend the installation by default of an adequate security application, which will certainly increase the defenses against malware, diallerware and other malicious applications, thus preventing or mitigating attacks from this type of applications.

The most concerning issue regarding the **browser category** is that 14% of users have the “block pop-up” feature disabled. Even though pop-ups are not always harmful for the system, in some situations they include links to download disguised malware applications. The pop-ups serve as a gateway for the malware, adware and spyware to be installed into the device. Moreover, the CVE-2015-1261 shows a known vulnerability that exploits the fact that the device has the block pop up disabled allowing remote attackers to spoof the URL bar or deliver misleading pop up content via crafted text. This is the reason why users should use the block pop-up configuration. So we recommend periodic warnings when the block pop-up feature is disabled.

Regarding the **content category**, we observed that users often store a high amount of files, including photos, videos, audio and SMSs that may contain sensitive information. Although the Android default messaging application includes a mechanism that allows limiting the number of SMSs and MMSs stored, this feature is seldom used in practice. Therefore, we argue that the manufacturers should consider other mechanisms like, for example, encrypt by default these files or periodically notify the user about the existence of a large number of files that may contain privacy sensitive information proposing to auto delete then after a period of no-use. These countermeasures could increase the user privacy and are already provided by several third party applications [80, 88], thus they could be easily included by manufactures.

Two of the concerning issues regarding the **network category** are the large usage of WiFi against a cellular data service and the ICMP echo reply (ping) being enabled. Although the large usage of WiFi is understandable as it is faster and cheaper to use, the manufacturers could easily disable the ICMP echo by default, thus preventing a malicious agent from checking the availability of a target device and from fingerprint the target operating system [69]. Another possible practice is disabling the bluetooth after a period of inactivity, reducing the attack surface for vulnerabilities such as CVE-2015-6618.

Regarding the **password category**, a login mechanism forcing the use of at least a password to unlock the device should be the first barrier faced by a malicious agent trying to access the information stored on a device that is physically not secured (this is particularly important as, over the last year, almost 10 million mobile devices holding sensitive business data have been lost by employees only across Britain [22]). Therefore, manufacturers should force the users to set a password on their devices (e.g., using a configuration wizard during first time execution). Periodic warnings to change the password should also be considered, including the definition of a password expiration date.

It is important to note that many users keep their devices without a login mechanism, as they perceive the usage of a PIN (Personal Identification Number) as inconvenient, therefore manufacturers face challenges resulting from the trade-off between security and usability. The most common login methods are: PIN - mainly used in iOS which consists in a 4 digit pass code; Passwords - commonly used in personal computers; and Pattern password - mainly used in Android which involve to draw a pattern. The problems behind those methods are the need to remember the password, so the users tend to choose the easy-to-remember ones, such as birth date, dictionary words or a simple pattern. With respect to other authentication methods, fingerprint identification is the users’ preferred choice from security and convenience perspectives [10].

In the **permission category** we observed that users do not use the existing *encrypt* *storage* feature, which protects personal data that would otherwise be easily recovered through a local or application attack [27]. First time storage encryption is a lengthy process that may require an hour or more and that also introduces overhead when accessing the data afterwards. In addition, there is a possibility of data-loss in case of failure during the encryption process, also contributing to the low acceptability by the users. Therefore, the Android device storage should be encrypted by default and also the decryption keys should be set during first time execution.

Regarding the **system category**, we recommend some configurations as *enable GPS*, *enable ADB* and *enable non official store application* to be automatically disabled after a period of inactivity. This would mitigate the situations where the user forgets these configurations enabled, thus decreasing the attack surface of the device. For instance, disabling the ADB after a period of inactivity would mitigate vulnerabilities such as CVE-2011-0640, CVE-2011-0639 and CVE-2011-0638. Another important aspect is to have the system updated: as stated in the last section, the vulnerabilities CVE-2015-6618, CVE-2015-5310, CVE-2014-8609, CVE-2016-0803, CVE-2016-0802, CVE-2016-0810, CVE-2016-0806 can be mitigated by updating the device firmware. This way, we recommend users to take into account the way that the manufacturers tend to treat OS updates, even for old devices.

As a final note, it is important to note that in this section we did not impose countermeasures, but instead propose some that can be used. These countermeasures should be seen as recommendations for the manufacturers for decreasing the security impact of delegating to the users the responsibility of managing the security configurations.

3.6 Conclusion

This chapter presented a detailed field analysis on the user-defined security configurations in a sample space of 561 Android devices. The main contributions of this work are the identification of the main misconfigurations that users set on their Android devices, and the definition of concrete countermeasures that can be applied by the manufacturers to mitigate the risks raised by those misconfigurations. The field analysis is supported by the uSEA tool that aims at evaluating the overall security of Android devices based on the users-defined security configurations. In practice, the main advantages of the tool are: 1) automating the collection and analysis of the security configurations for Android devices; and 2) facilitating the discovery of security flaws based on user's misconfigurations.

Users that follow configuration best practices should have no security issues, or be less prone to them, when using theirs devices into safe networks and environments. As Android users frequently do not follow best practices (due to lack of zeal or even knowledge), manufacturers should rethink the way that the users can control the security policies of their devices.

The results presented show impressive numbers as the majority of the users of Android devices neglect the security configurations. Therefore, they are exposed to several security and privacy threats, as the frequent misconfigurations lead to a vulnerable environment. As most users do not know security deeply enough to manage their security controls, the manufacturers should **provide best ways to help users make informed security decisions**. Also, **Android manufacturers could provide in advance proper default configurations** (so that by default more users are more protected). Accordingly, **manufacturers should rethink their policies in terms of the security configurations that can be modified by the users**.

It is important to note that the number of misconfigurations in a device can give a general idea about the device overall security, but a device with a few very high severity misconfigurations can be more unsecure than a device with several low severity misconfigurations. This way, the next chapter studies the risk that each single configuration poses to the device, giving a better perception on its security level.

Chapter 4

Risk assessment of user-defined security configurations for Android devices

Assessing the risk of user-defined security configurations in mobile devices is a step towards identifying the most severe configurations in terms of security. This is of great importance for security analysis of mobile devices, as it enables the analyst to put a special focus on the aspects that pose a higher risk to the user and thus facilitates the process of improving the security level of such devices.

This chapter presents an approach for assessing the security risk posed by user-defined configurations in mobile devices, including a case study focused on Android devices due to its representativeness in the state of art. The approach is based on the analysis of the risk (impact and likelihood) of user misconfigurations to harm the device or the user. The impact and likelihood values are defined based on a Multiple-Criteria Decision Analysis (MCDA) performed on the inputs provided by a set of security experts regarding the set of user-defined security configurations from Chapter 3. An experimental evaluation considering the user-defined configurations of 561 Android devices is presented (using the data set from Chapter 3), showing that the majority of the users neglect important and basic security configurations and that the proposed approach can be used in practice to characterize the security risk level of such devices.

Figure 4.1 shows the process followed to characterize the risk, using the inputs provided by eight security experts from academia and industry. In practice, the experts were asked to provide their qualitative perception about the security impact and exploitation likelihood of the 41 security configurations, considering a 1 to 5 Likert scale (very low to very high). As the perception of the different experts diverge in some points (as expected), we applied a Multiple-Criteria Decision Analysis (MCDA) [34] algorithm based on Generalized Regression with Intensities of Preference (GRIP) [26] to weight the security recommendations. Finally, we used the K-means algorithm [38] to group the security recommendations into five classes of risk exposure, where Class 1 represents a very low risk exposure and Class 5 stands for a very high risk exposure. As we will see later, these classes allow more consistent values and provide a better distinction among configurations during the process of risk analysis of real Android devices.

Due to the difficulty to obtain significant values for the impact and likelihood of a given misconfiguration to harm the device or its owner, we rely on the opinion of multiple

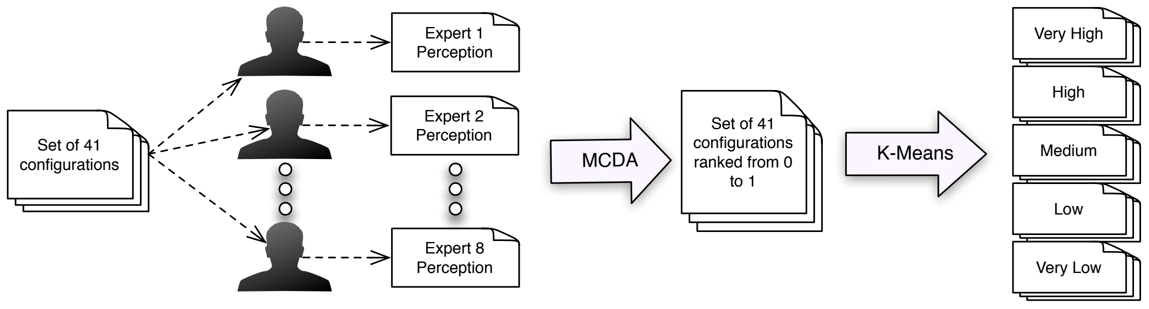

Figure 4.1: Overview of the process to define the security configurations severity level.

experts. Previous works have also considered the inputs from experts [39], [16], but mostly disregarding their divergence and only focusing on the arithmetic average of their perceptions, which may lead to poor results. This issue can be addressed by applying well-known techniques such as Multiple Criteria Decision Analysis (MCDA), which is a non-linear recursive analysis process that includes four steps [34]:

- structuring the decision problem: defining the domain where the MCDA will be applied and the objectives that should be achieved;
- modeling the preferences: specifying the attributes that will be considered, in our case, the user-defined security configurations;
- aggregating the alternative evaluations (preferences): defining comparisons between pairs of attributes; and
- making final recommendations: verifying the results of the process and, if necessary, repeat the previous steps.

In this work we use the GRIP method for ranking a finite set of actions (i.e. the user configurations) evaluated on multiple criteria (i.e. the impact and likelihood weights assigned by the experts). In practice, GRIP provides information about the intensities of preference for pairs of actions in a set considering the view of a given Decision Maker (DM). The preference information used in GRIP does not need to be complete: in fact, the DM is asked to provide comparisons for a subset of the pairs of reference actions for which his judgment is sufficiently certain considering a particular criteria [26]. An existing tool, Decision Deck [76], can be used to support this complex MCDA process using the GRIP method. It is capable of ranking the actions (i.e. the security configurations) considering the multiple criteria (the impact and likelihood assigned by the external experts) and the DM comparisons (provided by the authors of this work). This method was already successfully applied in other works in the dependability area [7], showing that it can be used to consolidate the diverse opinions of experts.

The outline of this chapter is as follows. Section 4.1 presents the analysis of the experts. Section 4.2 discusses the analysis of the configurations using GRIP and Section 4.3 presents five groups of security recommendations. Section 4.4 presents a case study considering 561 Android devices. Finally, Section 4.5 concludes the chapter.

4.1 Experts analysis

The risk (likelihood and impact) posed by not following a security recommendation depends on the context. In fact, the same misconfiguration in different scenarios may have diverse likelihoods of being exploited and diverse impacts. This way, to characterize the risk of the 41 security recommendations using MCDA, we considered a usage scenario adopted from [39] (this is a representative scenario, as several studies [12,21,59] show that there is a growing number of employees using their personal mobile devices for work):

The mobile device (smartphone or tablet) is used both for personal and professional purposes. The device is an integral part of a person's daily life - e.g. private phone-calls, take photos, share documents, social networking, e-mail, messaging, navigation, gaming, online banking, location based services, Internet browsing, e-health, etc. Furthermore the device is used in a business or government organization. It is used for business phone calls, corporate e-mail, expense management, customer relationship management, travel assistance, contact management, scheduling tasks and meeting, business social networking, video conferencing and reading documents.

The process to incorporate the impact and likelihood of each security recommendation in our approach includes the definition of weights drawn from the judgment of several experts.

4.1.1 Experts categorization

A total of eight experts, including four from academia and four from industry, were invited to answer a survey and give their qualitative perception about the impact and likelihood (both from very low to very high) of each security recommendation to be exploited (when not followed). The experts from industry are from different companies and have more than 5 years of experience working in projects related to security and mobile devices and the academics are from two distinct universities where they do research on cyber security, two of them hold a PhD degree and the other two are PhD candidates. We are aware that using a larger number of experts would increase the representativeness of the results regarding the general perceptions. Nevertheless, our process assures a diversity of experiences to be considered and the expectation is that, in average, the most important configurations are emphasized, even if there is no unanimity (this should represent as close as possible the reality taking into account a base scenario).

4.1.2 Gathering likelihood and impact information from experts

Figure 4.2 depicts the steps followed to support the analysis by the experts. We started by providing to the experts the list of 41 user-defined configurations. Each one was explained in detail, including a description of an example state where the configuration is not well set (in other words, it is incorrectly configured) and the security issues raised by that (the influence in the security). Finally, we formulated concrete questions for the experts to give their perception about the likelihood of that misconfiguration to be exploited and the corresponding impact if that happens.

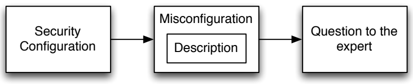

Figure 4.2: Process to support the analysis by the experts.

Table 4.1: Semantic of each value in the risk scale used.

Scale Intensity	Generic Description	Likelihood Description	Impact Description
Very Low	The configuration is not relevant	It is very unlikely that there is an exploit through this misconfiguration	The impact in case of an attack is minimum (e.g. disclosure of a music file)
Low	The configuration is relevant	It is unlikely that there is an exploit in case of misconfiguration	The impact in case of an attack is limited (e.g. disclosure of a non confidential document file)
Medium	The configuration is advisable to implement	It is somewhat likely that the misconfiguration may be exploited	The impact can bring some danger to the user (e.g. disclosure of the device location)
High	The configuration is important	It is likely that the misconfiguration will be exploited	It can lead to important information loss and disclosure and to limited financial losses (e.g. send text messages to premium rate numbers)
Very High	The configuration is critical	It is very likely that the misconfiguration will be be exploited	It can lead to significant financial, reputation and information loss and disclosure (e.g. disclosure of all information stored on the device)

Take as an example the configuration "*Enable encrypt device storage*", which is considered wrongly configured when the device storage is not encrypted. To help the experts understanding the dangers behind that configuration we provided the following description: *"Once the phone is encrypted, a numeric PIN or password is required each time the phone is powered on, protecting personal data that would otherwise be easily recovered through a variety of methods, for instance, removing the SD card from the target device and inserting it into another device to read the content."*. Considering this description, we then asked to the interviewees the following questions (considering the scenario introduced above): i) What is the likelihood of the device being attacked if it is not encrypted? ii) What would be the potential impact if an knowledgeable hacker had access to that non-encrypted device?

The possible responses to the questions range from very low to very high, or from 1 to 5, a scale widely used for risk analysis and that is adopted in other research works [39], [48]. The problem is that experts interviewing is a complex process and to capture their experience and knowledge the response scale should be well defined, easy to understand, and include a short and adequate number of options. For example, an excessively detailed scale, with many different values, would force the experts to make irrelevant considerations when deciding between close values (e.g. deciding between a 12 and a 13 out of 20 is very difficult and may be irrelevant). On the other hand, a vague scale (e.g. with 2 values) does not allow distinguishing and expressing the notions of likelihood and probability of different configurations. We believe that our five values scale with a specific semantic provides a very good balance (see Table 4.1 for the meaning of each value).

Table 4.2: Example of the experts perception for some security recommendations.

	Exp. 1		Exp. 2		Exp. 3		Exp. 4		Exp. 5		Exp. 6		Exp. 7		Exp. 8	
Configuration	L.	I.	L.	I.	L.	I.	L.	I.	L.	I.	L.	I.	L.	I.	L.	I.
C5	2	3	2	2	2	5	1	1	2	4	2	1	3	3	2	1
P02	4	4	5	5	4	5	5	5	4	5	3	5	4	4	4	4
PE5	5	5	5	5	5	5	4	3	2	1	3	5	5	5	4	5
S1	4	4	4	4	3	3	2	2	2	1	2	3	5	5	4	3

4.2 Analysis of the configurations using GRIP

Once the likelihood and impact perceptions were assigned by the experts, we proceeded to the analysis using the implementation of the GRIP method provided by the Decision Deck software. GRIP decision support is based on some key aspects [26]:

- The input data provided (i.e., the security recommendations and the weights assigned by the experts);
- The preference information provided by the Decision Maker (DM), which is composed of pairwise comparisons of actions (i.e. comparing two security recommendations);
- The computation phase of the decision process, where it checks the preference information provided by the DM (if it finds any problem, then support is provided for the DM to revise the preference information);
- The method derives the intensity of preference of each action (i.e. security configuration), which can then be used to determine the importance of the entire set of actions.

The reason behind the use of the GRIP method is the need to manage divergence in the opinions of the experts and, in that way, be able to calculate ranked intensities of preference regarding the risk of all the security configurations. Table 4.2 shows examples of the impact (I) and the likelihood (L) values defined by the eight experts for 4 configurations (C5, P02, PE5 and S1; see Table 4.3 for the extended names). As show, most of the experts selected high (4) and very high (5) values for impact and likelihood in both P02 and PE5 cases. However, there is some divergence: for instance, Expert 1 has the perception that the misconfiguration of PE5 has more impact and likelihood than the misconfiguration of P02; on the other hand, Expert 5 has an opposite perception. Similar divergences can be found for the other cases.

To rank the configurations based on the risk, the first action was to define the global settings in the Decision Deck software, namely the *Alternatives* and the *Criteria*. In our context, the *Alternatives* are the 41 security recommendations and the *Criteria* are the individual values provided by the experts for the Likelihood and Impact. Afterwards, as Decision Makers (DM) we provided the preference information for a number of alternative (security configuration) pairs: from the 820 possible combinatorial pairs, we provided preference information for 122 pairs (less than 15%). In practice, the DM can set three types of preferences: i) configuration a is preferred over configuration b; ii) configuration

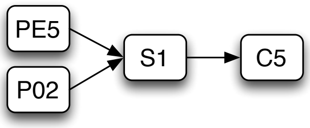

Figure 4.3: Ranking graph of the four exemplified configurations.

a is more or equally preferred to configuration *b*; and iii) both configurations are equally rated. The *preferred* term should be understood in the context of risk, where *a* is preferred over *b* if the incorrect configuration of *a* poses more risk than the wrong configuration of *b*. As mentioned before, the DM should provide information only for the pairs for which his judgment is sufficiently certain and the GRIP will infer the remaining ones.

Considering the examples in Table 4.2, the DM can state that P02 is preferred over C5 and S1. In fact, P02 is preferred over C5 on 15 criteria and over S1 on 11 criteria (out of the 16 criteria defined by the experts). However, when comparing P02 and PE5, both configurations are equally rated on 6 criteria, P02 is preferred over PE5 on 4 criteria and, finally, PE5 is preferred over P02 on 6 criteria. Based on these close and divergent values, the DM may not be sufficiently certain to state a preference between the two configurations and should not do so. The advantage of using the GRIP method is that it calculates the preference intensity based on the preference pairs for which the DM is capable of defining a clear preference. As a further example, Figure 4.3 shows a ranking graph for the four configurations in Table 4.2. As show, P02 and PE5 are equally ranked by the GRIP method, while S1 and C5 are less preferred respectively. Note that, similarly to traditional risk analysis, we consider that impact and likelihood have the same contribution to the risk exposure.

Based on the pairs of preferences provided by the DM, the GRIP algorithm computed a ranking of all configurations based on the intensities of preference (a graph similar to the one in Figure 4.3, but for the 41 configurations). In practice, such ranking includes all the alternatives, from the most risky to the less risky one (based on the perceptions of the experts). In addition to the graph, the ranking is defined by the GRIP using a 0 to 1 value (the configuration with value closer to 1 is the one posing the highest risk if incorrectly configured, while the configuration posing the lowest risk has the value closer to 0 - the remaining ones have a value in the between depending on their position in the ranking).

Table 4.3 shows the result of the MCDA analysis. The values in the column *Intensity of Preference* represent the approximate quantitative risk exposure of each security configuration using a scale from 0 to 1. The *Relative weight* column shows the weight of each configuration on the overall device security considering the perception of the experts. In practice, the relative weight (which is a percentage calculated using the rule of three) is defined as the individual *Intensity of Preference* of the configuration divided by the sum of all individual *Intensity of Preference* multiplied by 100. Finally, the *Cumulative weight* provides a better view of the impact that a group of user-defined configurations has on the overall device security. According to our analysis of the experts perceptions, the 10 most risky security configurations are responsible for approximately 39.73% of the overall risk (gray area in the top of the table), while the 10 less risky configurations are responsible

Table 4.3: Security concerns and their respective weights.

ID	Security Configuration Recommendation	Intensity of Preference	Relative Weight	Cumulative Weight
P01	Enable password	1.00	4.59	4.59
A2	Do not install root applications	0.974	4.47	9.06
P05	Do not make passwords visible	0.914	4.20	13.26
PE5	Do not root (“Jailbreak”)	0.888	4.08	17.33
P02	Require alphanumeric password	0.888	4.08	21.41
S5	Disable unknown sources	0.828	3.80	25.21
P11	Set timeout minutes for sleep	0.802	3.68	28.89
P06	Erase data upon excessive password failures	0.802	3.68	32.57
PE1	Enable encrypt device storage	0.791	3.63	36.20
PE4	Disable write permission on “/data” folder	0.769	3.53	39.73
P09	Insert minimum passcode length	0.765	3.51	43.24
S4	Disable developer options	0.743	3.41	46.65
P04	Enable SIM card lock	0.716	3.29	49.94
P12	Enable pattern password	0.716	3.29	53.22
PE3	Disable write permission on “/system” folder	0.679	3.12	56.34
P10	Set a minimum number of complex characters	0.679	3.12	59.46
A4	Do not install malware	0.631	2.90	62.35
P07	Set a maximum password age	0.631	2.90	65.25
B1	Disable Java Script	0.545	2.50	67.75
P08	Set number of password history	0.545	2.50	70.25
S1	Update firmware to latest version	0.545	2.50	72.75
N04	Disable bluetooth when not needed	0.53	2.43	75.19
P13	Disable visible lock pattern	0.515	2.36	77.55
A3	Install anti-malware	0.466	2.14	79.69
N07	Turn off personal hotspot when not needed	0.459	2.11	81.80
N11	Disable bluetooth discoverability	0.459	2.11	83.90
B4	Enable block pop-ups	0.459	2.11	86.01
N03	Disable Wi-Fi when not needed	0.41	1.88	87.89
S6	Disable location services when not needed	0.41	1.88	89.77
S9	Disable mock location	0.373	1.71	91.49
N01	Remove saved Wi-Fi entries	0.343	1.57	93.06
N10	Disable ICMP ping	0.272	1.25	94.31
C8	Limit the number of documents on the device	0.257	1.18	95.49
B3	Disable cookies	0.254	1.17	96.65
N02	Disable network notification	0.213	0.98	97.63
N06	Enable airplane mode	0.172	0.79	98.42
C4	Limit the number of MMS on the device	0.172	0.79	99.21
C7	Limit the number of videos on the device	0.086	0.39	99.61
C3	Limit the number of SMS on the device	0.086	0.39	100.00
C5	Limit the number of photos on the device	0.00	0.00	100.00
C6	Limit the number of audios on the device	0.00	0.00	100.00

for only 6.94% (gray area in the bottom of the table). The values presented in Table 4.3 portray the severity of each configuration, thus allowing to identify the major security problems regarding user-defined configurations. Moreover, they can be used to support a security assessment campaign on Android devices, warning users about the risk exposure of their devices.

The information in Table 4.3 shows the importance of each security recommendation. However, giving the fact that values are very close in many cases, it is very difficult to determine accurately the importance of each security configuration in a quantitative way, as in many cases making a distinction between two very close values is meaningless [70]. For example, P01 and A02 have a intensity of preference of 1.00 and 0.974 respectively, as for both configurations, many experts assigned a very high value for likelihood and impact. In fact, P01 has 13 criteria assigned as *very high* and 3 criteria assigned as *high*, while A2 has 12 criteria assigned as *very high*, 2 criteria assigned as *high* and 2 as *low*. Considering that the experts diverge in some criteria and alternatives (configurations), and that their perceptions may have outliers, we can generically state that, for both cases, the majority of criteria are very high. This way, making a qualitative analysis within the semantics presented in Table 4.1 by grouping the security configurations with similar preferences, will lead to more consistent values and provide a better distinction among configurations.

Table 4.4: Severity classification of the security recommendations.

Class	Risk (R)	Security Configuration
VH (5)	$1 \geq R \geq 0.8$	P01, A2, P05, PE5, P02, S5, P11, P06
H (4)	$0.8 > R \geq 0.594$	PE1, PE4, P09, S4, P04, P12, PE3, P10, A4, P07
M (3)	$0.594 > R \geq 0.358$	B1, P08, S1, N04, P13, A3, N7, N11, B4, N3, S6, S9
L (2)	$0.358 > R \geq 0.142$	N1, N10, C8, B3, N2, N6, C4
VL (1)	$0.142 > R \geq 0$	C7, C3, C5, C6

4.3 The groups of security recommendations

In this section we detail the process used to group the security recommendations taking into account the risk posed if not followed. The K-means grouping algorithm [38] implemented by the Weka software package [29] was used to define five classes of severity. In practice, the algorithm partitions n observations into k groups where each observation belongs to the group with the nearest mean. The section also describes the configurations in each group and discusses how the groups and configurations should be used to assess risk and identify security problems.

4.3.1 Grouping the security configurations

The 41 recommendations were divided in 5 classes (each one corresponding to a risk level, from very low to very high, or 1 to 5, using the well-known Likert scale with five points [15]) based on the *Intensity of Preference* shown in Table 4.3. The groups are presented in Table 4.4: the first column is the risk level/group, the second is the interval of *Intensity Preference* in the group, and the configurations are presented in the third column (each row is ordered by the computed intensity; this defines a relative order in each group). For example, the configurations in the class *M (3)* have an *Intensity of Preference* between 0.358 and 0.594 and pose a moderate risk exposure.

The classification presented in Table 4.4 puts each recommendation into one of five distinct groups: *Very Low (1)*: recommendations that are unanimously not relevant, *Low (2)*: the ones that are not relevant but should have some attention, *Medium (3)*: the ones that are advisable to be set correctly, *High (4)*: the ones that are not critical but are very important, and *Very High (5)*: the ones that are unquestionably very critical. Note that, in addition to supporting the risk assessment, this classification can also be used as a guide for identifying which best practices should be implemented in a device according to their critically. For example, consider the IT technical support office of an organization that controls the mobile devices of the employees: its quite clear that such control should first focus on the critical and important security recommendations, maybe disregarding the ones with very low relevance in order to reduce the maintenance cost.

A key observation when correlating the groups in Table 4.4 with the *Intensity of Preference* in Table 4.3, is that the recommendations in the two most important groups account for 65.25% of the risk exposure. This shows that there is a subset of the configurations that are generically considered as the most relevant by security experts and that cover more than half of the user-dependent security aspects in mobile devices.

4.3.2 Describing the groups

The security recommendations in the higher severity class (**Class VH (5)**) are related to the application, password, permission and system categories. The one related to the application category (A2) recommends not to install root applications as such application can go around some security mechanisms (i.e. file permissions). The ones in the password category (P1, P2, P5, P6 and P11) are all related to the strength of login mechanisms. Specifically, (P1) *enable password* states that it is important to enable at least one login method; (P2) *enable alphanumeric password* states that is important to enable a strong login method; (P5) *do not make passwords visible* is important to prevent an attacker from observing user input the password; (P6) *erase data upon excessive password failures* is important to avoid brute force attacks; and (P11) *set timeout before sleep* prevents access if the user lets his device unlocked somewhere. The configuration related to the permission category (PE5) is similar to A2 as a user with root permission is able to circumvent security mechanisms (i.e. delete system files). Finally, the recommendation (S5) *disable unknown sources* is key to avoid installing applications from untrusted sources.

The high severity class (**Class H (4)**) includes recommendations in the application (A4), password (P4, P7, P9, P10 and P12), permission (PE1, PE3 and PE4) and system (S4) categories. A4 is related to the installation of malware, whose can be used to steal confidential user information leading to financial and other losses. Regarding the configurations related to the password category: (P4) *enable SIM card lock* requires the SIM to be unlocked by using a PIN number before connecting to the carrier network and accessing the information stored on it, even when the SIM card is placed in a different phone; (P7) *set a maximum password age* is used to prevent the user from remaining with the same passcode for a long period of time; (P9) *insert a minimum password length* is important to force the user to set a strong passcode; (P10) *set a minimum number of complex characters* also avoids the user from setting a weak passcode; and (P12) *enable pattern password* requires the user to draw a pattern before unlocking the device. For the configurations related to permissions: (PE1) *enable encrypt device storage* is very important as it creates an additional barrier for a malicious agent to retrieve the storage information in case of a device theft or loss; (PE3) *disable write permission on "/system" folder* protects the critical part of the device software by making it read-only; and (PE4) *disable write permission on "/data" folder* does the same for the system data. Finally, (S4) *disable developer options* is important because, among other things, it prevents the access to the command terminal via an USB cable.

The security recommendations with a moderate severity (**Class M (3)**) are related to the application (A3), browser (B1 and B4), password (P8 and P13), network (N3, N4, N7 and N11) and system (S1, S6 and S9) categories. The recommendation (A3) concerns the installation of anti-malware which increases the device protection against malicious software and attackers. The configurations related to the browser category include (B1) *disable java script* that contributes to avoid some browser exploits, and (B4) *disable block pop-ups* that can be used to open untrusted malicious web sites and also for performing phishing attacks. As for the ones related to the password category, (P8) *set number of password history* prevents the user from repeating passwords, and (P13) *disable visible*

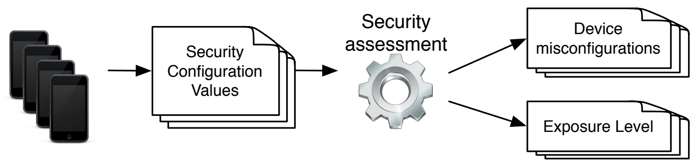

Figure 4.4: High level view of the risk analysis process.

lock pattern is related to visually tracking the user drawn motion while unlocking the device. The ones related to the network category are: (N3) *disable Wi-Fi when not needed*, (N4) *disable bluetooth when not needed*, (N7) *turn off personal hotspot when not needed*, and (N11) *disable bluetooth discoverability*, which are all related to communications and reduce the potential attack surface. Finally, in the system category, (S1) *update firmware to latest version* allows fixing some security flaws in outdated software, and (S6) *disable location services when not needed* makes it harder for a malicious agent to retrieve the users location.

The security recommendations with a low severity (**Class L (2)**) are related with the browser, content and network categories. The one in the browser category (B3) recommends disabling cookies in the browser, which protects against attackers tracking, altering or stealing confidential information that may be stored in the cookie. The ones in the content category (C4 and C8) are about the number of MMS (Multimedia Messaging Service) and documents stored on the device: the idea is that a device with less content of these types has a lower probability to store confidential or sensitive information on it, but this may not be a security issue. As for the network category, (N1) *remove Wi-Fi entries*, (N6) *enable airplane mode*, (N10) *disable ICMP ping* and (N2) to *disable network notification*, may improve communication security but they are either unfeasible (as is the case of N6 that makes the device unusable in most cases) or they are of low relevance (e.g. N2 prevents users from connecting to open or wrong networks with similar SSID, which is not a frequent issue).

Finally, the class with the lowest severity (**Class VL (1)**) includes recommendations in the content (C3, C5, C6 and C7) category, concerning videos, SMS (Short Message Service), photos and audios management. To the experts, the number of items of those types does not have a strong relation with the device security risk. The idea behind those recommendations is that, when a device stores less content there is a lower probability to store sensitive information that can be disclosed in the case of theft or cyber attack, but this is only true if that information is not protected using some other security mechanism (e.g. authentication, privileges, and encryption, which are addressed by other configurations with higher risk).

4.3.3 Using the groups to assess risk and identify problems

Figure 4.4 provides a high level view of the risk analysis process and of the outputs. First of all, the security configuration values of the Android device under analysis have to be extracted, which can be done in two ways: i) *manually*, by navigating through the device menu checking its configurations; or ii) *automatically*, by using an Android application able to extract the security configuration values (for example, the uSEA tool presented in Chapter 3). Once the values of each security configuration are extracted from the device, it is possible to start the security risk assessment. The output of such assessment can be: 1) the identification of the configurations incorrectly set (i.e. the values that are different from the recommended ones), which can be directly used to better configure the device; and 2) the calculation of the overall risk exposure level of the device by summing the exposure level of the configurations that are not correctly set. Such exposure intends to portray the security risk faced by the user and can also be used to compare different installation alternatives and/or devices from different manufacturers.

In order to allow calculating the exposure measure, we assigned values for each severity class defined in Table 4.4. Following a simple approach, we postulated the following: 5 for **Class VH (5)**, 4 for **Class H (4)**, and so on. With this we intend to maintain the *Intensity of Preference* generated by the GRIP method, assuming that such values (1 to 5) represent the risk exposure of each identified class (this is also the approach most followed in traditional risk analysis), thus allowing a good estimation of the overall risk exposure level for each device under analysis. For example, a device with all the configurations in the Class VH incorrectly set and all the others (in the remaining classes) correctly set would have a risk exposure of 40 (29.85%), while a device with all the configurations in the Class VL incorrectly set and all the others well set would have a risk exposure of 4 (2.98%).

A key aspect is the potential interdependence between user-defined security configurations, which should be taken into account when analyzing the risk exposure of a device. Recognizing that fully defining the interdependence cases among the 41 recommendations is a difficult goal, in this work we considered the following ones: 1) if N3 ("*Disable WiFi when not needed*") is well set, then N1 ("*Remove WiFi entries*") and N2 ("*Disable network notification*") do not cause any harm even if they are not well set as well, because N1 and N2 only make sense when the WiFi is enabled; 2) if N4 ("*Disable bluetooth when not needed*") is well set, then N11 ("*Disable bluetooth discoverability*") does not cause any harm because *discoverability* is automatically disabled when the bluetooth connection is disabled; 3) if P1 ("*Enable password*") or P2 ("*Require alphanumeric value*") are well set, then the configuration of P12 ("*Enable pattern password*") is not relevant, as there are stronger passwords enabled on the device; and 4) if P12 is wrongly set (i.e. a pattern password is not used), then P13 ("*Disable visible lock pattern*") should not be considered, as the visible lock pattern is only relevant when the user enables the pattern password. In practice, the interdependence among configurations is considered during the risk analysis of each device. For instance, if a user correctly sets N3 ("*Disable WiFi when not needed*") we automatically consider that N1 and N2 are correctly set, as the risks posed by these two configurations are not relevant and should not be included in the computation.

4.4 Case Study: Risk Analysis of real Android Devices

This section presents a case study that applies our approach using the dataset extracted from 561 Android devices (see Chapter 3). These data was extracted by the uSEA (user-defined SEcurity configuration Assessment tool), which means that any Android user can install the application and contribute to our dataset. As mentioned before, the dataset was gathered over a period of 15 month, collecting the devices' security configurations whenever the user executed the tool. However, for the analysis presented in this work, we consider only the results of the first execution in each device, thus portraying its security state before the user received any feedback from our tool (which avoids influencing the results). In the following sections we analyze a number of typical configurations, including factory configurations, thus providing a broad view on the possible best, worst, and default installations, and then go into the discussion of the dataset.

4.4.1 Analysis of typical installations

The analysis of real user-defined security configurations of Android devices allows discovering the more common and concerning issues. However, we know that following all security recommendations may not be possible as added security impacts other quality attributes like usability, performance, and utility in general. This way, to better understand the range of potential and typical installations of Android mobile devices, Table 4.5 presents the analysis of five possible cases considering the set of 41 recommendations that are part of our approach. The *Worst Case* represents the worst set of configurations that a user can do on his device, assuming that such user has programming and command line skills to jailbreak the device, change folders permission and other complex tasks. The *User Worst Case* represents the worst configuration that an inexperienced user can do on his device, considering that such user is not capable (or does not want) to jailbreak the device and perform tasks that need programming or command-line skills. *Factory* configuration refers to the factory settings of an Android device (actually, this set of configurations was extracted from a Samsung Galaxy S4 with Android 5.0.1 (Lollipop) version after performing a factory reset). The *User Best Case* are the securest configurations that an inexperienced user can perform in his device, assuming that he is not capable or does not want to install specific applications to set configurations that are only changeable by a third party application or programatically. Finally, the *Best Case* represents the securest configurations that a user can have in his device including the installation of applications that control specific configurations, such as P6, P7, P8 and P9 that can only be set by using a third party application or programmatically. The marks ✓ show the cases where a given configuration can be (or is) correctly set and the empty cells refer to the reverse (i.e. configurations that cannot be, or are not, correctly set). The next paragraphs discuss the main aspects shown in Table 4.5 for each case.

Table 4.5: Security configurations for five possible cases.

	A2	A3	A4	B1	B3	B4	C3	C4	C5	C6	C7	C8	N1	N2	N3	N4	N6	N7	N10	N11	P1	P2	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	PE1	PE3	PE4	PE5	S1	S4	S5	S6	S9
Worst Case																																✓									
User Worst Case	✓																												✓					✓	✓	✓		✓			
Factory	✓		✓			✓	✓	✓	✓	✓	✓	✓				✓		✓		✓									✓	✓		✓		✓	✓	✓	✓	✓	✓		✓
User Best Case	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Best Case	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

The set of configurations named *Worst Case* includes, not surprisingly, only one correct configuration (P13). As we discussed before, there is an interdependence between P12 and P13 (e.g. if the user does not enable pattern password (P12), then it automatically disables the visible lock pattern (P13)). This is relevant to show that a user can configure a device considering only one of the security configurations presented in Table 4.3. This set represents, in terms of user-defined security configurations, the worst possible case.

In the *User Worst Case* we can see that an inexperienced user can have in his device at most 35 incorrect recommendations. The exceptions are A2 ("*Do not install root applications*"), P10 ("*Set a minimum number of complex characters*"), PE3 ("*Disable write permission on "/system" folder*"), PE4 ("*Disable write permission on "/data" folder*"), PE5 ("*Do not root, "jailbreak", the device*") and S4 ("*Disable developer options*"). These configurations are correct by default and they require specific skills to be set in a different way (in fact, setting them requires executing procedures that only an expert user would look for). In other words, the well set security configurations are factory settings that cannot be directly changed by the user to an insecure state.

The *Factory* installation includes 19 wrong recommendations. This is an interesting case if one wants to compare a specific installation with the default one (e.g. assessing if the user has changed the configurations to a more or less secure state). Furthermore, it shows that, by default, mobile devices tend not to be very secure and that improved default settings could improve the security of more users (considering that many users decide to keep the default settings). It is important to highlight that the SIM card lock (P4) is enabled by default by some carriers (such lock is a *feature* of the SIM card that can be changed via the device but that remains unchanged when the card is moved from one device to another). However, there are many carriers that do not force SIM card lock by default, including the one used in the device that was the subject of this analysis (dual SIM was not considered in this research because its support was added on Android 5.1 API [17], which represents only 24% of Android devices [18]).

The *User Best Case* shows that an inexperienced user trying to follow the best practices will have to leave 5 configurations incorrectly set, four of the cases (P6, P7, P8 and P9) due to the fact that it is not possible to set them through the standard Android user interface (as aforementioned, it can only be set through a third party application that implements the Android Device Admin API [19]). N6 ("*Enable airplane mode*") is the other incorrect configuration, as during the extraction of the device configurations, our tool needs somehow to be connected to the Internet (to send the data to our server). Anyway, connectivity is a basic requirement of any mobile device, so this is an unpractical configuration.

The *Best Case* includes only 1 incorrect recommendation, which is indeed also incorrect in all the other cases (N6). This configuration is related to the airplane mode that cannot be activated most of the times, as discussed above. This set represents, in terms of user-defined security configurations, the best possible case.

Knowing the number of incorrect configurations and which recommendations are not followed gives an idea about the security issues of Android devices. However, such analysis takes all configurations equally and does not consider the risk exposure of each misconfiguration as proposed by our approach. This way, Table 4.6 shows the risk assessment for the five installations. For each installation we show the risk value considering the configurations in each of the five classes defined in Table 4.4 and also the risk value for all configurations (two right most columns, the first as an absolute value and the second

Table 4.6: Security assessment for five possible cases.

	Class 5	Class 4	Class 3	Class 2	Class 1	Total	Relative Risk
Worst Case	40	40	33	14	4	131	97.76%
User Worst Case	30	24	33	14	4	105	78.36%
Factory	20	20	15	10	0	65	48.5%
User Best Case	5	8	3	2	0	18	13.43%
Best Case	0	0	0	2	0	2	1.49%

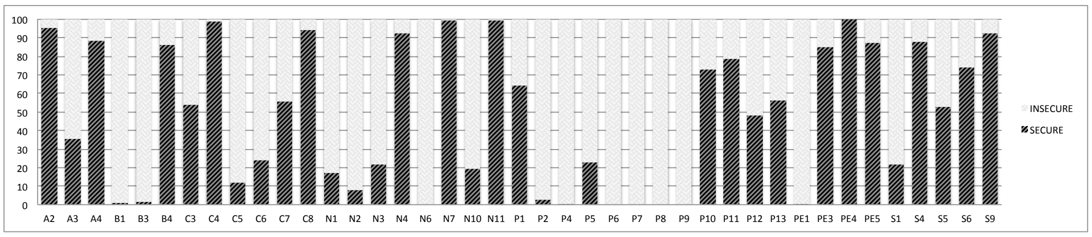

Figure 4.5: Evaluation of the 41 security recommendations evaluated in a universe of 561 devices.

as a percentage). For instance, as each misconfiguration in Class 5 (very high severity) corresponds to 5 points in terms of risk exposure, the *Best Case* installation as a value of 0 (all Class 5 configurations are correctly set) and the *Factory* as a value of 20 (four incorrectly set configurations (P01, P02, P05 and P06)).

Although Class 5 is the most severe one, classes 3 and 4 include more security recommendations, so if a user wants to have a secure installation then he should pay special attention to the three most severe classes. It is also important to note that even the configurations in the *Best Case* installation pose some risk, although a small one (1.49%), as there are some configurations that cannot be set. Finally, the default installation (*Factory*) has a risk value of about 48.5%, which shows the inadequacy of default settings.

4.4.2 Analysis of real configurations of Android devices

The analysis of real Android devices is a key contribution to discover the main and most common issues that users face while configuring their devices. Figure 4.5 presents the 41 security recommendations after an analysis of the 561 devices in our dataset. Each bar of the chart shows the percentage of devices that follow the security recommendation (*secure* legend). For instance, the first bar shows that more than 90% of the Android devices in the dataset correctly implement the A2, which recommends not to install root applications. Therefore, a higher bar represents better implementation of the recommendations by the users.

Regarding the recommendations with a very high risk severity, P2, P5 and P6 are the ones most commonly neglected by the users. Except for P2 and P5, which can be set through the Android user interface, the other (P6) can only be set by using a third party application or programmatically, which explains the low percentage of cases where this configuration is correctly set. On the other hand, A2, P11 and PE5 are the recommendations in the very high severity class that are most commonly correctly set by the users. This show that only few users, less than 13%, have root applications installed or root permission enabled on their devices and also that, in 79% of the cases, they are

Table 4.7: Presentation of the security configuration set of three installations.

	A2	A3	A4	B1	B3	B4	C3	C4	C5	C6	C7	C8	N1	N2	N3	N4	N6	N7	N10	N11	P1	P2	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	PE1	PE3	PE4	PE5	S1	S4	S5	S6	S9
Worst Instal.						✓		✓				✓				✓		✓		✓									✓	✓		✓			✓					✓	✓
Common Instal.	✓		✓			✓	✓	✓			✓	✓				✓		✓		✓	✓								✓	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓
Best Instal.	✓		✓			✓		✓		✓		✓	✓	✓	✓	✓		✓	✓	✓	✓			✓					✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓

concerned about the time that the device takes to lock after a period of inactivity.

For the recommendations that pose a high risk exposure, A4, PE3, PE4 and S4 are the ones most followed by the users. PE3 and PE4 are well defined by default (factory settings) and they require specifics skills to be set, such as the adb (Android Debug Bridge), which is mainly used by programmers. S4 ("*disable developer options*") requires a procedure that only an expert user would look for. A4 is related to the installation of malware, which occurs mainly when a user installs an application from outside the official Android application store (Google Play). On the other hand, the ones that are more neglected are P7, P9, P4 and PE1. The first two, P7 and P9, cannot be accessed through the Android user interface, so most of the users cannot change them easily (and they are incorrectly set by default), while PE1, which is about enabling storage encryption, is also not commonly set as it is time consuming, prone to data loss and the device looses some performance. Enabling SIM lock (P4) is also not followed by the vast majority of the users (the default value depends on the carrier), although it makes more difficult for a third party to use the SIM card, even in a different device.

In the recommendations that pose a moderate risk, A3, B1, N3, P8 and S1 are the most commonly neglected by the users. Some of them (A3, B1 and P8) are misconfigured by default (factory settings) and the *update firmware* (S1) recommendation depends on the manufacturer to provide an update. The *WiFi usage* (N3) is the most common way to connect to the Internet as it is normally cheaper and faster than the data network.

Regarding the recommendations in the low severity class, we can see that B3, N1, N2, N6 and N10 are neglected by a great part of the users (more than 80%). Some of these recommendations, for instance, enabling cookies (B3), using airplane mode (N6), disabling network notification (N2) and removing WiFi entries (N1), have a direct impact on the user experience regarding Internet access. The ping response (N10) is a recommendations that is integrated with the build, so it is defined by the manufacturers: some of them disable it, while others enabled it by default.

Finally, all the recommendations with a very low risk exposure (C3, C5, C6 and C7) are neglected by a great part of users. This is understandable as one of the main features of mobile devices is to store media, such as messages, videos, photos and audios. Although these recommendations are not followed by the users, they may not really compromise the security of the device.

Table 4.7 presents the security configurations for three concrete installations, marking with ✓ the configurations that are correctly set. The fist installation (*Worst Instal.*) is the worst case observed in our dataset. The second (*Common Instal.*) shows what we could call an average installation (actually there is no device with this specific set of configurations: the installation includes the correct/incorrect configurations that are present in more than half of the devices). Finally, the third installation (*Best Instal.*) refers to the best configured device in the dataset.

The risk exposure of the *Worst*, *Common* and *Best* installations is respectively 71.64%, 43.28% and 31.34%. Comparing the *Common Installation* with the *Factory* one in Table 4.5, we can observe that they are quite similar, which brings forth the idea that manufacturers should provide better factory settings (so by default more users would have a safer configuration). On the other hand, note that the *Best Install.* and the *Worst Install.* are very different from the possible *User Best Case* and *User Worst Case* in presented Table 4.5. For instance, the best configuration in the dataset has 31.34% of risk exposure while for the user best possible installation it is 13.43%. In practice, this shows that users are not aware or simply do not care about the risks of misconfiguration. Therefore, if an organization wants to implement a "bring your own device" (BYOD) policy, then it needs to provide or use *Mobile Device Management* (MDM) tools to control the security configurations of the devices and also provide security training to the users.

4.5 Conclusion

This chapter presented an approach for characterizing the security risks posed by incorrect user-defined configurations. The set of 41 recommendations from Chapter 3 were analyzed and characterized by 8 experts in terms of the likelihood of being exploited and the impact in case of a successful attack. MCDA was used to harmonize the opinions of the experts and a clustering algorithm was applied to divide the configurations in 5 qualitative groups of risk severity. The results presented show that the majority of the users of Android devices neglect security configurations. Therefore, they are exposed to several security and privacy threats, as misconfigurations lead to a vulnerable environment. The characterization of security recommendations for Android devices is an important step to rationalize risk assessment. We are aware that our approach may become obsolete as new threats, vulnerabilities, attacks and security configurations appear. However, the process presented in this work can be easily executed for new security recommendations, thus keeping it updated. Another important aspect is that, although the case study is focused on Android, the approach is generic enough to be applied in other mobile platforms. Obviously, adaptations will be required, namely regarding a set of configurations specific for the target platform.

The proposed approach is specially useful for organizations that intend to enable the usage of personal mobile devices to access the corporate systems, data and information. In fact, the approach provides the support required for them to perform a sound risk assessment and thus ensure a minimum security level on the mobile devices by discovering the main misconfigurations issues.

Analysing the security risks of each user-defined configuration is a crucial step to define a security configuration benchmark. The risk level of each configuration can be used to calculate metrics able to compare the security level of different instances of mobile devices, which is the main goal of a benchmark. This way, our risk-based approach approach is a key element of the benchmark proposed in the next chapter.

Chapter 5

Benchmarking security configurations

Few research has focused on assessing or preventing the threats raised by misconfiguration of user-defined settings, and most efforts have focused instead on malware detection. However, our previous analysis shows the critical importance of researching new techniques, tools, and methods to evaluate the way users interact with their devices. The definition of a benchmark for security configurations is indeed of utmost importance when one wants to evaluate the security level of such devices.

In order to allow characterizing the relative security level of mobile devices with respect to user configurations, this chapter proposes a benchmarking approach, supported by the uSEA tool, that is able to automatically gather, analyze and compare alternative configurations, thus providing key information for supporting decision making in what regards improving the security of mobile devices. The chapter presents a concrete benchmark and an experimental evaluation for the specific case of Android devices, demonstrating the use of the benchmark and validating its key properties. The results, based on the analysis of our data set of 561 devices, show that benchmarking is indeed a good way to identify the most secure user-defined configuration, allowing both the comparison of the configuration of different devices and the comparison of multiple configurations of a same device.

The benchmark process includes two main steps: security qualification, dividing the evaluated systems into acceptable and unacceptable, given the specific security requirements and security knowledge in the domain; and risk quantification, to calculate the security risk level of the device misconfigurations. In practice, the output of the benchmark characterizes the overall risk level of the assessed devices considering the risk level of each individual configuration, as presented in chapter 4.

The outline of this chapter is as follows. Section 5.1 presents a generic risk-based approach for benchmarking security configurations. Section 5.2 discusses the problem of specifying benchmarks for security configurations. Section 5.3 presents a case study focusing in a benchmark for Android devices. Section 5.4 presents an experimental evaluation to demonstrate the proposed benchmark and Section 5.5 validates it. Finally, section 5.6 concludes the chapter.

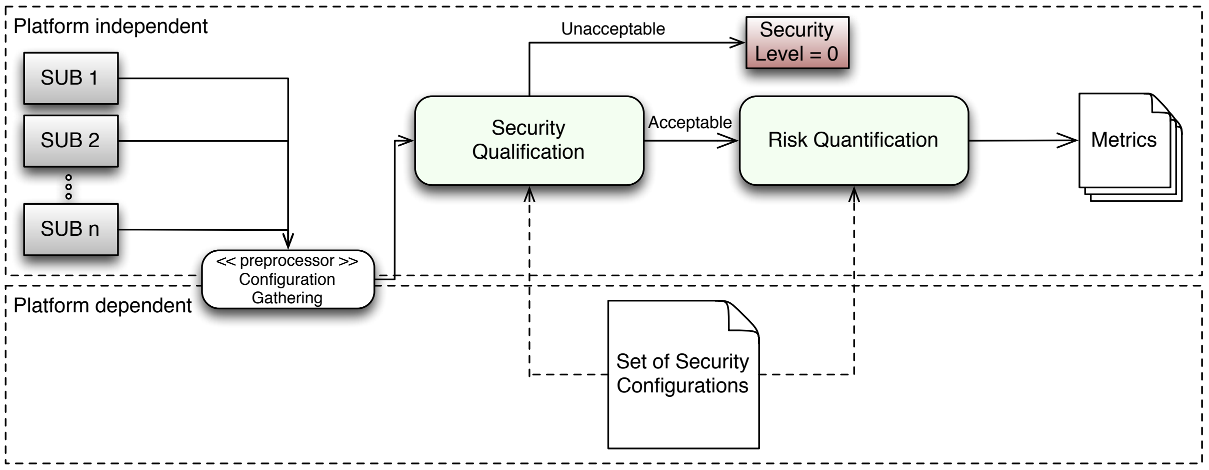

Figure 5.1: High level vision of the benchmarking process

5.1 A risk-based approach for benchmarking security configurations

A security benchmark should evaluate the explicit security mechanisms and visible defects that the system has, and it should consider the possibility of the system still having unknown security problems even after all possible checks [66]. This way, the proposed security configuration benchmarking approach, depicted in Figure 5.1, includes two distinct evaluation phases, namely: *security qualification* and *risk quantification*. This division into two steps was already successfully used in [16], but in the context of trustworthiness benchmark for OLTP (Online Transaction Processing) systems.

The *configuration gathering* activity is a preprocessor and is related with the process of collecting the user-defined configurations from the systems under benchmarking. Using our approach, there are two options during *configuration gathering*: 1) send he device state (set of security configurations values) directly to the next step (*security qualification*); or 2) store the device state in a database for further analysis. While in the first option the benchmark user executes the whole process for a single device (e.g. to compared its specific configuration against a list of good configuration practices), in the second, s/he can save the state of multiple SUBs, thus allowing running the benchmarking process for multiple devices at once (useful to compare different devices).

5.1.1 Security qualification

The *security qualification* step is responsible for assessing whether a mobile device can be qualified or not for the next step of the benchmarking process. In practice, the goal of this stage is to identify major user misconfigurations that lead to a certainly insecure state. The benchmark user should select the configurations that are mandatory for a device be considered as having the minimum security level required. For instance, the minimum absolute security level that we would expect for a mobile device is that it must require a password to be used and to have access to the information stored on it. A mobile device without a password would not be acceptable for most people, as the mobile device could be easily used to access the owner's e-mail, social media and make calls.

The *security qualification* step defines if the SUB is *acceptable* or *unacceptable* for the next step. If any of the security configurations analyzed in this step are not correctly set, then the SUB is *unacceptable* and it should not be considered secure (i.e., its security level is not acceptable for the benchmark user). On the other hand, if the device has no issue regarding this first set of security configurations, then the device is *acceptable* to be verified by applying the *risk quantification* step in which the risk exposure level (security level) of the device will be calculated. In practice, this step is related with the identifiable characteristics and properties that are considered *must have* security requirements for the target systems to have a security level higher than zero.

The concrete set of security configurations that are included in the *security qualification* step is highly dependent on the benchmark usage, on the activities performed by the device owner and on the required security level in his domain. For instance, if the benchmark is used in a company that wants to allow their employers to use their personal mobile devices to access the enterprise systems, then the concrete list of security configurations will be different for different levels of employees. For example, for a high-level officer it is normal to required storage encryption and the use of an alphanumeric passcode, while for a standard employee maybe only the use of any type of password is needed (assuming that a normal employee carries in is device less critical business information than a high-level officer).

The definition of the concrete set of user-defined security configurations that should be part of the security qualification is thus a major aspect that should be based on the most important (more risky) security configurations, according to the scenario and environment where the benchmark is being applied. Nevertheless, as there may be some scenarios where the *security qualification* phase is not required (i.e., there is no mandatory configurations to be checked), this phase can be seen as optional. In this case, the benchmark user can just set an empty list of mandatory security configurations.

Note that the *security qualification* step is platform independent, as it is based only in the comparison between the device configuration values and the expected ones. For example, at this stage, the P1 (Enable password) configuration has a boolean value: true, if its enabled, or false, if it is disabled. This way, checking if the configuration is well set consists in comparing primitive (integer, boolean, etc.) values (See Table A.1 for the specific values of each configuration). This step can be performed manually or automatically, although, at this stage, the values necessary for the analysis should have already been gathered (during the *configuration gathering* phase).

5.1.2 Risk quantification

The SUBs that pass the *security qualification* phase are eligible for *risk quantification*, which is responsible for calculating the benchmark measures, namely in the form of the approximate risk exposure that the user-defined security configurations poses to the device security. This way, after this stage, the benchmark user will have access to the calculated measures, so he/she should be able to rank and compare the SUBs.

The benchmark user is also responsible for defining which security configurations should be analyzed during the *risk quantification* phase. The main idea is that each configuration present in the set of user-defined security configurations must have a risk exposure portion in the device security. Moreover, for both stages, if two devices of the same platform but of different versions are being compared and one of them has a security configuration that is not available on the other (e.g. encrypt device storage is available for Android 4.0 but not for 2.3), the missing configuration should be assessed as incorrect (insecure).

A key point of our *risk quantification* approach, is the idea that the assessed security configurations should be related with threats and vulnerabilities of the system. In other words, these settings can be identified as potential contributors to the risk exposure level of the device, but without being decisive for the existence of a serious security flaw, which highlights the main difference between *security qualification* and *risk quantification.* By definition, the settings to be considered in this stage are usually not enough to allow attacks, on the other hand, they are partially related to known attacks or likely to present a high vulnerability. Thus, a system with a higher risk exposure in their incorrect settings (misconfigurations) must be rated as a low security.

5.2 Specifying benchmarks for security configurations

The act of benchmarking has been traditionally applied to computer performance [83, 89], and later extended to other computer science fields, such as dependability [47], resilience [102] and security [101]. A security benchmark is a standardized specification of a procedure to measure the security of computer systems (e.g. mobile devices) for comparative purposes. The security measurement should characterize the security level of these systems in such a way that enables their ranking. In fact, the most important goal of a security benchmark is to help users to distinguish the most secure (the less risky) component or system from among several choices in the same domain [57].

The key aspect that differentiates benchmarking from other experimental assessment techniques is that a benchmark is a standardized procedure that can be generically used to evaluate and compare different systems or components in a given domain. In this section, we propose a security benchmark for mobile devices regarding user-defined configurations. We first define the main benchmark elements (i.e., the benchmark target, the system under benchmarking and the benchmark management system) that are present in a security configuration benchmark. Then we define the components (i.e., measures, user-defined security configurations, experimental setup and procedure and rules) that compose a benchmark. Finally, we describe the properties that are essential to turn the benchmark into an acceptable process.

5.2.1 Benchmark elements

The elements of a security benchmark represent essentially the benchmark setup, thus defining the parts needed for implementing the whole benchmark. We propose the following main elements:

- **Benchmark Target** (BT): represents the system or component that is meant to be characterized. In our approach, the benchmark target are the user-defined security configurations, i.e. any setting, configuration or value that a user can define or set in his/her device impacting the overall security of the device (e.g. set a password to unlock the device or define a timeout to automatically lock the phone).

- **System Under Benchmarking** (SUB): represents a wider system that is used to run the benchmark target during the benchmarking experiments. This way, the SUB provides all the support needed to extract the measures from the BT. In a mobile device environment, the SUB includes the hardware, the operational system and the applications.

- **Benchmark Management System** (BMS): represents the system that is responsible for managing the benchmarking experiments. The goal is to make the execution of the benchmark a fully automated process [98]. However, the benchmark process can also be manual or semi-automatic (hybrid approach). Although an automated process is more practical and potentially faster, it is generally platform dependent. On the other hand, a manual approach is time consuming but less platform dependent.

Benchmarks can be provided in the form of a document [27] or as a computer program [83]. Although some adaptations to the SUB may be necessary, a benchmark in the form of a software is potentially ready to be executed by the user, while a benchmark in the form of a document has to be implemented before execution. On the other hand, benchmarks provided in the form of computer programs are normally more limited in scope than the ones provided in the form of documents. This is due to the fact that a computer program needs to be implemented assuming specific structure for the SUB, while a document specification can be designed based on a small set of assumptions regarding the services provided by the SUB [98].

Our proposal towards security configuration benchmarks for mobile devices is to follow the document approach. The key motivations are the larger platform independence and the flexibility for the user to develop its own tool to support the benchmarking process (although in this work we also provide the uSEA tool that can be used to support the process).

5.2.2 Benchmark components

The components of a benchmark conceptually define what has to be implemented on top of the elements defined before. The main components required are as follows:

- **Measures**: characterize the SUB security. The measures must be easy to understand and gather, representative and must allow the comparison between different systems in the benchmark domain;

- **User-defined security configurations**: represents the set of configurations that can be modified by the user (device owner) and that have some influence in the device security. This set may be divided in two to support both the *security qualification* and *risk quantification* phases (see Section 5.1);

- **Experimental setup**: describes the setup that the user must build (based on the benchmark elements defined in Section 5.2.1) to run the benchmark;
- **Benchmark procedure and rules**: describe the procedures and rules that must be followed by the benchmark user during the implementation and execution of the benchmark, namely in what concerns the collection of the required information and the execution of the *security qualification* and *risk quantification* phases.

Measures for a security configuration benchmark

The measures in a security configuration benchmark must be understood as a way to characterize the security level of the SUB in a relative way (e.g. to compare two mobile devices) and also to improve or tune the user-defined security configurations of the SUB. The measures should be easy to understand and representative. As mentioned before, our proposal is to include in measurement the notion of security risk, which is given by the impact of the successful exploitation of a vulnerability and by the probability of that vulnerability to be exploited [2], as discussed in Chapter 4.

The measures for a security configuration benchmark for mobile devices should be selected in a way that fulfills the following characteristics/goals:

- Should be calculated based on the user-defined security configurations;
- Focus on the risk that a user configuration poses to the system if not correctly set;
- Allow the characterization of the device security level in a relative manner;
- Should be easy to understand and gather by the benchmark users.

Although other measures can be considered, in this work we propose two alternatives. The first is the **number of misconfigurations** in a benchmarked device. Although the number itself does not have a direct relation with security (e.g., few risky misconfigurations can be worse than several harmless misconfigurations), it provides an indication on the settings that must be changed by the user in order to improve the security of a device.

The second and main measure is the **risk exposure**, which represents the sum of the risk exposure of each configuration, where higher values suggest a lower device security. As discussed in Chapter 4, this measure is based on the risk assigned to each user-defined security configuration, and summarizes the overall risk posed by the whole set of user-defined security configurations. In practice, it can be calculated using the following equation:

$$risk\_exposure = \sum_{i=1}^{size} W_{ci}(Insecure) \tag{5.1}$$

Where:

i is the identification number of the configuration;

ci represents a unique configuration;

$\mathbf{W}_{ci}$ is the weight of a unique configuration;

size represents the number of configurations that should be assessed;

Insecure is a selector stating if the configuration is well set or not, thus adding to the risk exposure (not well set) or not (well set).

In other words, equation 5.1 represents a sum of the risk exposure level of each security configuration. Such *risk_ exposure* measure can also be used to derive other measures, such as the **risk percentage**, according to the following equation:

$$risk\_perc = \frac{risk\_exposure}{\sum_{i=1}^{size}(W_{ci}(Secure) + W_{ci}(Insecure))} \tag{5.2}$$

Where:

Secure is a selector that is the reverse of Insecure.

In practice, equation 5.2 represents the risk percentage (in a scale from 0 to 1), calculated by dividing the *total_ risk* by the sum of the risks of each security configuration present in the set.

User-defined security configurations

The set of user-defined security configurations is the most important benchmark component. It should be complete, representative and verifiable enabling the characterization of the SUB security level. Works in the literature [27, 93], including our proposal presented in Chapter 3, already provide sets of security configurations for Android and iOS. In practice, those can be used as a starting point towards the definition of the security configuration set that should support the benchmarking process. An example on how user-defined configurations can be used in a security benchmarking context are discussed in Section 5.2.

Experimental setup

Figure 5.2 depicts the main elements of the experimental setup considering three distinct approaches. The benchmark target (BT) is the element that must not be altered regardless of the approach followed, thus requiring a non-intrusive implementation of the benchmark.

In the **manual approach** the BMS is the benchmark user, who is responsible to gather the user-defined security configurations from the SUB. This brings some advantages as there are configurations that can only be accessed via the user interface and for which there is no API available. However, there are also some disadvantages, namely related to the fact that this is prone to user errors due to the user experience and misinterpretations, which may influence the results.

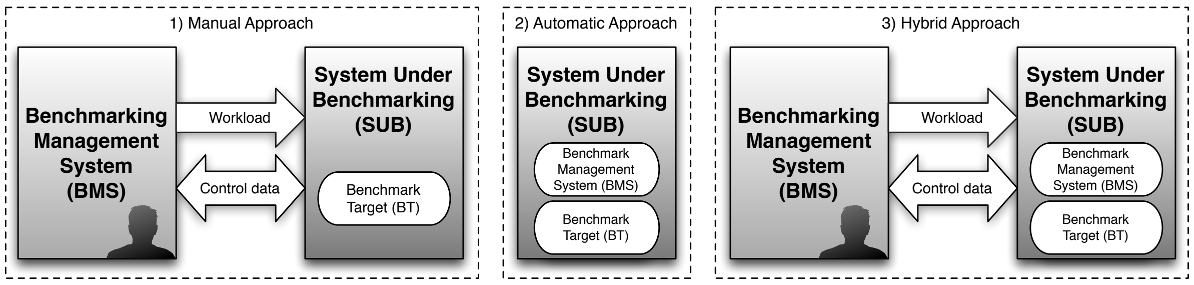
1) Manual Approach
Benchmarking Management System (BMS)
Workload
Control data
System Under Benchmarking (SUB)
Benchmark Target (BT)
2) Automatic Approach
System Under Benchmarking (SUB)
Benchmark Management System (BMS)
Benchmark Target (BT)
3) Hybrid Approach
Benchmarking Management System (BMS)
Workload
Control data
System Under Benchmarking (SUB)
Benchmark Management System (BMS)
Benchmark Target (BT)

Figure 5.2: Main elements of the experimental setup considering three approaches: A) Manual; B) Automatic; and C) Hybrid.

In the **automatic approach** the BMS is a tool responsible for assessing the BT. This has several advantages when compared with the manual alternative. First, once the BMS is implemented, it is much simpler and faster to execute the benchmark. Second, automatic analysis is less prone to user errors and misinterpretation. Finally, using an application for the analysis allows assessing some configurations that can not be accessed by manual means using the standard user interfaces provided by mobile operating system. However, this approach also has some drawbacks, such as the platform dependence and the effort required to implement it beforehand.

In the **hybrid approach** there is both a manual and an automatic BMS. This allows to assess a higher number of configurations as some of them are only accessed through manual analysis and others through an application. Therefore, an hybrid approach brings together the advantages of the others two approaches.

Benchmark procedure and rules

The procedure and rules that must be followed during the benchmark execution have to be specified during the definition of the security benchmark [98]. In practice, such procedures and rules should assure the benchmark to be executed in uniform conditions across different targets, and the system under benchmark to be tested with the same configurations used in realistic scenarios. For instance, the benchmark should not be executed after performing a factory reset in the SUB, as this will not reflect a realistic set of user-defined configurations.

An important rule for a security configuration benchmark is that the final measures and results must not vary if executed more than once in the same unmodified SUB. This is a big difference when compared with performance and dependability benchmarks, whose values commonly vary between distinct executions of the benchmark.

If the benchmark is based on an automatic BMS, then the SUB may need to have an Internet connection. An automatic BMS is capable to collect the required information from SUB and, for instance, store it in a remote database for further analysis. Note however that such Internet connection should not be mandatory as it is intrusive with regard to the device configuration (e.g., if a WIFI connection is required, then configuration N3 (*Disable WIFI when not needed) cannot ever be correctly set, which will reflect in the benchmark results*). On the other hand, such approach may improve the scalability and usability of the benchmarking process.

As our benchmarking approach is based on collecting the device configurations, the mobile device should not be under any other use during the benchmark execution, in order to provide more reliable results (if a configuration is changed during the benchmark execution, the results will not reflect the device real state). The BMS can make use of a third party application to extract the user-defined configurations from the device and then check which ones are misconfigured. Although this can be done in background, if there is any change in the configurations during the benchmark execution, the results will not portray the real security of the decide anymore.

As for the benchmarking steps, which define the sequence of activities that the benchmark user should perform, we propose the following ones:

1. **Defining the set of security configurations**, which should be specific for the SUB platform. Chapter 3 tackles this subject, presenting how to choose a representative set and defining a specific one for Android devices;

2. **Defining the risk of each security configuration** as mentioned before in Chapter 4. Other related works [39, 66] successfully used a risk based approach to assess the mobile device risks, however in a small configurations set.

3. **Defining the subsets of configurations for security qualification and risk quantification** dividing the set of configurations in two (see Section 5.2), one composed of mandatory configurations (the ones that must be well set in order to have a security level higher than zero), and the other including optional ones (the ones that will be used for calculating the *risk exposure* level);

4. **Preparing the experimental setup** for each target system being considered. For instance, if an automatic BMS is used, then the BMS software must be installed and the network must be configured as needed. On the other hand, if the manual BMS approach is chosen, the benchmark user should have access to the devices during the experiment (e.g., s/he should know the device password and other relevant information);

5. **Gathering the device configurations** values (by means of the BMS) from each SUB and for each user-defined security configuration;

6. **Calculating the metrics**, which consists in using the information collected during the previous step to conduct the *security qualification* of the devices and calculating the measures (based on equations 5.1 and 5.2) for the *risk quantification* phases.

Steps 1 to 3 are necessary to complete the security benchmark implementation, while steps 4 to 6 are related with the benchmark execution. After Step 6 the user should have all the information needed to compare and rank the systems under benchmarking (alternative configurations of different devices or of a same device), taking into account the configurations that are set in the mobile devices.

5.2.3 Benchmark properties

Regardless of the domain, a benchmark should satisfy a set pf properties in order to became a standard and/or to be accepted by the industry and the users community. In practice, a benchmark must be *representative*, *portable*, *repeatable*, *scalable*, *non-intrusive*, and *simple to use* [33, 100].

In order to report relevant results, a benchmark must represent real world scenarios in a realistic way [6]. In our work, **representativeness** is achieved by a realistic set of user-defined security configurations that should be gathered by the benchmark user, considering for example the set of configurations for Android devices presented in Chapter 3.

Portability is a very important property, as a benchmark should allow comparison between different targets and/or versions in a given domain. In practice, the user-defined security configurations are the component that has more influence on portability. This is due to the fact that the set of configurations may vary depending on the mobile device platform and versions of operational systems. For this reason, we propose an approach that the user can instantiate for his specific case.

Repeatability means that the benchmark must report similar (or even equal) results when executed more than once over the same target. In practice, small changes in measurements for some benchmarks domain are normal as it might be impossible to reproduce the same conditions concerning the target. For instance, performance benchmarks usually have small deviation in the measurements as the target may vary the amount of process, free memory and processor over the time. However, the security configuration benchmark is deterministic, in other words, it should not vary if executed in the same unmodified target.

Scalability can be seen as the property that enables the benchmark to evaluate systems of different sizes, as well as to evaluate a large number of systems. In practice, scalability enables the benchmark to handle a growing number of target sizes, which do not vary too much in the case of mobile devices. It also enables the benchmark to be executed on many devices.

A benchmark should require minimum or no changes in the benchmark target. **Non-intrusiveness** is a necessary property to avoid changing the architecture or behavior of the BT, which could lead to unwanted changes or bias in the results. In pratice, the Benchmark Management System (BMS) must collect the measurements from the System Under Benchmark (SUB), but it should not impact the Benchmark Target (BT) or the architecture of the SUB.

Simplicity of use is self explanatory. A benchmark that is difficult to understand, implement or execute will not be accepted by the industry or users community. Ideally, the benchmark should be provided as a computer program ready to be used or, if needed (e.g. for improving portability), as a document specifying in detail how the benchmark should be implemented and executed [100]. Our proposal follow a document-based approach (for improving portability), but with the support of the uSEA tool (to improve simplicity of use). This way, it can be defined as a semi-automated process whose advantages and disadvantages were discussed in Section 3.1.1.

5.3 Case study: A benchmark for Android devices

In this section we show in detail how to build a concrete security configuration benchmark for mobile devices following the approach discussed before. The target domain are Android devices and the usage scenario is the one discussed in Chapter 4.

Table 5.1: Set of Android security configurations and their respective risks.

ID	Security Configuration	Total Risk	Security Qualification	Risk Quantification
A2	Do not install root applications	5	✓	
P01	Enable password	5	✓	
P02	Require alphanumeric password	5		✓
P05	Do not make passwords visible	5		✓
P06	Erase data upon excessive password failures	5		✓
P11	Set timeout minutes for sleep	5	✓	
PE5	Do not root (“Jailbreak”)	5	✓	
S5	Disable unknown sources	5	✓	
A4	Do not install malware	4		✓
P04	Enable SIM card lock	4		✓
P07	Set a maximum password age	4		✓
P09	Insert minimum passcode length	4		✓
P10	Set a minimum number of complex characters	4		✓
P12	Enable pattern password	4		✓
PE1	Enable encrypt device storage	4		✓
PE3	Disable write permission on “/system” folder	4		✓
PE4	Disable write permission on “/data” folder	4		✓
S4	Disable developer options	4		✓
A3	Install anti-malware	3		✓
B1	Disable Java Script	3		✓
B4	Enable block pop-ups	3		✓
N03	Disable Wi-Fi when not needed	3		✓
N04	Disable bluetooth when not needed	3		✓
N07	Turn off personal hotspot when not needed	3		✓
N11	Disable bluetooth discoverability	3		✓
P08	Set number of password history	3		✓
P13	Disable visible lock pattern	3		✓
S1	Update firmware to latest version	3		✓
S6	Disable location services when not needed	3		✓
S9	Disable mock location	3		✓
B3	Disable cookies	2		✓
C4	Limit the number of MMS on the device	2		✓
C8	Limit the number of documents on the device	2		✓
N01	Remove saved Wi-Fi entries	2		✓
N02	Disable network notification	2		✓
N06	Enable airplane mode	2		✓
N10	Disable ICMP ping	2		✓
C3	Limit the number of SMS on the device	1		✓
C5	Limit the number of photos on the device	1		✓
C6	Limit the number of audios on the device	1		✓
C7	Limit the number of videos on the device	1		✓

5.3.1 Defining the set of security configurations

The definition of a representative set of security configurations is a fundamental part of our benchmarking approach. The set has to be verifiable (e.g. assessed by manual or automatic means), complete (e.g. include the largest number of security related settings possible), and clearly linked with security vulnerabilities (e.g. have a reason to be considered a security configuration).

For the present benchmark we adopted the list of security configurations discussed in Chapter 3, which can be obtained using the uSEA tool presented in that same chapter. Table 5.1 recalls that set of configurations. The different columns of the table will be discussed in the following subsections.

5.3.2 Defining the risk of each security configuration

User defined security configurations have difference relevance and impact in terms of security. In some cases, missing a singe key configuration can be worse than missing several harmless misconfigurations. In fact, not only the number of misconfigurations impacts the device security, but also the risk associated with each one of them.

To perform this step of the benchmark design, we followed the approach for assessing the security risk posed by user-defined configurations in Android devices that was previously proposed in Chapter 4. The *Total Risk* column of Table 5.1 recalls the defined risk values.

5.3.3 Defining the subsets for security qualification and risk quantification

Our security configuration benchmarking approach includes two phases: *security qualification* and *risk quantification*. Consequently, the benchmark user needs to select which security configurations are part of each assessment. The problem is that this strictly depends on the benchmark usage environment. For example, if the benchmark is used in a university environment, it probably will be less stringent than if used by the army. In this case study, we are considering the scenario described in section 4.1, but distinct benchmark users may select different sets for each stage.

As the *security qualification* stage is the more restrictive one, we decided to consider in it some of the security configurations with the greatest *Total Risk*. In practice, we selected five security configurations that must be correctly set on the SUBs in order for them to qualify. As shown in Table 5.1, the chosen configurations (A2, P01, P11, PE5 and S5) belong to the most severe class (5). Note that, although P02, P05 and P06 belong to that same risk class, they were not selected for the *security qualification* phase (we consider them not mandatory in our scenario). This means that, in this context, enabling a password (P01) and setting a timeout for automatically locking the device (P11) are the *must have* configurations in the password category, while the others (P02, P05 and P06) are seen as complementary mechanisms to be considered in terms of the risk exposure calculation.

The *risk quantification* is the phase that allows distinguishing the devices that demonstrated to be suitable after the previous phase. For this, we recommend to consider all the remaining user-defined security configurations (i.e. the ones that were not assessed during *security qualification*), although other benchmark users may decide to restrict the list of configurations being considered.

5.3.4 Preparing the experimental setup

The preparation of the experimental setup should be a simple task, as we intend to execute the benchmark in a large number of devices. In order to support this, we used the automatic BMS approach, which, in our case, is based on the uSEA Android application available at Google Play store [92]. As mentioned before, this application is based on a client-server model in which the client runs in an Android device and the server in a cloud environment. The client extracts the user-defined configurations and presents the results of the security analysis to the user. The data collected from the device is sent to the server by invoking a web service that replies with the results of the analysis.

Preparing the experimental setup consists into two steps: 1) install the application through the Google Play store; and 2) configure an Internet connection, which can be both WiFi or cellular data network. These tasks are the only ones necessary to run an automatic BMS over a SUB. However, if a benchmark user decides to use a manual approach, he needs to have at least: 1) the device (SUB) password, in case that it gets locked; 2) a checklist to remember the security configurations that should be assessed and take notes about the results; and 3) held the device during the benchmarking process, which in a manual analysis can take up to 1 hour. In fact, the manual analysis is more intrusive, as the benchmark user needs to have access to the device for a longer period of time.

5.3.5 Gathering the device configurations

One of the benchmark properties is that it should be easy to use, therefore, the benchmark execution should be simple. Using an automatic BMS simplifies the benchmark execution, as it is as simple as starting an application and accepting its user agreement during the first execution. An automatic benchmark can take up to few minutes and does not need the user intervention. After execution, the measurements retrieved from the device are stored in a cloud environment and these data become persistente and available for *calculating the metrics* at different moments in the future (i.e., can also be used to analyse the evolution of the configurations of devices).

5.3.6 Calculating the metrics

The final step consists of compiling the results obtained before in the form of measures by using equations 5.1 and 5.2. Calculating the SUB risk exposure is a simple task once the risk exposure of each configuration is already defined. Briefly, it consists of comparing the device configurations with a template (i.e. the correct value of each configuration), leading to the identification of the device misconfiguration. As the risk exposure of each single misconfiguration is already defined, the equation consists in simply summing those values.

The calculated metrics can then be used to compare the SUBs, which may include different devices or multiple (alternative) configurations of a single device. The benchmark user is responsible for such comparison, which is obviously based on the metrics calculated. Although the metrics can be easily used to rank different SUBs, there is one aspect that needs to be addressed: *how to distinguish devices when the risk-exposure metric is very close?* In fact, this is a frequent issue in benchmarking, which is commonly addressed by the definition of a threshold that allows distinguishing SUBs. A common value is 2% [97], which we also adopt in this work (although the benchmark user may consider other values). In other words, we consider devices that have a risk exposure difference lower than 2% to have an equivalent security (from the perspective of the user-defined configurations). In order to untie devices with equivalent risk exposure (inside the 2% threshold), we propose the following criteria (the benchmark user may opt to not use these criteria and consider the devices equivalent): first, consider the number of misconfigurations in the very high

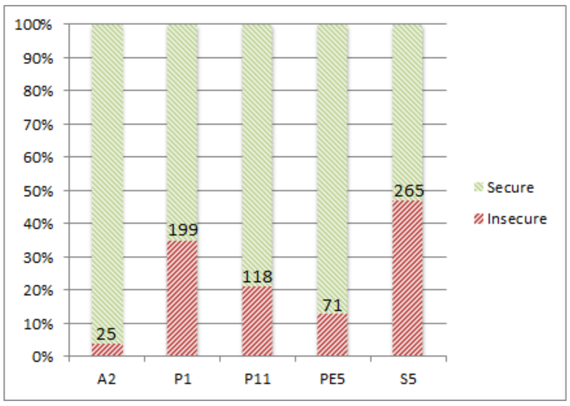

Figure 5.3: Analysis of the five security configurations that are part of the security qualification step.

severity class (the device that has the lower number is considered more secure); if the devices have the same value in that class, then move to the next class of severity and so on. If the devices remain with the same values they should be consider equal.

5.4 Experimental evaluation and results

To demonstrate the proposed benchmark, in this experimental evaluation we analyzed the user-defined security configurations of 561 Android devices. The goal is to show how the benchmark can be used to compare different real Android devices regarding their configurations.

The first step is to identify the devices that have the minimum security level necessary for the *security qualification* phase. In the present case, this requires checking configurations A2, P01, P11, PE5 and S5. Figure 5.3 illustrates the percentage of secure and insecure devices regarding these five user-defined security configurations (the number of devices that neglected a given configuration is also depicted in each bar). Note that, at least, a third part of users fails to correctly configure configurations P1 and S5, meaning that a great part of the users have important and severe issues in their configurations. Of the 561 devices analyzed, only 137 qualify to the second phase of the benchmarking process, while 242 of them fail in 1 of the 5 configurations, 132 fail in 2 configurations and, finally, 50 devices fail in 3 or more cases. It is important to emphasize that these numbers should be a reason for concern, since a large portion of the users (∼75%) are neglecting major security configurations, thus failing at the first phase of our benchmark.

The *risk quantification* phase was applied to the 137 devices that qualified . Table 5.2 shows the configurations for the best and worst devices in our data set regarding their security, as well as the most common security configurations. The device with the best configurations has a total *number of misconfigurations* equals to 15, a *total risk* of 44 and a *risk percentage* of ∼32.83%. These values were calculated considering the risk of each security configuration, as presented in Table 5.1 and equations 5.1 and 5.2. At the same time, the device with the worst configuration has a total *number of misconfiguration* equals to 25, a *total risk* of 74 and a *risk percentage* of ∼55,22. Finally, the common

Table 5.2: Presentation of the security configuration set of the best and worst configuration of the dataset, as well the most common misconfigurations.

	A2	A3	A4	B1	B3	B4	C3	C4	C5	C6	C7	C8	N1	N2	N3	N4	N6	N7	N10	N11	P1	P2	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	PE1	PE3	PE4	PE5	S1	S4	S5	S6	S9
Best config.	✓		✓			✓		✓		✓		✓	✓		✓	✓		✓	✓	✓	✓			✓					✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓
Common config.	✓		✓			✓	✓	✓			✓	✓				✓		✓		✓	✓								✓	✓	✓			✓	✓	✓		✓	✓	✓	✓
Worst config.	✓	✓				✓		✓								✓		✓		✓	✓									✓		✓		✓	✓	✓		✓	✓		✓

Table 5.3: Risk exposure and risk percentage of 10 devices chosen from the dataset.

Device ID	Class 1	Class 2	Class 3	Class 4	Class 5	Risk Exposure	Risk Percentage
1649	3	4	9	20	10	46	34.33
746	1	6	12	24	10	53	39.55
2649	1	8	12	20	15	56	41.79
1633	3	8	15	16	15	57	42.54
955	3	8	18	20	10	59	44.03
838	1	8	15	20	15	59	44.03
2791	2	10	18	16	15	61	45.52
1547	1	8	18	20	15	62	46.27
2707	2	8	18	20	15	63	47.01
2765	3	8	18	24	15	68	50.75

configurations (i.e. the ones that appear in more than half of the devices analysed), have a total of 20 misconfigurations, a *total risk* of 61 and a *risk percentage* of ∼45,52.

As shown, even the device with the best configuration has important security issues. For instance, it fails on configurations P02 and P06, which are part of the most severe risk class (5), being therefore prone to well known cyber-attacks (i.e. brute force). In addition, the set of most common configurations are closer to the worst configuration than to the best one, showing that a great part of users have a poor adoption of the best practices while configuring their devices. By supporting this kind of analysis, our benchmark approach should be seen not only as a way to compare and rank the mobile devices, but also as a way to find out the major misconfigurations mistakes that the users make. This way, it becomes a key tool to improve the security level of the systems under benchmarking.

Taking an alternative perspective, Table 5.3 presents the risk exposure of 10 devices chosen randomly from our dataset (we limit the analysis to 10 because it would be impractical to present a table with 561 and discuss all the results). The *Device ID* column represents the identification number that each device has in the dataset; the columns *Class 1 to 5* represent the risk exposure in each class of severity; *Risk Exposure* and *Risk Percetage* respectively represent the metrics calculated using equations 5.1 and 5.2. The 10 devices presented passed the qualification phase and are ranked by the Risk Exposure/Risk Percentage column, where higher values represent more risk. When two devices have the same risk exposure, the severity class is used to untie them (we are not considering the 2% threshold in this analysis). Take as example devices 955 and 838 in Table 5.3: both have a risk exposure value of 59, however, device 955 has 2 misconfigurations in class 5 (the value 10 represents the weight 5 multiplied by 2 misconfigurations) and device 838 has 3 misconfigurations in that same class (the value 15 represents the weight 5 multiplied by 3 misconfigurations). This way, we can state that device 838 has more important issues that should be addressed in order to improve the device security when compared with device 955.

Figure 5.4 depicts the same 10 devices in another perspective, showing the precedence of them considering the distance of, at least, 2% in the risk exposure percentage metric.

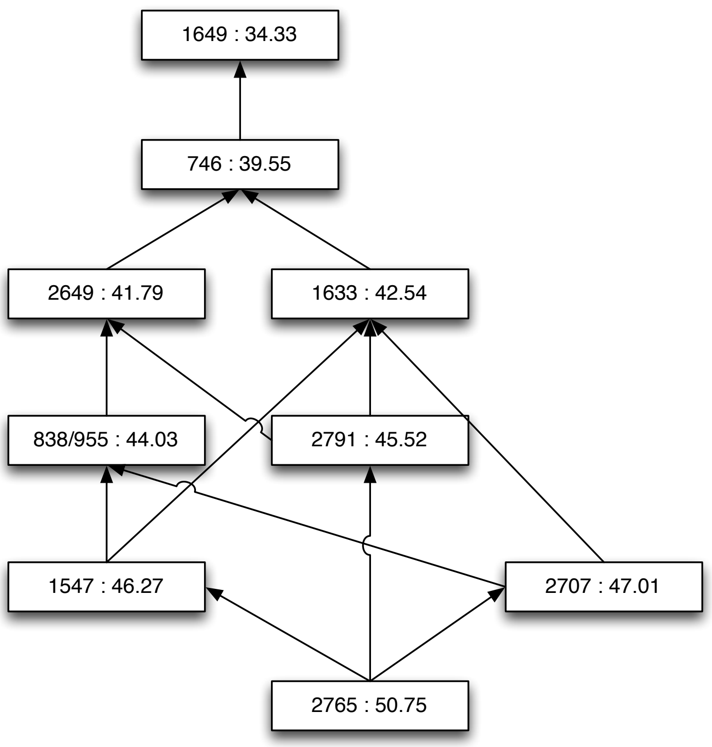

Figure 5.4: Devices ranking showing the precedence of them.

Take device 746 as an example, we can state that it is more secure then devices 2649 and 1633 because it has 39.55% of risk exposure, which is more than 2% lower than devices 2649 (41.79%) and 1633 (42.54%). On the other hand, devices 2649 and 1633 are very similar, with 0,75% of difference in the risk exposure metric, thus no conclusion can be drawn. In this case, the number of misconfigurations in each severity class can be considered to untie them. Because devices 2649 and 1644 have 15 points of risk exposure in class 5, class 4 needs to be considered: device 1633 has 16 risk exposure points, while device 2649 has 20 points, thus we can state that device 1633 is more secure than the device 2649. This perspective is interesting when one wants to choose between the devices that are more secure from a set of benchmarked devices.

5.5 Validating the proposed security benchmark

The word validation is used in this work in the sense that one needs to gain confidence on the benchmark properties through a set of representative experiments. Thus, this section discusses the validation of the security configuration benchmark properties previously listed in Section 5.2.3.

5.5.1 Representativeness

Representativeness is a key property in the specification of a benchmark, as its results must characterize the addressed aspects of the target system in a realistic way [46]. In the context of security configuration benchmarking, representativeness is related with the measures and the use of a realistic set of user-defined security configurations.

The measures should be useful and meaningful for the benchmark users and need to reflect the security level of the SUBs regarding user-defined security configurations. in practice, the measures must allow to easily compare systems. In this work, we propose three measures (number of misconfigurations, risk exposure and risk percentage), as discussed in Section 5.2.2), which we believe are easy to understand, as demonstrated by the results in Section 5.4.

The set of user-defined security configurations should be complete enough to be representative. The set used in our benchmark has been extracted from the well known CIS security benchmarks [27] and complemented by an additional set of security configurations (see discussion in Chapter 3). Note however that, the set used in this work is just a recommendation and can be complemented by specific ones according to the specific context and needs of the benchmark user.

5.5.2 Portability

Portability is the property that allows a benchmark to be used to compare different systems of a given domain, thus increasing the number of users and, consequently, its acceptance. In the context of mobile devices, an ideal benchmark should be able to compare any mobile device, independently of the hardware and operating systems used.

The components of our benchmark are described in a generic way, which improves portability. However, the set of user-defined security configurations will always be different for each operational system platform. Thus, comparing the security level of mobile devices regarding user-defined security configuration, makes sense only in the context of a given platform (e.g. Android vs. Android and iOS vs. iOS). In practice, our approach supports the comparison of mobile devices with different operational system versions within the same platform.

5.5.3 Repeatability

Repeatability is the property that guarantees that the security configuration benchmark reports the same result when executed more than once in the same SUB (assuming that the configurations are not modified). A repeatable benchmark provides credible results, as different users using the same approach will reach the same conclusions.

In our approach, one of the rules is that the measures must not vary when the benchmark is executed more than once in the same unmodified SUB (see Section 5.2.2). In fact, if the measures for a given device change during a benchmarking campaign, this should be seen as an error: if an automatic BMS approach is used, then the software may have some error; otherwise, if a manual BMS approach is considered, then the benchmark user could have made a mistake while extracting the information from the device.

To validate the repeatability of our benchmark, we ran it several times. In practice, the uSEA tool supports the process of information gathering and its results were analysed manually in several devices to attest correctness. After confirming that the uSEA tool was correctly extracting the configuration values, we executed the benchmark several times to confirm the determinism of the results.

5.5.4 Scalability

Scalability can be seen as the property that enables the benchmark to evaluate systems of different sizes, as well as to evaluate a large number of systems. In general, mobile devices have more or less the same size and a similar number of security configurations (this is what we observed for the Android devices in our experimental study). However, the number of systems that should be evaluated in different benchmarking campaign may vary according to the organization or context in which the benchmark is being used.

In this work, we presented an experimental evaluation over 561 Android devices using an automatic BMS approach. The usage of the uSEA application to gather the configurations for the devices helps increasing the scalability of our approach. In fact, the bottleneck of a benchmarking campaign is the data gathering activity, which may require a long time (especially in the case of a manual analysis).

To assess scalability we executed the benchmark over several different Android devices analyzing the accuracy of the results. In practice, we collected information from 561 devices, which executed the uSEA application 3,054 times with only 22 Crash and ANRs (Android Not Responding) reported (this represents 0.72% of errors during execution) by the Google Play statistics developer console.

5.5.5 Non-intrusiveness

Non-intrusiveness is related with the changes needed in the target systems before the benchmark execution. While intrusion is a large problem for other types of benchmarks, that is not the case for a security configurations benchmark, as it normally requires simple or no modifications at all. In practice, an intrusive benchmark may affect the results as it will characterize the modified system and not the system used in the field.

In our benchmark, it is not necessary to modify the operating system to assess and gather the values for each configuration in a mobile device context. However, a BMS should be installed on the SUB to gather the configurations values in case of an automatic approach. On the other hand, for the manual approach there is no need to change the SUB, but this raises other issues related to intrusiveness, namely in what regards privacy, as the user will have full access to the device during the benchmarking process.

To validate our benchmark, we verified if the process does influence the results and measurements by performing a manual analysis before the automatic analysis and then comparing the results. In practice, our approach has some intrusiveness (besides the uSEA installation), which is related with the need for an Internet connection to retrieve the results (that are stored in a database on the cloud).

5.5.6 Simplicity of use

Simplicity of use is a property that, in some way, is similar to cost, but expressed in terms of effort required to develop and run the benchmark and obtain the results [45]. The acceptability of a benchmark is related to its simplicity, as users tend to ignore benchmarks that require a high effort to develop and run. Therefore, running the benchmark must be cost effective and practical.

The complexity of a benchmark can be assessed during its implementation and execution. The effort to implement the BMS, in the case of an automatic approach, is a key aspect to be considered while analyzing the complexity. Although, there is a need to implement a BMS, once the benchmark is stabilized, it may be supported by benchmarking organizations (e.g. SPEC [83], TPC [89], CIS [27]) and used in different contexts and environments, thus reducing the cost and time (since the implementation will be performed only once for different use cases). The benchmark specification also plays an important role in the complexity of the implementation, thus it should be clear and complete in such a way that it do not raises major questions during the implementation phase.

In the case of our benchmark, we validated simplicity by observing that each uSEA is easy to install and use, and each analysis took in average 20.3 seconds. The other steps (security quantification and risk qualification) consist in the simple verification of primitive values (e.g. boolean and integers) against a template, which can be easily supported by a spreadsheet.

5.6 Conclusion

In this chapter, we presented a novel approach for security configuration benchmarking of mobile devices, including a concrete benchmark and an experimental evaluation for the specific case of Android devices. The properties of the proposed benchmark have been discussed, namely representativeness, portability, repeatability, scalability, non-intrusiveness, and complexity of the benchmark approach.

The benchmark process puts together all the contributions of this thesis. In fact, it is supported by the set of user-defined security configurations (see Chapter 3), by the uSEA tool (see Chapter 3) and by the risk assessment approach (see Chapter 4). By putting all these contributions together with a well-defined procedure, the benchmark allows comparing the security level of distinct mobile devices in what regards the user-defined security configurations.

Results shows many users neglect major security configurations, thus failing at the first phase of our benchmark. In practice, following security configuration best practices is not a silver bullet, but clearly most users disregard them and some Android installations make it hard for users to change the existing configurations. This suggests that we must increase awareness among users, manufacturers, and researchers. Users must know the threats they are facing and understand that their actions greatly influence security. Manufacturers must understand that the current state of practice is not good enough and that they need to do better. Finally, researchers should direct their efforts toward providing new solutions that improve the security mechanisms and configurations of mobile devices. Just as individuals share the responsibility for their personal safety, the security of the information managed by a mobile device strongly depends on its user. End users do not appear to understand cyber security sufficiently to manage their security configurations. Therefore, researchers and manufacturers should strive to improve the current scenario and avoid or reduce the perils of mobile device configuration.

Organizations that want to enable the usage of personal mobile devices to access corporate systems, data and information, are the potential users of our approach. In fact, the benchmark enables them to perform security qualification, ensuring a minimum security level on the mobile devices, and risk quantification, discovering the main misconfigurations issues of the SUBs and ranking the security of the devices. This may potentially lead to an improvement of the user-defined security configurations of the mobile devices used to access their cyber-systems.

Chapter 6

Conclusions and future work

This work presented a novel security configuration benchmark approach for mobile devices. The benchmark is supported by a representative set of user-defined security configurations that impact the device security and by a risk analysis that characterizes the risks posed by each configuration. This work represents a first step towards tackling a gap that exists in the literature with respect to methodologies to evaluate the security of mobile devices, in particular regarding user-defined security configurations, which clearly impact security. For example, correctly managing the configuration of a computational system increases the security level by reducing the potential attack surface (as is the case of turning of the WiFi) and/or increasing the bar of effort to break into the system (as the case of setting a password to log in).

The main contributions of this work include the identification of a set of user-defined security configurations that influences the device security; a detailed analysis of the configurations of Android devices in the field; an approach to calculate the risk that each configuration poses to the device security; and a benchmark thatcan be used to rank and compare the security level of mobile devices based on their configurations.

According to our observations, most Android users do not configure their devices taking into account the most relevant security recommendations, being thus a potentially vulnerable target to attackers that want to steal confidential or sensitive information. This suggests that Android manufacturers should provide better security settings in advance to protect more users by default.

Correctly setting all available configurations does not prevent all possible attacks against mobile devices, but for sure increases the bar of effort necessary for an attacker to be successful in his intent. For instance, correctly setting user-defined security configurations does not avoid attacks that exploit vulnerabilities in the operating system (e.g. that allow unlocking a device without entering any password due to an error in the home application) or in the hardware (e.g. that allow entering flash mode to change the operating system), but there are many known vulnerabilities that are avoided by correctly setting the security configurations. Take as an example the vulnerability CVE-2015-6618 extracted from the Common Vulnerabilities and Exposure database [60]: *"Bluetooth in Android 4.4 and 5.x before 5.1.1 allows remote attackers to execute arbitrary code"*. In this case, if the bluetooth is disabled, as stated by our recommendations (or enabled only when strictly needed), such vulnerability can not be exploited, even if the device has a vulnerable version of the Android firmware. This reinforces the idea that a user that follows the best practices of security recommendations has a device less vulnerable to cyber attacks.

It is important to note that a device that correctly implements all the security recommendations discussed in this work will have some of its features switched off. Although facing a low risk of being attacked, such device might become unusable in many scenarios. This way, the responsible for maintaining the security of the device (e.g. the device owner) should consider the trade-off between security and usability. For achieving this, the requirements to be fulfilled have to be understood beforehand (i.e. the essential features should be identified) and the results of the risk analysis should be used wisely to decide how to configure the device taking into account the security risks faced.

This work provides valuable information for those who need to evaluate (users and manufacturers), assess or control the security configurations of Android devices. **Users** greatly influence the overall security of their devices, mobile or otherwise. They are responsible not only for creating and manipulating the information stored on the device, but also for how the device is used. Previous research suggests that users are concerned about the privacy and security of their mobile devices when performing privacy-sensitive tasks, especially those involving money. However, many users follow digital trends regardless of their security implications. It is therefore important to make users aware that misconfigurations can open security vulnerabilities on their devices. On the other hand, **Manufacturers** should help users making more informed security decisions. Limiting user configurations might enhance device security, but decreases their ability to customize the devices (which also influences market acceptance). Nevertheless, manufacturers should change the value of some factory-defined configurations so that more users have safer devices. For example, a key area that requires manufacturers to change their approach is authentication: many users do not use a login mechanism on their devices because they perceive such mechanisms as inconvenient and annoying. The most common login methods available are PINs, mainly used on iOS and consisting of four-digit passcodes; passwords, commonly used on PCs; and pattern passwords, which are mainly used on Android devices and involve drawing a pattern on the touchscreen. The problem with these methods is that users must remember a code and thus tend to choose one that is easy to remember, such as a birth date, a dictionary word, or a simple pattern. A recent alternative is fingerprint identification, which is becoming the preferred choice for many users because of its convenience.

6.1 Main achievements

List of publications:

1. Qualis A1: Daniel Vecchiato, Marco Vieira, and Eliane Martins. A security configuration assessment for android devices. In Proceedings of the 30th Annual ACM Symposium onApplied Computing, SAC'15, pages 2299–2304, New York, NY, USA, 2015. ACM [93].

2. Qualis A2: Daniel Vecchiato and Eliane Martins. Experience report: A field analysis

of user-defined security configurations of android devices. In Software Reliability Engineering (ISSRE), 2015 IEEE 26th International Symposium on, pages 314–323, Nov 2015 [90].

3. Qualis A1: Daniel Vecchiato, Marco Vieira, Eliane Martins, "The Perils of Android Security Configuration", Computer, vol.49, no. 6, pp. 15-21, June 2016, `doi: 10.1109/MC.2016.184`. [94].

4. Qualis A2: Daniel Vecchiato, Marco Vieira and Eliane Martins. Risk Assessment of User-Defined Security Configurations for Android Devices. In Software Reliability Engineering (ISSRE), 2016 IEEE 27th International Symposium on, Oct 2016 [95].

5. Qualis B3: Daniel Vecchiato and Eliane Martins. Benchmarking user-defined security configuration of mobile devices. In Dependable Computing (LADC), 2016 7th Latin-American Symposium on, Oct 2016 [91].

Software prototypes developed:

1. Daniel Vecchiato, Eliane Martins, and Marco Vieira. Security assessment - android apps on google play. `https://play.google.com/store/apps/details?id=br.unicamp.ic.lsd.securityassesment`, 2014.

6.2 Threats to validity

We are aware that our work has limitations and that the proposed approaches are largely based on expert knowledge and not on precise definitions. This way, there are some threats to validity that should be emphasised:

1. The list of user-defined security configurations may not be complete, although it was adopted from the well known CIS security benchmarks [27] and complemented with the analysis of other works in the literature [10, 13, 20, 50]. Anyway, adding new configurations does not impact the approaches proposed and is just a matter of repeating the analysis (including the risk likelihood and impact);

2. The uSEA tool can generate false-positives and false-negatives trying to identify root apps, malware and antivirus [93]. However, it provides a good estimate of the presence of these kind of applications. To address this subject, works focused on the malware identification should be integrated with the uSEA tool.

3. The weights postulated for the severity classes directly influence the calculation of the risk exposure (for example, doubling the values would increase the distance between classes and would further stress the more severe ones). Finding the ideal weights requires assigning and comparing different sets of weights for the five classes, which is in the scope of future work. The postulated weights are widely used in traditional risk analysis and we are confident that they provide a good basis for characterizing risk exposure.

4. Without further validation, the conciliation of the opinions of the multiple experts may result in a classification that is not directly related to any of the experts. However, the use of decision theory to combine divergent opinions allows obtaining a consistent result, as also stated in other works in different domains [26]. The perception of more experts can be considered in order to improve the approach.

6.3 Future work

This work is a contribution towards the security assessment and benchmarking of user-defined configurations of mobile devices and opens new possibilities for research, including:

- **Instantiate the approach to other mobile platforms**. The approach described is generic enough to be applied in other mobile platforms, but a deeper analysis is needed to define in detail the adaptations that should be made. The initial perception is that the set of security configurations should be adapted to the desired domain and then the risk analysis and benchmark process should be re-executed considering this new set. As the uSEA tool was developed specifically for the Android platform, similar tools should be implemented targeting different platforms;

- **Improve the risk analysis by adding experts.** A detailed analysis of the experts perception should be performed in order to find an optimal threshold between cost and benefits of adding more experts to improve the risk assessment. Other approaches, able to converge the opinions of the experts should be investigated, confronting them with the MCDA/GRIP approach. In practice, a comparative analysis of different techniques can enhance the process of defining the weights postulated by the experts.

- **Apply other approaches to the risk analysis**. The risk analysis using MCDA to generate the risk level of each security configuration may be replaced by other approach of risk analysis that provides more accurate results (e.g. historical data, if available). This way, future work should be done to compare different approaches in this context and evolve the solution as new techniques appear and the technology advances.

- **Propose a security certification process for mobile devices**. The security benchmark proposed in this work is able to compare different mobile devices in a defined scenario. Future works should expand this approach to a certification process that considers, beyond the environment where the device is inserted, the role that the device owner plays in that environment and the value of the accessed assets.

This work is simply a step forward into a longer journey towards defining practical mobile devices security assessment techniques and tools that can be used for measuring security compliance and improving the overall information security stance for mobile devices.

Bibliography

[1] Haider Abbas, Louise Yngström, and Ahmed Hemani. Security evaluation of it products: bridging the gap between common criteria (cc) and real option thinking. In *Proceedings of the World Congress on Engineering and Computer Science 2008, San Francisco, CA, USA, 22-24 October*, pages 530–3, 2008.

[2] Christopher Alberts. Common elements of risk. Technical Report CMU/SEI-2006-TN-014, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2006.

[3] Christopher Alberts, Audrey Dorofee, James Stevens, and Carol Woody. Introduction to the octave approach. *Pittsburgh, PA, Carnegie Mellon University*, 2003.

[4] Christopher J. Alberts and Audrey Dorofee. *Managing Information Security Risks: The Octave Approach.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[5] Ståle Amland. Risk-based testing:: Risk analysis fundamentals and metrics for software testing including a financial application case study. *Journal of Systems and Software*, 53(3):287 – 295, 2000.

[6] N. Antunes and M. Vieira. Benchmarking vulnerability detection tools for web services. In *Web Services (ICWS), 2010 IEEE International Conference on*, pages 203–210, July 2010.

[7] N. Antunes and M. Vieira. On the metrics for benchmarking vulnerability detection tools. In *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*, pages 505–516, June 2015.

[8] Zarni Aung and Win Zaw. Permission-based android malware detection. *International Journal of Scientific and Technology Research*, 2(3):228–234, 2013.

[9] Adam J. Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M. Smith. Smudge attacks on smartphone touch screens. In *Proceedings of the 4th USENIX Conference on Offensive Technologies*, WOOT'10, pages 1–7, Berkeley, CA, USA, 2010. USENIX Association.

[10] Noam Ben-Asher, Niklas Kirschnick, Hanul Sieger, Joachim Meyer, Asaf Ben-Oved, and Sebastian Möller. On the need for different security methods on mobile phones.

In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, pages 465–473, New York, USA, 2011. ACM.

[11] Sven Bugiel, Lucas Davi, Alexandra Dmitrienko, Thomas Fischer, Ahmad-Reza Sadeghi, and Bhargava Shastry. Towards taming privilege-escalation attacks on android. In *NDSS*, volume 17, page 19, 2012.

[12] Jeffrey Burt. Byod trend pressures corporate networks. *eweek*, 28(14):30–31, 2011.

[13] Erika Chin, Adrienne Porter Felt, Vyas Sekar, and David Wagner. Measuring user confidence in smartphone security and privacy. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, SOUPS '12, pages 1:1–1:16, New York, USA, 2012. ACM.

[14] Domenico Cotroneo, Luigi De Simone, Antonio Ken Iannillo, Anna Lanzaro, and Roberto Natella. Dependability evaluation and benchmarking of network function virtualization infrastructures. In *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, pages 1–9. IEEE, 2015.

[15] John G Dawes. Do data characteristics change according to the number of scale points used? an experiment using 5 point, 7 point and 10 point scales. *International journal of market research*, 51(1), 2008.

[16] Afonso Comba de Araújo Neto. *Security Benchmarking of Transactional Systems*. PhD thesis, University of Coimbra, Coimbra, september 2012.

[17] Android Developers. Android 5.1 apis. `developer.android.com/intl/pt-br/about/versions/android-5.1.html#multisim`, 2016. Accessed: 2016-05-02.

[18] Android Developers. Dashboards. `developer.android.com/intl/pt-br/about/dashboards/index.html`, 2016. Accessed: 2016-05-02.

[19] Android Developers. Device administration. `developer.android.com/guide/topics/admin/device-admin.html`, 2016. Accessed: 2016-02-15.

[20] Alessandro Distefano, Antonio Grillo, Alessandro Lentini, and Giuseppe F Italiano. Securemydroid: enforcing security in the mobile devices lifecycle. In *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, page 27. ACM, 2010.

[21] Jeff Drew. Managing cybersecurity risks. *Journal of Accountancy*, 214(2):44–48, 2012.

[22] EE. 10 million mobile devices lost over the last year puts business data at risk, says ee. `http://ee.co.uk/our-company/newsroom/2014/03/26/10-million-mobile-devices-lost-over-last-year-puts-business-data-at-risk`, 2014. Accessed: 2014-12-05.

[23] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, pages 1–6, Berkeley, USA, 2010. USENIX Association.

[24] William Enck, Damien Octeau, Patrick McDaniel, and Swarat Chaudhuri. A study of android application security. In *Proceedings of the 20th USENIX conference on Security*, SEC'11, pages 21–21, Berkeley, CA, USA, 2011. USENIX Association.

[25] Adrienne Porter Felt, Matthew Finifter, Erika Chin, Steve Hanna, and David Wagner. A survey of mobile malware in the wild. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, SPSM '11, pages 3–14, New York, NY, USA, 2011. ACM.

[26] José Rui Figueira, Salvatore Greco, and Roman Słowiński. Building a set of additive value functions representing a reference preorder and intensities of preference: {GRIP} method. *European Journal of Operational Research*, 195(2):460 – 486, 2009.

[27] Center for Internet Security. Security benchmarks. `http://benchmarks.cisecurity.org`, 2016. Accessed: 2016-06-01.

[28] GAO. Information security better implementation of controls for mobile devices should be encouraged. `http://www.gao.gov/assets/650/648519.pdf`, 2012. Accessed: 2016-02-10.

[29] Stephen R Garner et al. Weka: The waikato environment for knowledge analysis. In *Proceedings of the New Zealand computer science research students conference*, pages 57–64. Citeseer, 1995.

[30] Inc Gartner. Gartner says worldwide smartphone sales recorded slowest growth rate since 2013. `gartner.com/newsroom/id/3115517`. Accessed: 2016-02-12.

[31] Inc Gartner. Gartner says worldwide tablet sales grew 68 percent in 2013, with android capturing 62 percent of the market. `gartner.com/newsroom/id/2674215`. Accessed: 2015-09-09.

[32] Inc Gartner. Gartner says global smartphone sales to only grow 7 per cent in 2016, 2016. Accessed: 2016-05-30.

[33] Jim Gray. *Benchmark Handbook: For Database and Transaction Processing Systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.

[34] Adel Guitouni and Jean-Marc Martel. Tentative guidelines to help choosing an appropriate {MCDA} method. *European Journal of Operational Research*, 109(2):501 – 521, 1998.

[35] S.T. Halkidis, A. Chatzigeorgiou, and G Stephanides. A qualitative evaluation of security patterns. In Javier Lopez, Sihan Qing, and Eiji Okamoto, editors, *Information and Communications Security*, volume 3269 of *Lecture Notes in Computer Science*, pages 132–144. Springer Berlin Heidelberg, 2004.

[36] S.T. Halkidis, N. Tsantalis, A. Chatzigeorgiou, and G. Stephanides. Architectural risk analysis of software systems based on security patterns. *Dependable and Secure Computing, IEEE Transactions on*, 5(3):129–142, July 2008.

[37] Benjamin Halpert. Mobile device security. In *Proceedings of the 1st annual conference on Information security curriculum development*, InfoSecCD '04, pages 99–101, New York, NY, USA, 2004. ACM.

[38] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108, 1979.

[39] Giles Hogben and Marnix Dekker. Smartphones: Information security risks, opportunities and recommendations for users. *European Network and Information Security Agency*, 710(01), 2010.

[40] Michael Howard and Steve Lipner. Inside the windows security push. *IEEE Security & Privacy*, 1(1):57–61, 2003.

[41] IDC. Smartphone os market share, 2015 q2. `http://www.idc.com/prodserv/smartphone-os-market-share.jsp`, 2016. Accessed: 2016-16-23.

[42] ISO/IEC_JTC1/SC27. Iso/iec 15408-1/2/3:2005 - information technology — security techniques — evaluation criteria for it security, iso/iec 15408:2005 (common criteria v3.0), 2005.

[43] W. Jackson. Under attack common criteria has loads of critics, but is it getting a bum rap?, 2007.

[44] Woongryul Jeon, Jeeyeon Kim, Youngsook Lee, and Dongho Won. A practical analysis of smartphone security. In *Proceedings of the 2011 International Conference on Human Interface and the Management of Information - Volume Part I*, HI'11, pages 311–320, Berlin, Heidelberg, 2011. Springer-Verlag.

[45] Karama Kanoun, Yves Crouzet, Ali Kalakech, and Ana-Elena Rugina. Windows and linux robustness benchmarks with respect to application erroneous behavior. *Dependability Benchmarking for Computer Systems*, 72:227, 2008.

[46] Karama Kanoun, Henrique Madeira, and Jean Arlat. A framework for dependability benchmarking. In *Proceedings of the 2002 Workshop on Dependability Benchmarking*, 2002.

[47] Karama Kanoun and Lisa Spainhower. *Dependability benchmarking for computer systems*, volume 72. John Wiley & Sons, 2008.

[48] Bilge Karabacak and Ibrahim Sogukpinar. Isram: information security risk analysis method. *Computers & Security*, 24(2):147 – 159, 2005.

[49] Yi kun Zhang, Su yang Jiang, Ying an Cui, Yi kun Zhang, and Hui Xia. A qualitative and quantitative risk assessment method in software security. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, volume 1, pages V1–534–V1–539, Aug 2010.

[50] M. La Polla, F. Martinelli, and D. Sgandurra. A survey on security for mobile devices. *Communications Surveys Tutorials, IEEE*, 15(1):446–471, 2013.

[51] Jean-Claude Laprie. Dependability: Basic concepts and terminology. In *Dependability: Basic Concepts and Terminology*, pages 3–245. Springer, 1992.

[52] Jean-Claude Laprie. Dependable computing: Concepts, limits, challenges. In *Special Issue of the 25th International Symposium On Fault-Tolerant Computing*, pages 42–54, 1995.

[53] Lookout. Phone theft in america. `https://www.lookout.com/resources/reports/phone-theft-in-america`, 2015.

[54] P.K. Manadhata and J.M. Wing. An attack surface metric. *Software Engineering, IEEE Transactions on*, 37(3):371–386, May 2011.

[55] Gary McGraw and Greg Hoglund. *Exploiting Software: How to Break Code*. Person, 2004.

[56] Daniel Mellado, Eduardo Fernández-Medina, and Mario Piattini. A common criteria based security requirements engineering process for the development of secure information systems. *Computer Standards & Interfaces*, 29(2):244 – 253, 2007.

[57] N. Mendes, J. Duraes, and H. Madeira. Benchmarking the security of web serving systems based on known vulnerabilities. In *Dependable Computing (LADC), 2011 5th Latin-American Symposium on*, pages 55–64, April 2011.

[58] Patricia Mayer Milligan and Donna Hutcheson. Business risks and security assessment for mobile devices. In *Proceedings of the 8th Conference on 8th WSEAS Int. Conference on Mathematics and Computers in Business and Economics - Volume 8*, MCBE'07, pages 189–193, Stevens Point, Wisconsin, USA, 2007. World Scientific and Engineering Academy and Society (WSEAS).

[59] Rene Millman. Surge in byod sees 7/10 employees using their own devices. `itpro.co.uk/mobile/19944/surge-byod-sees-710-employees-using-their-own-devices`, 2013. Accessed: 2015-12-02.

[60] Mitre. Cve - common vulnerabilities and exposures (cve). `https://cve.mitre.org/`, 2016. Accessed: 2016-02-10.

[61] Collin Richard Mulliner. Security of smart phones. Master's thesis, University of California, Santa Barbara, 2006.

[62] Alexios Mylonas, Stelios Dritsas, Bill Tsoumas, and Dimitris Gritzalis. Smartphone security evaluation the malware attack case. In *Security and Cryptography (SECRYPT), 2011 Proceedings of the International Conference on*, pages 25–36, July 2011.

[63] Alexios Mylonas, Dimitris Gritzalis, Bill Tsoumas, and Theodore Apostolopoulos. A qualitative metrics vector for the awareness of smartphone security users. In *Trust, Privacy, and Security in Digital Business*, pages 173–184. Springer, 2013.

[64] Alexios Mylonas, Anastasia Kastania, and Dimitris Gritzalis. Delegate the smartphone user? security awareness in smartphone platforms. *Computers & Security*, 34(0):47 – 66, 2013.

[65] Emilio Tissato Nakamura and Paulo Lício de Geus. *Segurança de Redes em Ambientes Cooperativos*. Novatec, 2007.

[66] A.A. Neto and M. Vieira. Towards assessing the security of dbms configurations. In *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008*, pages 90–95, June 2008.

[67] AA Neto and M. Vieira. To benchmark or not to benchmark security: That is the question. In *Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on*, pages 182–187, June 2011.

[68] Stack Overflow. `stackoverflow.com`. Accessed: 2016-06-02.

[69] S. Panjwani, S. Tan, K.M. Jarrin, and Michel Cukier. An experimental evaluation to determine if port scans are precursors to an attack. In *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*, pages 602–611, June 2005.

[70] Shari Lawrence Pfleeger. Risky business: what we have yet to learn about risk management. *Journal of Systems and Software*, 53(3):265 – 273, 2000.

[71] M. Polte, J. Simsa, and G. Gibson. Comparing performance of solid state devices and mechanical disks. In *2008 3rd Petascale Data Storage Workshop*, pages 1–7, Nov 2008.

[72] Georgios Portokalidis, Philip Homburg, Kostas Anagnostakis, and Herbert Bos. Paranoid android: Versatile protection for smartphones. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, pages 347–356, New York, NY, USA, 2010. ACM.

[73] Consumer Reports. Smartphone thefts drop as kill switch usage grows but android users are still waiting for the technology, 2015. Accessed: 2016-05-30.

[74] Linda H. Rosenberg, Ruth Stapko, and Albert Gallo. Risk-based object oriented testing. In *In: Proceedings of the 24 th annual Software Engineering Workshop, NASA, Software Engineering Laboratory*, 1999.

[75] Seth Rosenblatt. Avast to go mobile, get vpn. `http://download.cnet.com/8301-2007_4-20074377-12/avast-to-go-mobile-get-vpn/`, 2011. Accessed: 2014-12-05.

[76] Thomas Veneziano Sébastien Bigaret, Patrick Meyer and Alexandru-Liviu Olteanu. The decision deck project. `http://www.decision-deck.org/project/`, 2013.

[77] A Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, and C. Glezer. Google android: A comprehensive security assessment. *Security Privacy, IEEE*, 8(2):35–44, March 2010.

[78] Asaf Shabtai, Yuval Fledel, Uri Kanonov, Yuval Elovici, and Shlomi Dolev. Google android: A state-of-the-art review of security mechanisms. *CoRR*, abs/0912.5101, 2009.

[79] Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer, and Yael Weiss. "andromaly": a behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems*, 38(1):161–190, 2012.

[80] Inc Snapchat. Snapchat. `https://www.snapchat.com`, 2014. Accessed: 2014-12-06.

[81] Filomena Sousa. O que é "ser adulto": as práticas e representações sociais sobre o que é "ser adulto" na sociedade portuguesa. *Acolhendo a Alfabetização nos Países de Língua Portuguesa*, 1(2), 2007.

[82] E. Souza, C. Gusmão, and J. Venâncio. Risk-based testing: A case study. In *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, pages 1032–1037, April 2010.

[83] SPEC. Spec - standard performance evaluation corporation. `spec.org`, 2016. Accessed: 2016-06-23.

[84] William Stallings. *Cryptography and network security: principles and practices.* Pearson Education, 2015.

[85] Symantec. Internet security threat report. Technical report, Symantec, 2013. Accessed: 2014-12-04.

[86] Symantec. Internet security threat report. Technical report, Symantec, April 2016.

[87] Marianthi Theoharidou, Alexios Mylonas, and Dimitris Gritzalis. A risk assessment method for smartphones. In Dimitris Gritzalis, Steven Furnell, and Marianthi Theoharidou, editors, *Information Security and Privacy Research*, volume 376 of *IFIP Advances in Information and Communication Technology*, pages 443–456. Springer Berlin Heidelberg, 2012.

[88] TigerText. Tigertext - security text messaging app for the enterprise. `http://www.tigertext.com`, 2014. Accessed: 2014-12-06.

[89] TPC. Tpc - benchmarks. `tpc.org/information/benchmarks.asp`, 2016. Accessed: 2016-06-23.

[90] Daniel Vecchiato and Eliane Martins. Experience report: A field analysis of user-defined security configurations of android devices. In *Software Reliability Engineering (ISSRE), 2015 IEEE 26th International Symposium on*, pages 314–323, Nov 2015.

[91] Daniel Vecchiato and Eliane Martins. Benchmarking user-defined security configuration of mobile devices. In *Dependable Computing (LADC), 2016 7th Latin-American Symposium on*, Oct 2016.

[92] Daniel Vecchiato, Eliane Martins, and Marco Vieira. Security assessment - android apps on google play. `https://play.google.com/store/apps/details?id=br.unicamp.ic.lsd.securityassesment`, 2014.

[93] Daniel Vecchiato, Marco Vieira, and Eliane Martins. A security configuration assessment for android devices. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, pages 2299–2304, New York, NY, USA, 2015. ACM.

[94] Daniel Vecchiato, Marco Vieira, and Eliane Martins. The perils of android security configuration. *Computer*, 49(6):15–21, 2016.

[95] Daniel Vecchiato, Marco Vieira, and Eliane Martins. Risk assessment of user-defined security configurations for android devices. In *Software Reliability Engineering (ISSRE), 2016 IEEE 27th International Symposium on*, Oct 2016.

[96] Timothy Vidas, Daniel Votipka, and Nicolas Christin. All your droid are belong to us: A survey of current android attacks. In *WOOT*, pages 81–90, 2011.

[97] M. Vieira and H. Madeira. Detection of malicious transactions in dbms. In *Dependable Computing, 2005. Proceedings. 11th Pacific Rim International Symposium on*, pages 8 pp.–, Dec 2005.

[98] Marco Vieira. *Dependability benchmarking for transactional systems*. PhD thesis, University of Coimbra, Coimbra, Portugal, 2005.

[99] Marco Vieira and Nuno Antunes. Introduction to software security concepts. In Domenico Cotroneo, editor, *Innovative Technologies for Dependable OTS-Based Critical Systems*, pages 29–38. Springer Milan, 2013.

[100] Marco Vieira and Henrique Madeira. A dependability benchmark for oltp application environments. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 742–753. VLDB Endowment, 2003.

[101] Marco Vieira and Henrique Madeira. Towards a security benchmark for database management systems. In *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*, pages 592–601. IEEE, 2005.

[102] Marco Vieira, Henrique Madeira, Kai Sachs, and Samuel Kounev. Resilience benchmarking. In Katinka Wolter, Alberto Avritzer, Marco Vieira, and Aad van Moorsel, editors, *Resilience Assessment and Evaluation of Computing Systems*, pages 283–301. Springer Berlin Heidelberg, 2012.

[103] Min Wu, Robert C. Miller, and Simson L. Garfinkel. Do security toolbars actually prevent phishing attacks? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 601–610, New York, NY, USA, 2006. ACM.

[104] xdadevelopers. `forum.xda-developers.com`. Accessed: 2016-06-02.

[105] Yajin Zhou and Xuxian Jiang. Dissecting android malware: Characterization and evolution. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 95–109, May 2012.

[106] Yajin Zhou, Zhi Wang, Wu Zhou, and Xuxian Jiang. Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets. In *NDSS*, volume 25, pages 50–52, 2012.

Appendix A

Security configuration values

Table A.1: Template values of security concerns from CIS security benchmarks.

ID	SECURITY CONCERNS	VALUE
APPLICATION		
A1*	Enable automatic app updates	Enabled
BROWSER		
B1	Disable JavaScript	Disabled
B2*	Disable form auto-fill	Disabled
B3	Disable cookies	Disabled
B4	Enable block pop-ups	Enabled
B5*	Disable plug-ins	Disabled
B6*	Disable remember passwords	Disabled
B7*	Enable basic SSL checks for websites	Enabled
B8*	Turn on private browsing when needed	Enabled
CONTENT		
C1*	Disable SMS preview when the device is locked	Disabled
C2*	Disable apps views when the device is locked	Disabled
C3	Limit the number of text messages	20 or less messages saved
C4	Limit the number of multimedia messages	20 or less messages saved
NETWORK		
N1	Remove Wi-Fi entries	WiFi entries must be equals to 0 (zero)
N2	Disable network notification	Disabled
N3	Disable Wi-Fi when not needed	Disabled
N4	Disable bluetooth when not needed	Disabled
N5*	Disable auto-join for all Wi-Fi networks	Disabled
N6	Enable airplane mode	Enabled
N7	Turn off personal hotspot when not needed	Disabled
N8*	Turn off VPN when not needed	Disabled
N9*	Turn off AirDrop discoverability	Disabled
PASSWORD		
P1	Enable password	Enabled
P2	Require alphanumeric value	Enable alpha-numeric device password
P3*	Disable passcode unlock for fingerprints	Disabled
P4	Enable SIM card lock	Enabled
P5	Do not make passwords visible	Passwords must not appear as plain text
P6	Erase data upon excessive password failures	Erase data after 4 (four) failed password attempts
P7	Set a maximum password age	The recommended state for this setting is 90 days or less
P8	Set number of password history	24 or more passwords must be remembered
P9	Insert minimum passcode length	The recommended setting is 5 or more characters
P10	Set minimum number of complex characters	The recommended setting is set to 1 or more characters
P11	Set timeout minutes for sleep	Set a timeout of 90 seconds or less
PERMISSION		
PE1	Enable encrypt phone	Enabled
PE2*	Encrypt credentials storage	Encrypted
SYSTEM		
S1	Update firmware to latest version	Firmware updated
S2*	Disable access to control center on lock screen	Disabled
S3*	Erase all data before discard the device	Perform factory reset before discard the device
S4	Disable developer options	Disabled
S5	Disable unknown sources	Disabled
S6	Disable location services when not needed	Disabled
S7*	Enable "find my device" feature	Enabled
S8*	Set security to disallow profile removal	Profile removal must be disallowed
S9	Disable mock location	Disabled