



FFI-RAPPORT

16/00319

Modern mobile platforms from a security perspective

—
Federico Mancini

Modern mobile platforms from a security perspective

Federico Mancini

Keywords

Mobiltelefon
Datasikkerhet
Android
Risikovurdering
Informasjonssikkerhet

FFI-rapport:

FFI-RAPPORT 16/00319

Prosjektnummer

1294

ISBN

P: 978-82-464-2752-2

E: 978-82-464-2753-9

Approved by

Nils A. Nordbotten, *Research Manager*

Anders Eggen, *Director*

Summary

Commercial mobile technology has transformed the way we produce and consume information. Smart devices like phones, tablets, watches and even TVs, are all the time inter-connected through networks. These devices are packed with sensors and apps that allow us to easily collect and share instantly all types of data. Not surprisingly, many have realized that this technology could bring important innovations also in a military setting, and various projects have been started to explore the potential applications. Being able to report, aggregate, share and visualize important information in real-time just by downloading an app, is undoubtedly an attractive idea. Furthermore, the current interest in adopting LTE for military communications would make the case for using smartphones even more pressing.

Security is often presented as a main obstacle because commercial products cannot meet strict military security requirements without some additional hardening. Defining how this should be done is not trivial, especially if the same product is to be used in a variety of situations. The result is either that security is left as an after-thought yielding insecure products, or that commercial technology is advised against to be on the safe side. The reality is that smart devices do offer a wide range of security mechanisms, but the protection they can provide depends heavily the way they are used. That is why one should rather assess whether the operative effect gained by using them outweighs the potential risks on a case-by-case basis, and develop solutions that are secure enough for the specific situation. This kind of risk analysis should be based on a clear understanding of which assets (information) are to be protected and for how long, and the consequences of failing to do so. Once this is established, it is possible to determine whether the security mechanisms available can provide adequate protection or not, possibly by employing additional mitigation strategies.

With this report we want to place commercial smart mobile technology in a clearer security perspective and understand which threats they are best suited to protect against. First we give an extensive overview of the security models of the most popular mobile platforms on the market, namely Android, iOS, Windows and BlackBerry, and the security mechanisms they implement. We then review publicly available reports, statistics and documentation that show how effective these mechanisms are in a consumer market setting. Despite alarming reports of newly discovered vulnerabilities and malicious applications, it is in fact only a very small percentage of active devices that are affected. In many cases, the users are to blame because they intentionally deactivated or bypassed the security features that could otherwise protect them. Naturally, there are some threats that these mobile devices still cannot mitigate, but significant security is in place and is improving continuously. Efforts to create dedicated devices with military grade security based on commercial mobile platforms are also ongoing, showing that commercial grade security can provide acceptable protection also in a military environment thanks to the stricter control on both equipment and users. Dedicated management tools, careful data management and additional security technology like smartcards could further reduce the risk of compromise and thus expand the range of scenarios where this technology can be adopted and make a substantial difference.

Sammendrag

Kommersiell mobilteknologi har endret måten vi produserer og bruker informasjon på. Smarte enheter som telefoner, nettbrett, klokker og til og med TV er hele tiden sammenkoblet gjennom mobil- eller internett. Enhetene er fullpakket med sensorer og apper som lett lar oss samle og dele alle mulige typer data i sanntid med hvem vi måtte ønske. Ikke overraskende er det flere som mener at denne teknologien også kan ha spennende militære anvendelser, og det er allerede i gang mange prosjekter som utforsker forskjellige muligheter. Å kunne rapportere, sammenstille, dele og visualisere viktig informasjon i sanntid bare ved å laste ned en app, er utvilsomt en attraktiv idé. Til og med LTE, den kommunikasjonsstandarden som i dag brukes av de fleste smarttelefonene, er under vurdering for mulig militær bruk.

Sikkerheten er imidlertid ofte sett på som en potensiell hindring. Problemet er at militære sikkerhetskrav er så strenge at kommersielle produkter ikke alltid kan møte dem uten noe ekstra tilpasning. Dette kan være vanskelig å få til i praksis, spesielt hvis samme produkt skal være sikkert uansett situasjon. Konsekvensen er enten at sikkerhetsutfordringer bevisst ignoreres i håp om at de vil kunne løses i etterkant, eller at all bruk av kommersielle produkter frarådes for å være på den sikre siden. Realiteten er at smarte enheter tilbyr et bredt spekter av sikkerhetsmekanismer, men beskyttelsen de kan gi er svært avhengig av måten de brukes på. Man bør derfor vurdere om den operative effekten som oppnås i et konkret scenario er større enn den potensielle risikoen, og utvikle tilpassede løsninger som er sikre nok for den situasjonen. Denne typen risikoanalyse bør være basert på en klar forståelse av hvilken informasjon som skal beskyttes, hvor lenge den skal beskyttes og konsekvensene av kompromittering. Når dette er klargjort, vil det være mulig å finne ut om de tilgjengelige sikkerhetsmekanismene kan gi tilstrekkelig beskyttelse eller ikke, muligens ved å igangsette flere risikoreduserende tiltak.

Denne rapporten setter kommersiell mobilteknologi inn i et sikkerhetsperspektiv, og gjør det lettere å vurdere i hvilke situasjoner slik teknologi kan brukes. Først gir vi en omfattende oversikt over sikkerhetsmodellene til de mest populære mobile plattformene på markedet, nemlig Android, iOS, Windows og BlackBerry, og en oversikt over truslene de er tenkt å beskytte mot. Deretter gjennomgår vi offentlig tilgjengelige rapporter, statistikker og dokumentasjon som viser hvor effektive disse mekanismene er i et forbrukermarkedsperspektiv. Til tross for alarmerende rapporter om nylig oppdagede sårbarheter og ondsinnede apper, viser det seg imidlertid at det er en svært liten prosentandel av aktive enheter som faktisk er berørt. I mange tilfeller er det brukernes egen skyld fordi de bevisst deaktiverer eller omgår sikkerhetsfunksjonene som ellers kunne ha beskyttet dem. Naturligvis er det noen trusler som disse mobile enheter fortsatt ikke kan håndtere, men mye av sikkerheten er allerede på plass og blir stadig bedre. Forsknings- og industrimiljøer jobber også intenst med å utvikle enheter for militære formål basert på kommersiell teknologi. Administrasjonsverktøy, tilpasset datahåndtering og annen sikker teknologi som smartkort kan redusere risikoen for kompromittering ytterligere. Dermed kan denne teknologien tas i bruk i et større utvalg av scenarioer og gjøre en betydelig forskjell.

Content

1	Introduction	7
1.1	Context	8
1.2	Methodology	9
1.3	Scope	9
1.4	Report overview	10
2	Mobile security	10
2.1	Modern mobile platform security	11
2.2	Mobile platforms security models and features	18
2.2.1	ARM security	18
2.2.2	Android security	20
2.2.3	iOS security	23
2.2.4	Blackberry 10 security	25
2.2.5	Windows 10 Mobile security	27
2.3	Discussion	28
3	Security in practice	30
3.1	Threats, vulnerabilities and related concepts	31
3.2	Overview of security reports and statistics	34
3.2.1	Cross-platform vulnerabilities	35
3.2.2	Android	35
3.2.3	iOS	38
3.2.4	BlackBerry 10 and Windows 10 Mobile	39
3.3	Discussion	39
4	Possible approaches to enhance security	41
4.1	Ownership models supported by COTS device	42
4.1.1	User ownership	42
4.1.2	BYOD: shared ownership	42
4.1.3	Enterprise ownership	43
4.1.4	Management solutions	43
4.2	Dedicated solutions	44
4.2.1	Commercial (based) products	44

4.2.2	Dedicated proprietary solutions	46
4.2.3	Military research projects	46
4.3	Discussion	47
5	Risk mitigations and challenges – scenario analysis	51
5.1	The Common Operational Picture scenario	51
5.1.1	Initial risk assessment	52
5.1.2	Local aggregation and cryptographic material	52
5.1.3	Reliability	53
5.1.4	Classified server	53
5.1.5	Information sharing	54
5.1.6	Off-line	54
5.1.7	Civilian-Military cooperation	54
5.2	Discussion	55
6	Conclusions	55
	Appendix	58
	References	63

1 Introduction

The reason why commercial off the shelf (COTS) products have long been proposed as an alternative or complementation to especially developed military products is mostly because of the potential cost reduction, the quick adoption of new technology and the readily available products [1]. The main drawbacks are usually their inability to comply with the high military requirements for security and robustness and their lack of compatibility with military standards. In addition, commercial products are usually not made to last particularly long and especially IT products are continuously patched and updated during their life time. This can lead to problems like obsolescence, which is the lack of replacement parts or updates that renders relatively new products quickly obsolete so that they must be replaced or disposed of. A famous case is the super-cluster the US Air Force built out of PlayStations 3, but where a firmware update made it impossible to replace broken units with new PlayStations as it was now impossible to install Linux on them¹.

Commercial “smart” technology, and especially its mobile incarnations such as smartphones, tablets or even smartwatches, is making the case for adopting COTS products as current as ever. These devices changed the way we produce and consume information. They are with us all the time, are interconnected and packed with sensors that can continuously collect data about our environment, our movements or our preferences and present us with personalized information based on what we might need in that particular situation. Thanks to intuitive interfaces and apps for any thinkable purpose, we can also easily generate and share information instantly with whomever we like. Not surprisingly, many have already realized that this technology can have many innovative applications also in a military setting, despite the possible pitfalls mentioned above. Related activities are already exploring the use of apps for improving reporting [2], situational awareness [3], information sharing in crisis situations [4] and command and control information systems (C2IS) [5]. The proposal of using the Long Term Evolution (LTE) standard for the Norwegian Defence [6] also strengthens the case for adopting commercial smartphones that natively support it.

However, simply integrating commercial mobile devices into a military information system is no trivial task, and security is one of the main reasons. On one hand, there are many that are enthusiastic about what can be achieved with this technology and that focus on developing new apps, frameworks and interfaces tailored for military use, but intentionally leave security as an after-thought. This often results in insecure products that cannot be deployed unless they are completely redesigned from scratch with security in mind. On the other hand, we have military systems that can protect classified information mostly because they are kept isolated from other potentially harmful systems by an air-gap approach. This means that they are not designed to handle secure information exchange with other systems which cannot guarantee the same level of assurance or control, even though the information itself is not highly classified. Thus, commercial smartphones and tablets are often not considered secure enough to access these

¹ <http://arstechnica.com/gaming/2010/05/how-removing-ps3-linux-hurts-the-air-force/>

military systems even to retrieve or report unclassified information, because of the lack of secure information exchange solutions across different security domains.

However, the aim of this report is not to solve the above mentioned problems, but rather to place modern mobile platforms in a clearer security perspective, so that those who may be interested in adopting them for tasks or projects where security is critical, can get a better understanding of what protection they can expect, what solutions are available and what challenges they may encounter in actual deployment. Increased awareness around what these devices can realistically offer in terms of security can then help to identify the situations where they can provide an adequate level of protection and to design solutions that are usable in practice.

1.1 Context

The common claim that a commercial mobile device is generally “not secure enough” is misleading. There will always be some risks associated with the adoption of any technology and it is important to correctly understand what these risks are and assess whether they are acceptable² on a case-by-case basis. The concept that the protection required for some given information is not static, but can change with time and circumstances, should also be central in this assessment process. The current military approach to information security, however, is not flexible enough to account for this kind of dynamism.

The first problem is that when information is to be exchanged between systems, the security requirements are to a large extent based on the classification of the systems rather than that of the information. So, unclassified information may not be accessed by an unclassified device if it is stored on a classified server for fear that other classified information may leak out. In a similar way, an unclassified device may often not even send information to a classified server in the chance that malicious code could sneak in and compromise it. This effectively reduces the effect of adopting commercial devices even in situations where no sensitive information is involved, but where they have to operate within a military infrastructure.

The other problem is that information is classified statically, often based on long-term confidentiality requirements, and in order to make it available to lower classified systems or users with lower clearance, it must be manually declassified. This is a time-consuming process and does not account for many possible situations where other requirements like integrity and availability may weigh more than confidentiality, or where confidentiality protection does not need to be long-lasting. For instance, smartphones may not be trusted to protect classified information due to their mobile nature that makes them particularly vulnerable to physical attacks, but if the information could be considered classified only for a short period of time for which we can guarantee reasonable protection, then the risk of using smartphones could be acceptable given some additional mitigations. In other situations, one may decide that

² «Acceptable risk» is defined as : «A risk that is understood and tolerated by a system's user, operator, owner, or accreditor, usually because the cost or difficulty of implementing an effective countermeasure for the associated vulnerability exceeds the expectation of loss” in RFC4949.

confidentiality is not so important as long as we can trust the information provenance (trusted sources), so that integrity requirements should be used to assess whether a specific device can be used, rather than its capability to keep information secret. Similarly, connectivity and interoperability requirements could be prioritized over confidentiality in scenarios where cooperation is the key to success rather than secrecy.

So, it might indeed be difficult to define commercial mobile devices as “secure” in the current context. However, the definition of a more flexible security concept based on a dynamic risk assessment rather than static security classifications has already been envisioned in the context of a Network Based Defence [7], so that even commercial devices may effectively be considered secure enough for some situations. Activities that deal with the specific problems mentioned above are also ongoing [8]. Thus, what can be considered secure will most likely also change, and this report wants to provide the starting point to evaluate commercial mobile devices in this new context.

1.2 Methodology

This report surveys a selection of commercial mobile technologies based on available open sources. The goal is to present an organized overview of the security mechanisms they implement, a discussion of what they are meant to protect and how well they work on a large scale in the consumer market. We estimate their general effectiveness by reviewing publicly available security reports. These are mostly compiled by antivirus companies and other third parties that collect and analyze data about the known vulnerabilities and infections present on the devices that run their clients. Based on this analysis we can gain an idea of what constitutes a threat for these devices and why security may have failed to defend against it. We then use this information to evaluate additional solutions specifically developed to enhance mobile platform security, both for commercial and military purposes. Finally we put everything together by describing a possible scenario and discuss practical challenges one may encounter and possible mitigations when deploying a mobile solution in a military setting.

1.3 Scope

There are various limitations that should be considered when reading this report. First of all, we consider mainly platform security and only touch upon application and infrastructure security. Application security is very important, but it would require a separate report and is something over which we have much greater control than platform security. The infrastructure is also essential to a complete secure solution, but it is being looked at in other upcoming reports. Secondly, the overview of security mechanisms we present here is not exhaustive. We selected the four most popular commercial mobile platforms, which together have a market share of over 99%, and described only the most relevant of their security mechanisms in order to give an idea of what kind of security is offered in general. We do not have as a goal to write a detailed technical reference. Besides, many implementation details are not made public, so there may be additional mechanisms that are not documented at all, or that are implemented in different ways

on different devices. The imbalance in the publicly available information for the different platforms we consider is also reflected in the level of detail in this report. Open platforms are naturally discussed more thoroughly than closed ones.

The estimate of the infection rates of active devices, and therefore of their ability to defend against various threats is also based on known attacks and vulnerabilities and their detection is dependent on information reported by the device itself. Devices that for instance are kept off-line, those that do not share information with antivirus companies or similar third parties, or victims of new unknown attacks, are therefore not included in these estimates.

Thus, the report should be used as an indication of what security can be achieved with commercial mobile phones given certain assumptions, but whether a given solution and device model is indeed secure enough for a specific purpose must be evaluated separately through some dedicated testing or certification process, and by requesting more detailed documentation from the manufacturer.

1.4 Report overview

The next chapter explores in detail the security models and mechanisms adopted by the most popular commercial mobile platforms, namely Android, iOS, Windows and Blackberry, in order to understand what kind of security they offer and for what purpose. Chapter 3 analyses how effective these mechanisms are in practice in a commercial setting based on public security reports and statistics. What emerges is that the number of actual compromised devices is surprisingly small in percentage, in contrast to the widespread opinion that mobile devices are relatively insecure and easy to compromise. In Chapter 4 we compare different Bring Your Own Device (BYOD) approaches and other products and projects aiming at improving mobile device security for classified and tactical use. Chapter 5 presents a possible scenario where commercial mobile devices are to be used and integrated within a military infrastructure and analyzes some of the practical challenges users and staff might meet. Possible mitigations are then discussed based on our previous work on the subject [2]. Finally, in Chapter 6 we give some conclusions.

2 Mobile security

Modern mobile devices like smartphones and tablet are used for more and more security-sensitive tasks like: mobile payments; secure authentication; remote control of house alarms, cars, and even drones; enjoying premium services and paid content; storing privacy-sensitive information; accessing company networks; and so on. This means that there are many actors that need to trust these devices with protecting their assets, and if mobile platform providers and device manufacturers want to appeal to a wide audience and gain market share, they need to

provide sufficient and convincing security on which to build all these services. Lack of adequate security has long been a problem, but it seems that recently much more focus has been dedicated to this issue and systems have been redesigned to fill the existing gap, so that even mid-range devices can provide sufficient protection for most daily tasks. In this chapter we give first an introduction to the most common security mechanisms found today in commercial mobile devices, and explain what they are intended to protect and at which level of the platform they operate. Afterwards, we present in more detail the security models of the most widespread mobile platforms on the market, namely Android, iOS, BlackBerry and Windows, which together account for more than 99% of the device platforms present on the global market as shown in Figure 2.1.

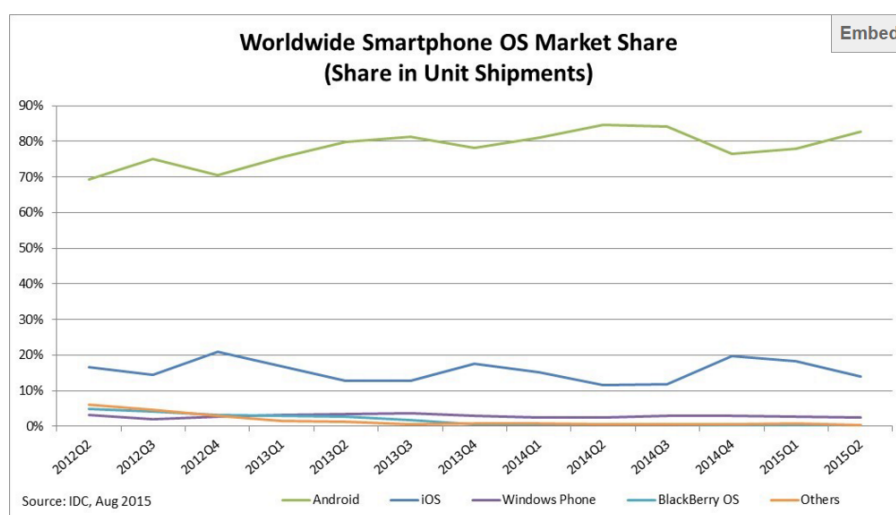


Figure 2.1 Data about global mobile platform market share from IDC³.

2.1 Modern mobile platform security

In order to understand the security of a mobile platform, we need first to describe the components of such platform, the actors involved and the assets they intend to protect. A generic mobile device architecture is shown in Figure 2.2.

A mobile device consists of some basic hardware not too unlike that of a laptop. We have a processor, the volatile memory or RAM, permanent storage, a display, a microphone, speakers, some peripherals and a battery. Unlike other computing devices we find also various other sensors like GPS, accelerometer, temperature and light sensors and others. In the case of smartphones we also find a radio processor, or baseband, to handle the connection to a mobile

³ <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>

network. Finally, rather than a BIOS (Basic Input-Output System) we have one or more bootloaders that boot the software on the device. Assembling or even producing all these components is responsibility of a *manufacturer*.

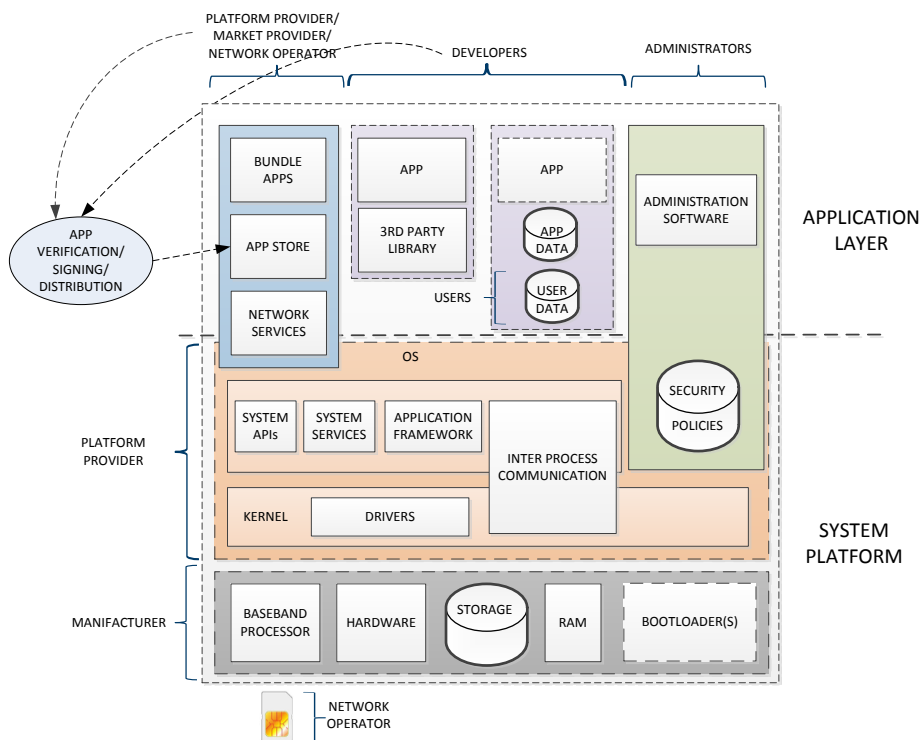


Figure 2.2 A generic mobile architecture where we roughly illustrate which components are under the control of which actors. We distinguish also between the system platform that includes hardware and operating system, and the application layer which includes the apps running on top of it and that usually are installed afterwards by users or administrators.

On top of the hardware we have an operating system (OS). Its main component is a kernel that takes care of the most critical functionalities like process scheduling, interface with the hardware, resource handling, and low level communication. At a higher level it provides interfaces to the application layer to communicate with the hardware and access various services and system libraries. This component is provided by the *platform provider*. Applications that run on top of the OS can be pre-installed or developed independently by third-party *developers* and made available for download by distributing them through *app-stores*. App-stores are run by *market providers* who can enforce various policies about what kind of application their app-store accepts and who can publish applications there. If the market provider coincides with the platform providers, tighter control on what can be installed on a specific platform can be enforced. However, it is usually the *user* who can decide what to install and what not, unless the device does not belong to the user. In this case an *administrator* can enforce a specific security policy about how the device should be used. Finally, in smartphone, the *mobile operator*

controls the secret parameters needed to connect and use the mobile network through a SIM (Subscriber Identity Module) card, and sometimes it can also operate as device administrator when it comes to which application can be installed or which networks can be used. Summarizing, we can identify seven main actors: Users; Manufacturers; Mobile Operators; Developers; Platform Providers; Marketplace Providers and Administrators. Each of them has control and responsibility for different parts of the mobile device and the infrastructure around it, and they have different assets they want to protect from different potential adversaries. Table 2.1, reported from [9], summarizes the relationships among these actors, the assets they wish to protect and their adversaries.

Actors	Incentives	Resources to protect	Primary adversary	Additional adversaries
<i>Users</i>	Preserve privacy, use device freely	Private user data	Remote attacker	Attacker with temporary physical access
<i>Manufacturers</i>	Business model, regulatory services	Device identifiers, configuration parameters, platform version	User	External attacker
<i>Mobile operators</i>	Subscriber contract enforcement	Usage of subsidized devices, mobile network resources	User	External attacker
<i>Developers</i>	Mobile service protection	Application data and code	Remote attacker	User
<i>Platform providers</i>	Business model	Platform functionality	Malicious/sloppy developer	User
<i>Marketplace providers</i>	Marketplace popularity and reputation	Distributed applications	Malicious/sloppy developer	User
<i>Administrators</i>	Company business model	Company confidential data	Remote attacker	Attacker with temporary physical access

Table 2.1 Summary of actors who have some asset associated to the mobile platform they want to protect [9].

Based on this table we can identify some basic security mechanisms that we can expect to find in a modern mobile platform, which are summarized in Table 2.2.

SECURITY MECHANISMS AT PLATFORM LEVEL			
HARDWARE	Secure Boot	Hardware-based security	SIM card
OPERATING SYSTEM	Sandboxing and isolation	Data-at-rest protection	Data-in-transit protection
	Exploit mitigations	Crypto	Authentication
INFRASTRUCTURE	Secure application provisioning	Security updates	Security management

Table 2.2 Summary of the main security mechanisms expected in a modern mobile platform, organized per level.

Let us now go through Figure 2.2 based on the protection needs listed in order to understand how we came to Table 2.2. First of all, we separate the mobile architecture into two parts: the mobile platform and the application layer. We can then distinguish between *platform security* and *application security*. Platform security concerns all security mechanisms that come with the device, including the hardware, OS, mobile operator and pre-installed services based on some existing infrastructure. In other words, what we get when buying an off-the-shelf device. Application security is the security implemented in each app we install on the device, and builds mainly upon the platform security. Although application security is also very important, here we focus mostly on platform security both because it would be too large of a topic to tackle both in one report, and because platform security is the building stone on which application security depends on. If the platform is compromised, no security built in the application layer can be completely trusted. We are going to summarize the typical application security concerns at the end of this section anyway, but we are first going through the actors in Table 2.1 with focus on the platform.

Manufacturers want to protect platform identifiers, hardware firmware like boot-loaders and baseband processor, and other configuration parameters. In other words they want to protect the *platform integrity* and make sure that everything is in order every time the system is booted. The platform provider, or better the operating system, needs to make sure that malicious developers, users and external attackers, cannot easily subvert the platform and compromise or steal sensitive user data, cryptographic keys, credentials and copyrighted material, or install spyware that can record transactions or conversations taking place on the device. This means that it must provide a series of security mechanisms to prevent for instance: applications from reading each other memory; users from misusing or redistributing paid services and material; and malicious code from exploiting potential vulnerabilities and gain system privileges. This should happen both in a preventive manner, by allowing only trusted applications to be installed and giving them only the permissions they need, and at run-time by making sure attacks to or from these apps are either prevented or mitigated. In order to establish trust in the applications that are downloaded and installed on the device, someone must verify their genuineness before they are made available to the end-users. Usually it is the marketplace providers that should verify both the identity of the developers and that the applications are developed after certain quality and security standards. Users are also critical to security, as they can often take actions that override or deactivate the mechanisms that are there to protect them. Based on these security needs, we summarized some security mechanisms in Table 2.2, which we now explain more in detail:

- *Trusted or secure boot*: This technology is used to verify the integrity of the system components at boot time. It builds a transitive chain of trust starting from the first component run on the platform, which is assumed to be correct, non-bypassable and immutable. This component is called the *root of trust*, and in mobile platforms it is usually implemented in firmware as the first stage boot-loader. This root of trust can either “measure” or verify the signature of the next component, before letting it run on the system. If the verification is successful, the next component is run and iterates the process until the OS image, which is now guaranteed to be the genuine one, is loaded and takes control of the platform.
- *Sandboxing and isolation*: Protection of the applications running on top of the operating systems should be implemented and enforced at platform level. Sandboxing and isolation should guarantee that each application runs in a protected and isolated environment, so that the rest of the system can neither compromise or be compromised by it. Sandboxed applications should not have access to any other resources than the ones they need, and should not be able to directly access system resources or other apps memory. In addition, the inter-process communication between apps or processes should also be carefully designed and under the control of the operating system. Access control can be used to enforce a sandbox model.
- *Mitigation of exploits*: The platform should offer mitigation for possible exploitation of vulnerable apps on the system, so that malicious code loaded in memory cannot be executed so easily by the malicious app even though it managed to break out of the

sandbox. Memory pages marked as non-executable and random allocation of code in memory are examples of these mechanisms.

- *Secure application provisioning*: anything that can be installed on the platform should be verified to be free from malicious code and possibly come from a trusted source. In addition it may be installed only if it conforms to some given security policy. The process of verification can take place both on the platform itself, but also in other parts of the infrastructure before an application is even allowed to be made available for download. Signature-based verification of application binaries is the standard.
- *Protection of data at rest*: The platform should offer a secure storage mechanism to the applications running on it, in order to protect their data also at rest. Sensitive data can be personal data, identity credentials, or cryptographic keys and certificates. Full disk encryption and locations with restricted access rights are some of the common measures taken to achieve this.
- *Protection of data in transit*: Standard and well established cryptographic protocols should be offered by the platform to protect communication to and from the device. VPN, TLS and PKI certificate support are some examples.
- *Centralized and frequent security updates*: As new vulnerabilities in mobile platforms are continuously discovered, it is essential that an infrastructure is in place to provide the required security patches as soon as they are available and possibly without the need of user intervention or further adaptations.
- *Support for security management*: In order for the platform administrator to create and enforce the company security policy, the platform must offer an adequate interface to management software, so that policy enforcement can take place at system level. Native BYOD solutions are also becoming a standard in modern mobile devices.
- *Device lock*: some form for local authentication is needed to lock the device to others than the legitimate user. This can be in the form of a screen-lock, pin, password, smart-card, facial recognition or geo-fencing, to name a few.
- *Crypto libraries*: In addition to data protection at rest and in transit, in order for applications to implement their own layer of security, the platform should provide functional and certified cryptographic tools, so that encryption, signing, generation of random numbers and secure connections can be used without having to rely on third party libraries shipped with the application.
- *Mobile network parameters*: secret parameters needed to authenticate a subscriber to the mobile network are securely stored in the SIM card. These smart cards receive network information directly from the baseband processor, but can also use the operating system to provide some services to the user or collect user input, through the

(U)SIM Application Toolkit [10]. They can be bound to a specific device by an operator lock implemented usually in the firmware⁴.

- *Other hardware-based security mechanisms*: trusted boot, SIM cards and some exploit mitigation mechanisms are examples of hardware-based security mechanisms. Others can be cryptographic co-processors, access control to peripherals embedded in the processor itself rather than the OS, trusted execution environments, eFuses⁵ and so on.

To conclude this section and for completeness, we report the “The Open Web Application Security Project (OWASP) Mobile Top 10”⁶, which is a list of the most common security risks in mobile application development. In other words, security aspects developers tend to ignore or fail to properly implement, but which do not directly depend on platform security:

1. *Weak Server Side Controls*: This is actually pretty much anything that can go wrong, but that does not directly take place on the mobile device.
2. *Insecure Data Storage*: Most applications store sensitive data unencrypted on the device, or even worse on an external memory card. Even when encrypted, the key may not be strong enough or protected correctly.
3. *Insufficient Transport Layer Protection*: The connection to a server is not always cryptographically protected, and even when it is it may use old algorithms, weak keys or invalid certificates.
4. *Unintended Data Leakage*: This is often in the form of aggressive caching or logging of sensitive information, or clear text metadata sent to websites or third party server.
5. *Poor Authorization and Authentication*: Local authentication that is based on weak passwords; storing keys and passwords in clear text on the device; and assuming that because a user authenticated successfully locally, there is no need to authenticate also remotely on the server; are all examples of this problem.
6. *Broken Cryptography*: This includes third party crypto providers whose algorithm implementations have not been properly tested and certified; weak key management; custom cryptographic protocols; use of insecure or deprecated algorithms.
7. *Client Side Injection*: This concerns all possible types of code injections an attacker can perform when proper input validation is not implemented. For instance SQL injections, JavaScript injections and XML injections. These attacks are often executed via a web component.

⁴ <https://www.theiphonewiki.com/wiki/Unlock>

⁵ <https://www.samsungknox.com/en/blog/about-rooting-samsung-knox-enabled-devices-and-knox-warranty-void-bit>

⁶ https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks

-
-
8. *Security Decisions Via Untrusted Inputs*: Somewhat similar to the previous, it refers mostly to inter process communication and the lack of input validation.
 9. *Improper Session Handling*: This problem refers to the failure of properly invalidating a session, setting a reasonable time-out, rotating cookies and generating tokens in an unpredictable manner.
 10. *Lack of Binary Protections*: Refers to the possibility of easily reverse engineering and modifying application binaries in order to bypass security mechanisms.

We have now enough terminology and background to look at the specific security mechanisms implemented in the four mobile platforms we chose to analyze.

2.2 Mobile platforms security models and features

Although the term “platform” sometimes indicates only the OS, in this context we use it to intend both the hardware, the OS, and the infrastructure around it. The reason is that many platform vendors like Apple and BlackBerry control all aspects of the production chain of a mobile device, from hardware to apps, in such a way that it makes sense to see all these components as a whole. Others like Microsoft provide only the OS, but they still rely on some hardware features that are expected, if not required, to be present on the device on which they run. Finally, there is Google that provides a reference Android implementation (AOSP), but leaves carriers and manufacturers free to modify it as they wish in order to run it on their devices and with their additional software. Still, all the mobile platforms we present in this report run on a common hardware platform, namely ARM SoCs (System on Chip). Therefore, we start by briefly presenting the hardware security extensions that are built in ARM chipsets, so that it will be easier to understand how they are used by the different platform providers.

2.2.1 ARM security

Unlike Intel and AMD, ARM provides only the specifications for their chipsets, which are then manufactured by different companies like Qualcomm, Texas Instruments, Exynos, Broadcom, Apple, and others. However, most new ARM processors include some interesting security extensions, collectively called TrustZone. TrustZone is an implementation of what is known as a Trusted Execution Environment (TEE). In simple terms, it is a technology that allows virtualizing a secure processor on top of a normal one, and that integrates access control at hardware level. By setting a special bit called the NS (Non Secure) bit in the processor, one can run the same processor core in either a non-secure or a secure mode, or “world”, and at the same time make the memory and peripherals aware of which mode is currently active, so that they can grant or deny access to some pre-configured secret, devices or memory areas.

Using this technology, one can effectively run two different operating systems on the same device, where one runs in the “normal world” and does not have any access to the memory and dedicated devices of the other one, which runs in the so called “secure world”. However, there are different possible architectures that can be built on top of TrustZone, including simple APIs, secure services, or complete operating systems [11]. Possible secure services one can implement can be: secure boot, firmware TPM, crypto services, and isolated execution. More details can be found in a previous report [12]. This technology is not new, as it was first presented in 2004 [13], but it has been actively used to offer better security in mobile devices only in the last years. Some companies like Gemalto and Giesecke & Devrient developed also their own TrustZone-based solutions, but eventually joined in a common venture called Trustonic⁷. Standardization efforts have also been ongoing, and Global Platform has now released specifications for various aspects of mobile TEEs⁸. The most updated and complete open source implementation of a TrustZone-based TEE is now the Linaro OpenTEE⁹. Their solution architecture is reported in Figure 2.3. Another security feature of ARM processors is the XN (eXecute Never) or NX (No-Execute) bit, that allows the OS to mark memory page as non-executable, so that the code stored in them will never be run.

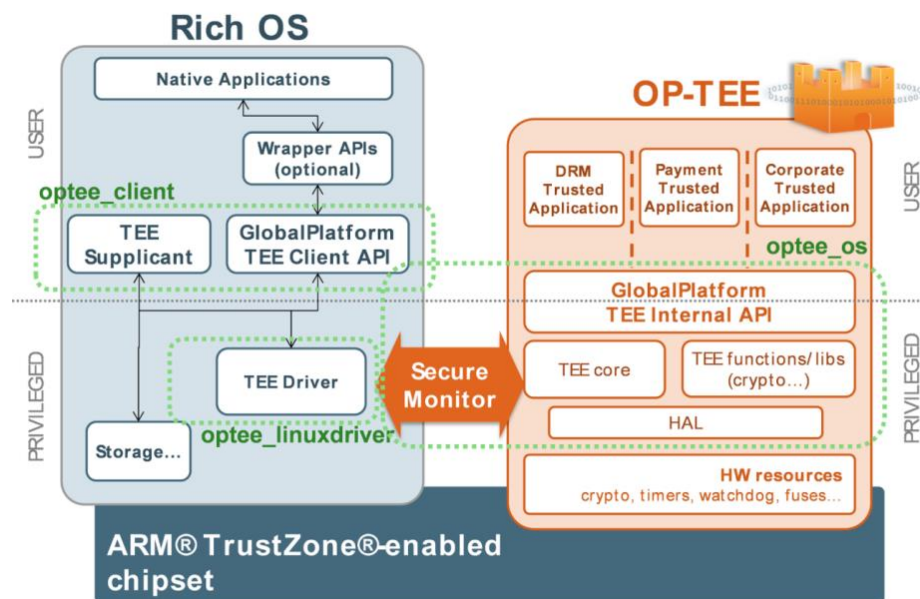


Figure 2.3 Linaro OpenTEE architecture⁶.

⁷ <http://www.arm.com/about/newsroom/arm-gemalto-giesecke-devrient-form-joint-venture-deliver-next-generation-security.php>

⁸ <https://www.globalplatform.org/specificationsdevice.asp>

⁹ <http://www.linaro.org/blog/core-dump/op-tee-open-source-security-mass-market/>

2.2.2 Android security

Android is the most popular OS around, running on more than 80% of the mobile phones shipped globally last year as shown in Figure 2.1 **Error! Reference source not found.** Android as a mobile operating system is maintained by Google, but it is installed on devices manufactured by many different companies like Samsung, LG, Motorola, Huawei, Sony and recently even Blackberry, just to name a few. This means that although the basic Android features are developed, maintained and documented by Google, different devices may ship with their own customized versions, possibly with additional modules and features and varying underlying hardware. This has of course an effect also on the security offered by each single device, although a vulnerability found in the original Android by Google would most likely affect all of them in some way. In addition, patches released by Google will not reach all Android devices simultaneously, because it is the single OEM's (Original Equipment Manufacturer) responsibility to provide patches for their own systems. The main components of the operating system are shown on Figure 2.4, taken from the official Android Open Source Project web-page. Security is built mostly in the kernel, which is based on the Linux kernel and from which it inherits the user-based permission system, the process isolation and secure Inter Process Communication (IPC). Sandboxing, various authentication methods, full disk encryption, verified boot, VPN and device administration capabilities are also standard security mechanisms that have long been part of the OS.

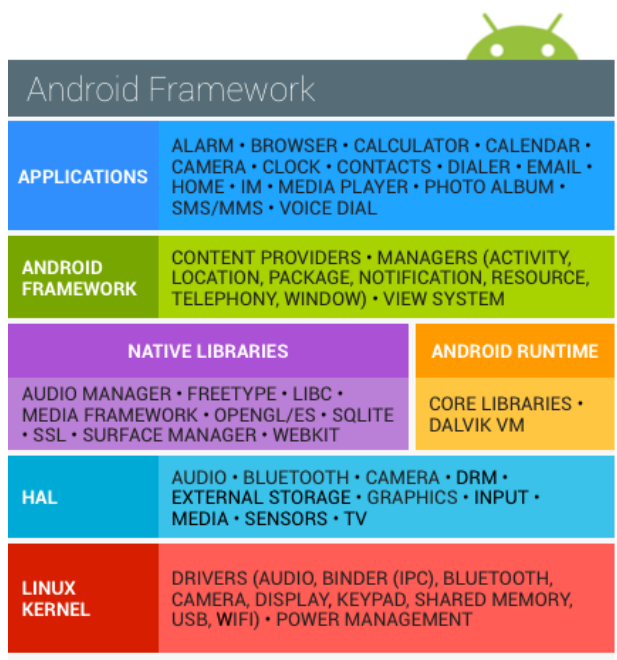


Figure 2.4 Android architecture ¹⁰

Recently, however, from Android Lollipop, some security features that require TrustZone support have also been introduced, together with other security enhancements. This is partly shown in Figure 2.5. The Keystore and Digital Rights Management (DRM) manager have been moved to the “secure world” so to be inaccessible to the normal OS and tamper-resistant. The actual services however, must be implemented or provided by the device manufacturer, while Android simply offers an interface to its apps for using these services if they are present. The verified boot mentioned earlier, assumes also some kind of hardware support, as it is dependent on the boot-loader verifying the kernel image with some preloaded key, in order to assess its integrity¹¹.

The NX bit discussed earlier is also actively used, and SELinux (Security Enhanced Linux) [14] has been integrated in Android to provide Mandatory Access Control (MAC). Unlike Discretionary Access Control (DAC), MAC can enforce more flexible and fine-grained security policies that can help to mitigate many types of attacks also after a vulnerability has been exploited. Android for Work that allows partitioning the device into a Work and a Personal space is built on these technologies [15].

A last fundamental difference between Android and other mobile platform is the App Store model. The official android store is called Google Play, and in order to publish third party apps on it, it is only necessary to pay a nominal fee at registration time for the developer¹², and sign the app binary with a self-signed certificate¹³. This gives little trust in the identity of the developer. In addition there does not seem to be any manual revision of the apps to guarantee their quality, but only an automatic checker called Google Bouncer¹⁴. While the effectiveness of this approach is still debated, it does give some security. The real problem is that users can decide to install also third party apps from unknown sources that give no guarantees whatsoever on the origin and genuineness of their apps. In this sense, what distinguishes Android from other platforms is its openness, which can be good for developers and users, but probably not so good for security. In order to mitigate this problem, Google introduced Verify Apps¹⁵, a service that scans third-party apps at installation time, warning the user of potential harmful applications. In addition, a service called SafetyNet continuously monitor the device and collects information about configuration, installed apps, network usage and more, in order to uncover possible compromise in the form of rooting, installed malware or in general CTS (Compatibility Test Suite) compatibility¹⁶. SafetyNet is also provided as a service for app developers that can use it to “attest” the status of the device before running security sensitive services. Its internal

¹⁰ <https://source.android.com/security/>

¹¹ <https://source.android.com/security/verifiedboot/verified-boot.html>

¹² <http://developer.android.com/distribute/googleplay/start.html>

¹³ <http://developer.android.com/tools/publishing/app-signing.html>

¹⁴ <http://googlemobile.blogspot.no/2012/02/android-and-security.html>

¹⁵ <http://officialandroid.blogspot.no/2014/04/expanding-googles-security-services-for.html>

¹⁶ <https://source.android.com/compatibility/cts/index.html>

functioning is not clear, but it appears that Google invested a lot of effort to provide a robust and secure service¹⁷.

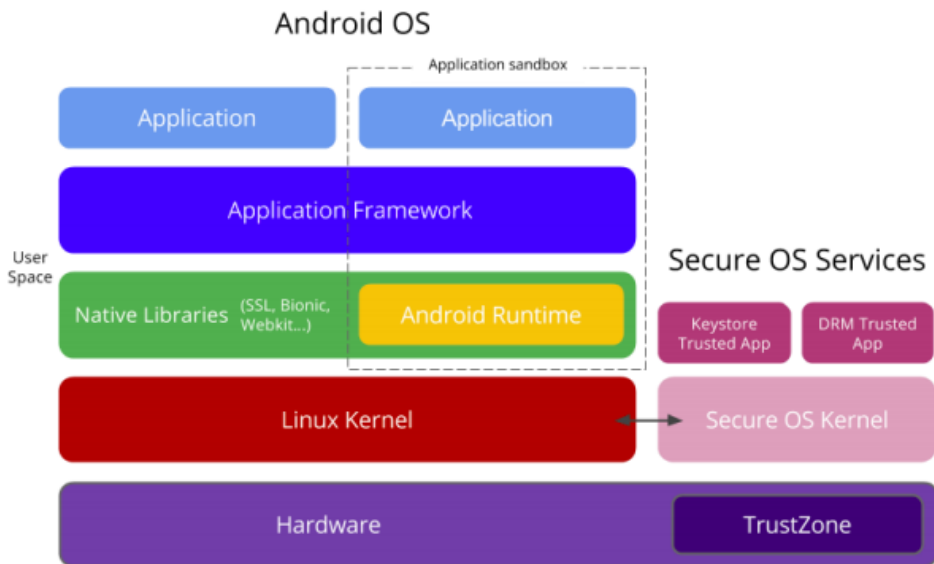


Figure 2.5 Extended security model of Android, including TrustZone support, from [15].

2.2.2.1 Samsung Knox

Since Android can freely be modified by OEMs, we have also other incarnations that provide extra security on top of what is offered out-of-the-box. The main example is Samsung Knox [16], a security enhanced version of Android developed by Samsung, which was used as the base to build the new security functionalities of Android Lollipop¹⁸, among which Android for Work. A detailed comparison between the two can be found in [17], but as Figure 2.6 shows, the main difference is the trusted computing approach they implemented in the pre-boot environment leveraging the TrustZone capabilities. This provides integrity protection both at boot and run-time, and even attestation capabilities (more on trusted computing in [12]). In addition, they offer reinforced work-space containers, called Knox workspaces. Still, the actual Knox environment, which offers a parallel environment with its own desktop, apps and services, appears to be nothing more than an app run on top of the common Android kernel rather than in TrustZone, so the “secure-world” is probably only used to run the pre-boot services rather than a dedicated secure OS. Besides the dedicated secure workspace, Knox secure APIs can also be used by other apps to provide additional security on Samsung devices in a transparent way for the user. Finally, customized secure operating systems based on Knox can also be developed and installed by Samsung so that they are locked to the specific device at manufacturing time and no rooting is necessary.

¹⁷ <https://koz.io/inside-safetynet/>

¹⁸ <https://www.samsungknox.com/en/androidworkwithknox>

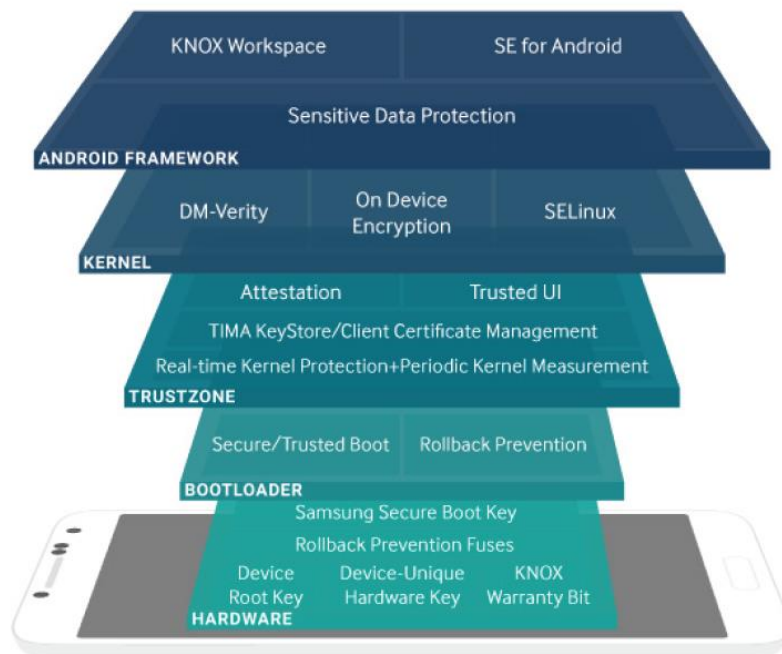


Figure 2.6 Samsung Knox architecture [16].

2.2.3 iOS security

Apple is the second largest actor on the mobile market, and it is one of the platform providers that has complete control over the whole production chain. Apple produces even its own processors, the ARM-based Apple A series. This means that it can tightly integrate hardware security with OS security, and provide a unified experience to users across different types of devices as well. App distribution is also centralized in the AppStore, and updates and security patches can be distributed directly to all Apple devices with no delays typical of the Android model.

Apple mobile devices rely heavily on hardware-assisted security and have even a dedicated cryptographic coprocessor called the “Secure Enclave”, which has a dedicated secure boot process in addition to the usual one for the rest of the system, and a secure element, as can be seen in Figure 2.7. In addition, each device has some unique cryptographic keys and certificates installed or generated directly in the processor at manufacturing time, which uniquely identify the device and cannot be extracted. These are used for many security critical tasks like encryption, integrity checking, secure boot and signature verification. The XN bit is also used to mark memory pages as non-executable.

The operating system, iOS, is derived from Darwin, an open source Unix system, and implements the protection at application level like secure installation, sandboxing and file encryption. Only signed apps can be installed on the device, and once installed they are sandboxed so that they can only access resources through standard services exposed by the

operating system. Most of the OS itself runs as an unprivileged user and the whole OS partition is mounted as read-only, so that a malicious app cannot try and modify it or escalate privileges [18]. Encryption, shown in Figure 2.8, is performed on a per-file basis, but the keys and the cryptographic algorithms are handled by the dedicated crypto-processor.

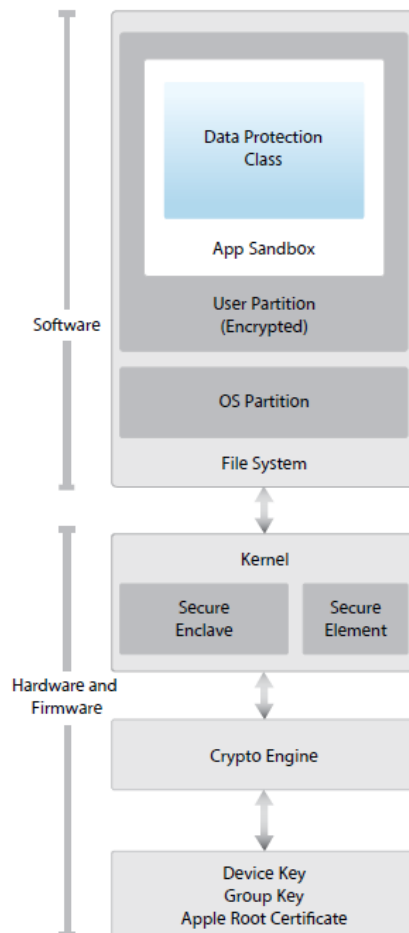


Figure 2.7 Apple mobile security architecture [18].

Apple devices support also separation between Personal and Work space, a wide range of cryptographic algorithms for encryption, VPN and network protection, and a proprietary mobile payment solution that leverages the secure element.

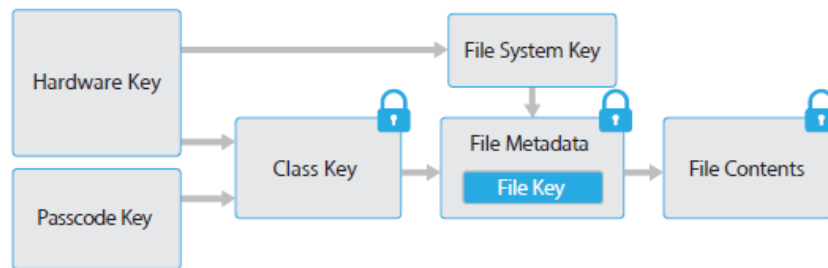


Figure 2.8 iOS encryption [18].

As mentioned at the beginning, the distribution of apps is centralized, and the review process before allowing an app to be published is quite thorough. Apple requires developers to register to their iOS Developer Program in order to issue a certificate that will be used to sign their apps. Developers are to be identified personally. Apps are reviewed by Apple also manually in order to ensure that they operate as described and do not contain obvious bugs or other problems. All approved apps are then signed by a certificate issued by Apple, so that they can be validated on the device by the Apple root certificate installed at manufacturing time. Signatures are also checked every time an app or some code is run [19]. The Apple store is the only way to install apps on a device, no third party stores are allowed. However, companies can obtain special certificates to create in-house apps that can be installed bypassing the appstore [18].

2.2.4 Blackberry 10 security

Blackberry (BB) is known for their focus on security and lately they completely redesigned their platform with BlackBerry 10 (BB10). Previously, they heavily relied on Java and their proprietary OS BBOS, but now that is gone, and a completely new operating system is at the base of the platform: the Neutrino QNX RTOS (Real Time OS) showed in Figure 2.9. This OS is certified to Common Criteria EAL (Evaluation Assurance Level) 4+ and is based on a microkernel that enforces strong isolation already at kernel level rather than application level¹⁹, so that file system, device drivers and network are not part of the kernel.

Security features built on this OS are similar to the ones we have seen for Android and iOS. Like Apple, BlackBerry also controls the whole manufacturing process, and installs hardware root of trust in form of keys directly in the processor. These keys are used to provide encryption also at file level, and to partition the device in Personal and Work space, mostly based on two different encryption keys [20]. This BYOD solution is called Balance. Actual sandboxing seems to be based only on user and group IDs and filesystem permissions, rather than some physical memory separation, virtualization or more advanced mechanism²⁰. This means that the whole security system relies on the expectation that it is impossible to escalate root privileges thanks to the micro-kernel architecture.

¹⁹ http://www.qnx.com/products/certified_os/secure-kernel.html

²⁰ <https://www.youtube.com/watch?v=z5qXhgqw5Gc>

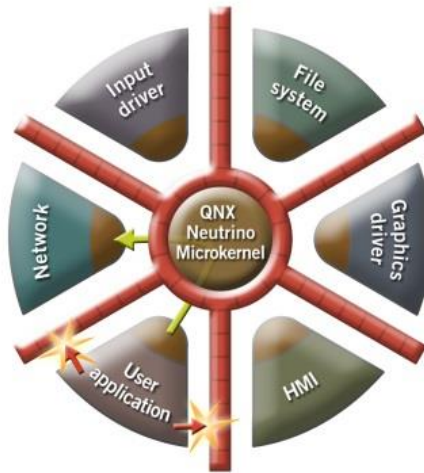


Figure 2.9 The Neutrino QNX operating system²¹.

The secure boot is executed from a root of trust embedded in the CPU itself, as shown in Figure 2.10. The CPU bootloader verifies the digital signature of the bootloader code before it can run, and the bootloader in turn verifies the digital signature of the OS [21]. In the same figure we see also that the platform supports four different kinds of applications.

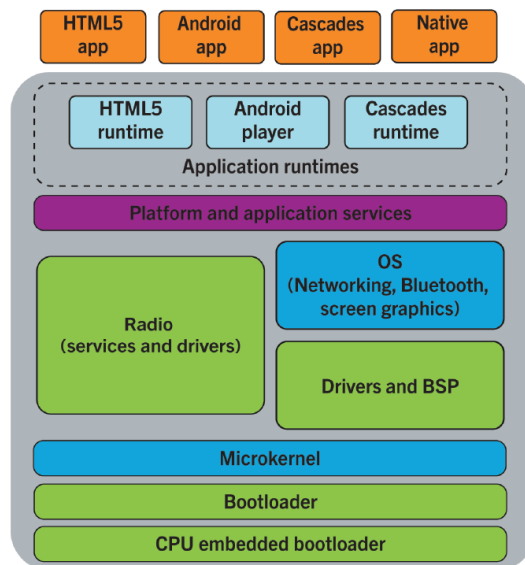


Figure 2.10 BlackBerry 10 platform architecture [21].

Application distribution is, like Apple, handled centrally by an official appstore called BlackBerry World. The verification process is also similar to Apple, where developers must first enroll and then submit their apps for verification, which according to BlackBerry is quite

²¹ http://www.qnx.com/products/certified_os/secure-kernel.html

aggressive²². Unlike other platforms, the apps are signed by Blackberry with the developer certificate online, meaning that they have a copy of the developer's private key²³. There are, however other ways to install apps on a BlackBerry device, but they mostly involved enabling the developer mode on the device [19].

An interesting curiosity to conclude this section is the recent debut of the last BlackBerry device, Priv, which runs Android as main OS. BlackBerry claims to have hardened the security of Android in various ways, by enhancing various security features already existing in Android and by integrating BlackBerry hardware based security²⁴. What this means for the future of the QNX OS is unclear.

2.2.5 Windows 10 Mobile security

The newest incarnation of Windows mobile has been launched in December 2015 and it is not easy to find detailed information about its security, except for a quite informative video from Microsoft Ignite 2015 Conference held in May 2015 [22]. All information in this section comes from that source. In Figure 2.11 we can see that most of the security features resemble those discussed for the other mobile platforms, so we will focus only on what is different.

Starting from the boot process, the mobile devices running this OS are required to comply with the UEFI (Unified Extensible Firmware Interface) specifications and implement a secure UEFI boot. In particular, TrustZone is used to implement a firmware version of the TPM 2.0 [23]. Having a TPM implementation enables also remote attestation, trusted boot and BitLocker functionalities. BitLocker, in particular, is the choice for disk encryption in Windows 10 Mobile. This means that everything that is written to disk is encrypted with the same key, rather than having a per-file key like iOS and BlackBerry.

The other interesting feature that distinguishes Windows from other mobile platforms is the tight integration with its desktop counterpart and the portability of its apps. The idea is to have what they call "universal apps" that can easily be ported with minimal modifications from desktop to mobile and vice-versa. Mobile devices can then be turned into full-fledged computers by connecting them to a dock station that provides keyboard, mouse and a screen.

Finally, it seems that although Windows can be run on devices from different OEMs, Microsoft will provide centralized security and system updates directly to all types of devices²⁵. In this sense it can be seen as an interesting compromise between the openness of Android and the centralized control of Apple and BlackBerry.

²² <https://developer.blackberry.com/builtforblackberry/documentation/criteria.html>

²³ <http://devblog.blackberry.com/2013/08/code-signing-keys-be-gone-welcome-blackberry-id/>

²⁴ <http://blogs.blackberry.com/2015/11/why-blackberrys-android-is-best-for-security-and-privacy/>

²⁵ <http://www.cnet.com/news/microsoft-to-control-software-updates-for-windows-10-mobile/>

Windows 10 Mobile: Layers of security






 App security	Store Apps	Cloud Services
	Business Store Portal	Browser security
 Data protection	Enterprise Data Protection	IRM & S/MIME
	Device encryption	Mobile Device Management
 Access Control	Microsoft Passport	Windows Hello
	Two Factor authentication	Network security
 Windows One Core	OS Services	App Platform
	Trusted Boot	Single source updates
 Secure Hardware	Only signed binaries	Attestation
	UEFI Secure Boot	TPM 2.0

Figure 2.11 Windows 10 mobile security architecture [22].

2.3 Discussion

In this chapter we have gone through most of the technical security offered by the four mobile platforms we considered. Finding the right level of detail is not easy, as we want to include as many relevant mechanisms as possible while not diving into too many technical details or report lists of supported protocols and security algorithms. We gave a recapitulatory overview in Table 2.3, but again, in no way exhaustive. A more detailed comparison of some chosen mechanisms can be found in the appendix. The main point we want to make is that these platforms do offer a wide range of security mechanisms and have a comparable level of maturity. The most important differences lie in the type of the deployment model. When more of the actors listed in **Error! Reference source not found.** coincide with the same entity, we can expect better security because there are less conflicting interests, fewer adversaries and therefore fewer threats. For instance, Apple and Blackberry are at the same time platform manufacturer, platform providers, marketplace providers, and actually administrators for normal users. This means that they: have a much greater control over their devices; can enforce a consistent policy that governs all layers; can push updates centrally to all devices simultaneously; have stricter policy and guidelines about app development and distribution; and can optimize their software for a specific hardware also from a security point of view. Google, on the other hand, only provides the OS, namely Android, which can be customized and run on many different types of hardware. This causes system fragmentation, delayed updates, greater possibility of bugs in some customized implementation, and device specific security.

	Mechanisms	Android	iOS	BB	Windows
<i>Hardware level security</i>	ARM TrustZone	Implementation dependent	Not clear if used	Not clear if used	YES
	Root of Trust (unique key)	Implementation dependent	YES	YES	YES
	Microkernel	NO	NO	YES	NO
	Trusted Boot	Implementation dependent	YES, Signatures	YES, Signatures	YES, TPM, UEFI
	Crypto processor	NO	YES	NO	NO
<i>Kernel/OS level security</i>	Encryption	Full-disk encryption	Full-disk + per-file encryption	Full-disk + per-file encryption	Bit-Locker
	Sandboxing/Permissions	YES	YES	YES	YES
	Exploit Mitigations	NX, ASLR	NX, ASLR	NX, ASLR	NX,ASLR
	Network security	Certificate pinning, VPN, TLS, EAP.....	VPN, TLS, EAP.....	VPN, TLS, EAP.....	VPN, TLS, EAP.....
	Authentication (Screen lock)	PIN, password, pattern, fingerprint, face recognition, ...	Passcode, fingerprint	Password	PIN, Password
<i>Infra-structure security</i>	Native BYOD support	Android for Work	iOS for enterprise	BlackBerry Balance	YES
	App-store	Google Play Third parties	App Store	BlackBerry	Windows Store
	Security updates	Delegated	Centralized	Centralized	Unclear

Table 2.3 Simplified summary of security mechanisms in the four mobile platforms.

Besides, to be consistent with their open model, Android allows also users to install unverified apps from third party app-stores if they wish to do so. Microsoft seems to have moved to a more open approach where they provide the OS, but still require the manufacturer to implement specific hardware support, and retain the possibility to update the software directly. We expect therefore that closed systems are easier to protect, and therefore less subject to infections or compromise. The data in the next chapter seems to partly confirm this assumption, but a more careful analysis reveals that this is not completely true, and that it may be possible to achieve similar levels of protection in all these platforms given a more restrictive deployment setting.

As a final note, we would like to point out that the security presented here assumes an attack model where we can somehow trust the infrastructure and where an attacker does not have prolonged physical access to the device. For instance, if we cannot trust the mobile network protocols to implement adequate protection, an attacker could use an infrastructure-based attack to break security. Using false base stations to eavesdrop on the user calls or send remote attacks to the baseband processor is a prime example. Similarly, if we cannot trust the marketplace providers to thoroughly verify apps before making them available for download, there is little the device can do on its own to prevent malicious apps to be installed. Supply chain attacks, where malicious code may intentionally be injected in the device at production time, either by an attacker or by the manufacturers themselves, is also something that goes beyond the security mechanisms presented here. Finally, mobile devices are difficult to protect physically due to their mobile nature. It is easier to forget them unattended, lose them, borrow them or steal them. An attacker with enough time, motivation and resources will most likely manage to bypass most security on the average device in most cases. Nevertheless, by being aware of these weaknesses, it can be possible to design ad-hoc mitigations or additional security measures that can reduce the risk of compromise to an acceptable level for some specific situations.

3 Security in practice

From what we presented in Chapter 2, it is clear that security is emphasized in modern off-the-shelf smartphones and tablets. However, the demand for new fancy consumer functionalities and the need to be the first to offer them on the market still seems to be the priority. So, while it is true that security is getting more attention, the probability of finding new bugs and vulnerabilities in these increasingly larger systems is also growing. In addition, despite the efforts of market providers to check for malicious apps and to enforce some quality standard²⁶, poorly coded and harmful apps still find their way also in official app stores²⁷. Even genuine apps that appear to do just what they advertise can become a security threat if used in the wrong way or if not properly tested against the specific security requirements of the environments

²⁶ <https://developer.apple.com/app-store/review/guidelines/>

²⁷ <https://blog.lookout.com/blog/2016/01/06/brain-test-re-emerges/>

where they will be used²⁸. Users are also part of the problem as they often consider security as a hindrance or do not pay particular attention to the risks to which they expose their devices. Despite this, we will show how statistics indicate that only a small percentage of devices seem to have been successfully compromised by some known threat, suggesting that the security mechanisms in place do work as intended in the majority of situations.

In the remainder of this chapter we review various security reports reporting these statistics together with a selection of known vulnerabilities in order to understand exactly what types of threats and attacks these devices are exposed to, whether they manage to protect against them, and if not, why.

3.1 Threats, vulnerabilities and related concepts

In order to correctly interpret the data that we are going to present in this chapter, it is important to clearly define some basic security concepts like threats, vulnerabilities and attacks.

A *threat* in general is defined as “any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service. Also, the potential for a threat-source to successfully exploit a particular information system vulnerability” [24]. Here we focus especially on this last sentence, where the threat-source can be identified with an *attacker* and the threat is the possibility of the system’s confidentiality, integrity or availability being compromised through the exploitation of a particular vulnerability.

A *vulnerability* is a “weakness in an information system, system security procedures, internal controls, or implementation that *could* be exploited by a threat source” [24], but not all vulnerabilities are as severe or easy to exploit. This is why all vulnerabilities documented in the Common Vulnerabilities and Exposures (CVE) system²⁹ are given a severity assessment based on the Common Vulnerability Scoring System (CVSS) standard, now at version 3.0³⁰. This assessment takes in consideration various metrics, like for instance what kind of access the attacker needs, if remote or local, how difficult it is to meet all conditions required for the exploitation to be successful, whether authentication is required, and so forth.

This leads us to the concept of *attack*. An attack is the series of actions that an attacker must execute in order to exploit a particular vulnerability on the system and realize the threat. Part of the attack is finding a path that allows getting to the system where the target vulnerability is located in order to execute the actual attack. The elements constituting this path are called *attack vectors*. Attack vectors can be classified according to what role they have in the attack, for instance “channels” that allow to transfer the code from one place to another, “enablers” that allow the malicious code to be unpacked or executed, “targets” that are the services or the data

²⁸ <http://www.gartner.com/newsroom/id/2846017>

²⁹ <https://cve.mitre.org/index.html>

³⁰ <https://www.first.org/cvss>

vulnerable to the attack, and so forth [25]. All the possible attack vectors, weighed by their criticality, form the *attack surface* of the system, which can give a measure of how probable or easy it may be to successfully attack the system and exploit its vulnerabilities to realize a threat. The malicious code used to actually exploit the target vulnerability and achieve the attacker's goal is called *payload*, while the object hiding and transporting the code is called the *carrier*.

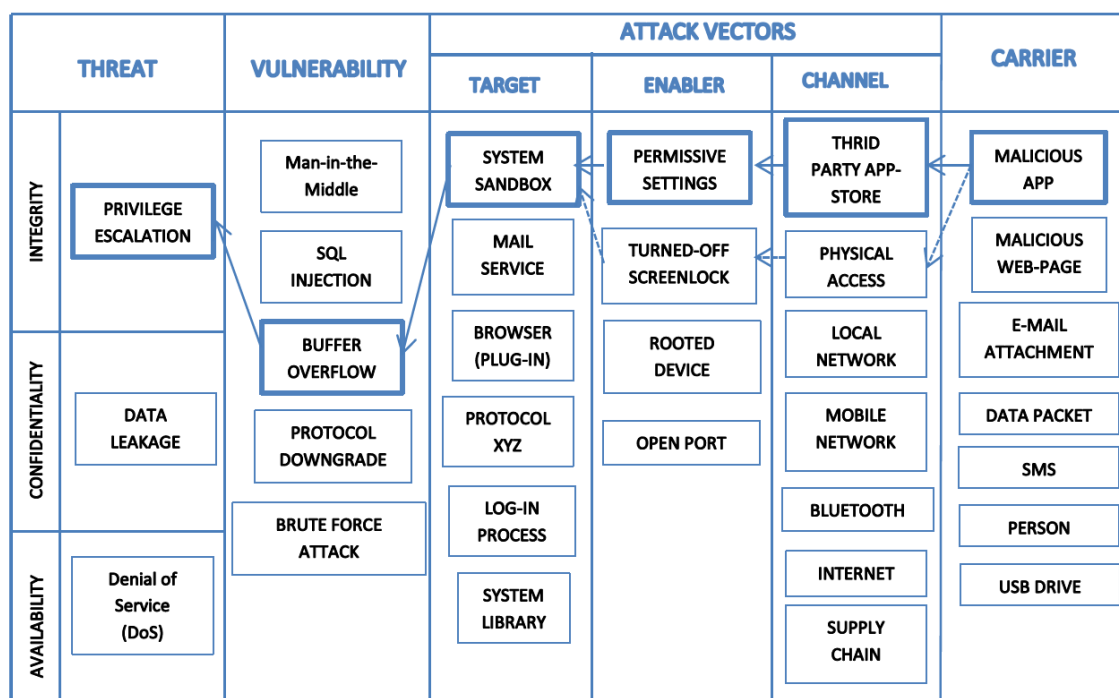


Figure 3.1 An example of an attack where the goal of the attacker is to gain root privileges on the device by executing a payload that targets a buffer overflow vulnerability in the device sandbox. This can be achieved by embedding the payload in a malicious app and trick the user into installing it from an app-store without review process. This can be enabled by the user by allowing side-loading. Other attack paths are also possible, like gaining physical access to an unlocked device.

To illustrate the concepts explained in this section we show some examples of attacks and attack vectors in Figure 3.1. This is not meant to be exhaustive or completely exact in any way, it is only meant to give the reader a feeling of how the different concepts are related to each other. In fact, another important aspect that does not emerge from the figure, is that for an attack to be successful, it is not always sufficient to exploit only a single vulnerability in one target. More often a series of pre-conditions must be satisfied, and more security mechanisms need to be circumvented in order for the attacker to reach his goal. So the picture is much more complex than shown here. However, it does contain some typical attack vectors that are very relevant for mobile devices [26]. In particular, *malicious apps* are the most common carrier of malicious code. The reason being that there are various channel attack vectors that can be used to

distribute them and it is relatively easy to inject a malicious payload into another seemingly innocuous application and trick the user into installing it on the device. A resourceful attacker may even manage to install them on the device before it reaches the market, by performing a so called *supply chain* attack. This type of threat is very insidious and particularly prominent when governmental threat-agents are involved. Official and third-party app-stores are, however, the typical channels. In the first case a weakness in the review process is to be used to first publish the malicious app, so it is much more likely to find these apps in third party stores with little or no review. Alternatively they can be directly installed *remotely* from malicious web-sites or manually installed by an attacker with *physical access* to the device. An enabler that is needed in almost all these cases is that the user must intentionally or indirectly allow installation of untrusted application. This can be done on some platforms by changing some security settings, while in others it is possible only by forcefully bypassing the system locks. We can distinguish three such techniques:

- *Jail-breaking*: is the process of removing restrictions imposed by the manufacturer on the device, like the impossibility of installing non-approved third-party software. This is in itself an attack that exploits a vulnerability in the system as it is not a provided feature.
- *Rooting*: refers to gaining root privileges in the system, so that full control can be achieved. For instance, it becomes possible to turn off security mechanisms, grant access to non-trusted application, remove system components, and so on. Unlike jailbreaking, rooting is sometimes allowed on some systems like Android, but if not it can also be achieved through a vulnerability exploited by malware or users interested in customizing their device.
- *Unlocking*: a third alternative is to unlock the boot-loader of the device in order to install a custom OS already configured with root access. This option can also either be offered as functionality or achieved through exploitation.

It is easy to see how deactivating or compromising the security mechanisms in place on the device opens up the system also to external attackers, that can now more easily reach the vulnerability they try to exploit. The fact that “normal” devices are less susceptible to infections compared to “rooted” ones is also confirmed by the data presented in the next section.

Summarizing, mobile devices can be vulnerable to attacks similar to those of their stationary counter-parts, like supply chain attacks and remote attacks through malicious web-sites, mails or similar. In common with portable devices like laptops we find attacks that leverage local access like Wi-Fi, Bluetooth and NFC (Near Field Communication). However, they are also more exposed to physical attacks, given they are more easily accessible: removing the memory card or connecting them to a malicious device via USB is relatively straightforward. Attacks through the *mobile network* are also exclusive to this class of devices: malicious base-stations, vulnerabilities in the base-band processor, and faulty processing of SMS and MMS are also all possible entry points to the system.

One last observation is that it is difficult to classify what code or activity is actually malicious. Based on different definitions, one can get different statistics on the infection rates of active devices. For instance, some apps can be considered malicious even if they do not exploit any vulnerability. A lot of sensitive data can be collected from a device simply by having completely legitimate permissions. Examples are: contact lists; position; IP-address; and phone number. Again, the user can make a big difference by wisely choosing what to trust and what not at installation time. The numbers reported in the next sections come from documents that may have different definitions of malware and harmful code, but since they all seem to validate each other we will not discuss these definitions in detail. Just to give an example, though, as of 11/1/2014 Google used the term PHA (Potential Harmful Applications) to indicate the following categories: Generic PHA; Phishing; Rooting Malicious; Ransomware; Rooting; SMS Fraud; Backdoor; Spyware; Trojan; Harmful Site; Windows Threat; Non-Android Threat; WAP Fraud; and Call Fraud [27].

Therefore, although the existence of a vulnerability can in principle constitute a threat, it is not enough alone to claim that the system is not secure, because whether the threat can be realized in practice depends also on the existing attack surface. For instance, Java-script vulnerabilities in a web-browser may not be exploited if Java-script is disabled, or if no internet connection is available. The reader should keep this in mind when considering the infection rates of modern mobile devices reported later in this chapter. Those numbers are not a definitive indication of their security, but they indicate only how secure they are in average given the typical usage and configuration in a global consumer market setting. What they can tell us, is which attack vectors are more likely to be used in modern mobile devices. This knowledge can later help us to reduce the attack surface by adopting additional security mechanisms, mitigations and careful usage, so that adequate security may be provided in specific scenarios, despite existing vulnerabilities.

3.2 Overview of security reports and statistics

Here we give an overview of the status of vulnerabilities and known infection rates based on publicly available reports and sources. Most of this data comes from the feedback that antivirus companies get from the devices on which their software is installed, so it gives only a partial sample of the complete picture. In the case of Android, however, the feedback comes from the Google services installed on the majority of Android devices and it gives a much more comprehensive overview compared with antivirus software feeds. Also, some statistics refers to the number of devices potentially exposed to a threat, i.e., which have an unpatched vulnerability, while others refers to actually infected devices, i.e., where some vulnerability was actually exploited. We will, for what is possible, distinguish between the two. One thing we have not considered here are the reports investigating the number of apps containing malicious code or new malware found in the wild. Those numbers would only give an idea of the potential threats around, but not how effective they are, which is our main concern.

3.2.1 Cross-platform vulnerabilities

Some vulnerability might affect more than one platform because they are found in some standard common component, library or protocol used by all of them. Examples are the SSL vulnerabilities HeartBleed³¹ and Freak³² found in the Open SSL library used by Android, iOS and BlackBerry. Vulnerability in browser plug-ins like Adobe Flash³³ can also affect various platforms simultaneously. It is not clear whether any of these vulnerabilities was at any point exploited on mobile devices. Baseband processor vulnerabilities may be platform specific, but often the same firmware and radio processor component is used by different manufacturers in their devices [28] and vulnerabilities can be leveraged through the mobile network by using false base-stations as it was recently demonstrated for a Samsung device³⁴.

3.2.2 Android

As shown in Figure 3.2, Android is consistently reported as the platform most exposed to attacks, with 99% of all mobile malware targeting it [29]; this despite having less than half of the documented vulnerabilities compared with iOS^{35,36}. There are mainly three reasons for this [30]. One is that Android is the most widespread platform on the mobile market, and the sheer number of potential targets alone makes it particularly attractive for attackers. The other is that Android is very fragmented, which makes it more difficult to keep all existing versions up-to-date, especially since the responsibility for patching the different versions falls to the specific manufacturer rather than Google itself. This means more vulnerable targets. Last, but not least, the possibility of easily enabling the installation of untrusted sources (side-loading) without the need of rooting the devices has led to the flourishing of third-party app-stores where the vetting process of applications is often non-existing. Here it is extremely easy to publish apparently legitimate apps that have been repackaged with some hidden malicious code and then self-signed by the attacker.

³¹ <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160>

³² <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-0204>

³³ <http://www.adobe.com/support/security/bulletins/apsb13-11.html>

³⁴ http://www.theregister.co.uk/2015/11/12/mobile_pwn2own1/

³⁵ https://www.cvedetails.com/product/19997/Google-Android.html?vendor_id=1224

³⁶ https://www.cvedetails.com/product/15556/Apple-Iphone-Os.html?vendor_id=49

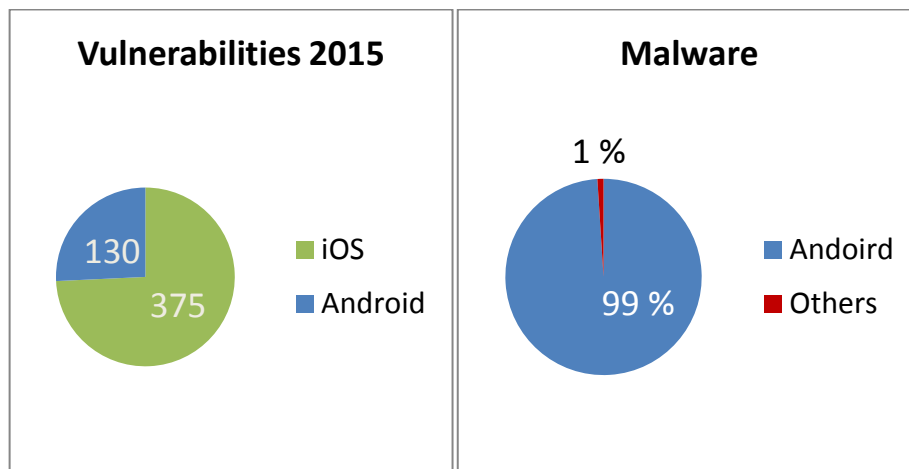


Figure 3.2 Despite fewer vulnerabilities, Android is the most targeted platform.

However, the data in the Google report on Android security for 2014 [27], confirmed also by other sources [31, 32, 33], reveals some interesting details that indicate that this difference is not due to inadequate security, but rather other external factors. In fact, it appears that worldwide only less than 1.3% of Android devices is infected by some sort of PHAs. Of course, 1,3% of hundreds of millions of devices is still a big number, but if the reason why Android stands for most infections among mobile platforms was just bad security, we would expect much higher infection rates. Instead, in accordance with our earlier observations, it turns out that if we consider only devices that are not rooted and that only download apps from Google Play store, the number of infected devices is less than 0.1% as shown in Figure 3.3. Non-rooted devices that download from other stores than Google Play (side-loading), account for 0.7% of the infections, while devices with some PHAs installed (including also rooting apps) are in average around 0.5% of the total. In other words, devices whose integrity has not been compromised by rooting or with side-loading enabled, and that are therefore comparable with closed systems like iOS and BlackBerry, are an almost negligible percent. It is also interesting to see how there are huge variations by locality. In some countries like China where Google services are not available and side-loading and rooting are therefore necessary to use Android devices, the percentage of infected devices reaches peaks of over 7%. One thing to point out is that Google did not start detecting devices with side-loading before the last quarter of 2014, so it is a somewhat limited sample time-wise. Nevertheless, the amount of infected devices during 2014 with and without side-loading, excluding rooting applications, has been lower than 0.5 %.

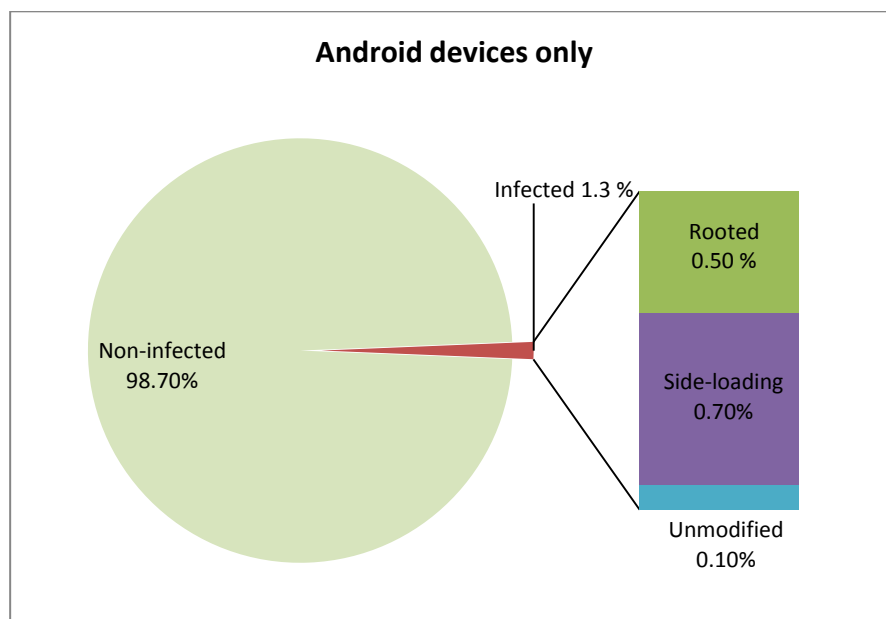


Figure 3.3 In practice the number of infected Android devices that have not been rooted and only use Google Play to install apps is around 0.1% of all Android devices worldwide. The reason why so much malware targets Android is that the amount of devices that download from unsafe sources or are rooted, and therefore are easier to compromise, is much higher than for other platforms.

The incidence of exploitation attempts for other types of vulnerabilities outside of Google Play is also discussed in the Google report, but it is used mostly to show the effectiveness of the Verification Apps Tool and not very relevant to this report. The research in [27] has also not uncovered any large scale malicious exploitation attempt of other SSL vulnerabilities.

This does not mean that downloading apps only from Google Play guarantees 100% protection. There is, after all, a small percent of such devices that are somehow infected. This can be explained in various ways. One is that not all malicious apps can be detected by Bouncer, and therefore some end up in Google Play [34, 35]. The other is that legitimate, innocuous apps contain exploitable vulnerabilities and can be used as an attack vector once installed [34]. There are even some documented cases in which malicious apps were pre-installed on the phone, meaning that the attacker managed to subvert the supply chain³⁷. Additionally, vulnerabilities that make it possible to inject malicious code into legitimate applications without altering the original digital signature³⁸, or create fake certificates³⁹, could also be used to install malware without enabling untrusted sources, or pushing malicious updates on legitimate installed apps. Besides, due to the high fragmentation of Android versions, it is also possible that most

³⁷ <https://blog.lookout.com/blog/2014/12/04/deathring/>

³⁸ <https://bluebox.com/uncovering-android-master-key-that-makes-99-of-devices-vulnerable/>

³⁹ <https://bluebox.com/android-fake-id-vulnerability/>

infections affected older versions of the platform that still have many vulnerabilities that have been patched in newer devices.

Finally, it is quite certain that many infections have not been detected at all, either because the infected device does not use Google services or any anti-virus, or because the exploited vulnerability is either still unknown or difficult to detect. Other known vulnerabilities discovered recently, like Stagefright⁴⁰, could also have been exploited using other attack vectors than side-loaded application and network access, making it unclear whether someone has been affected by them. A further possibility is that an attacker can have physical access to the device, and bypass security if the USB debugging option is enabled, and the screen-lock is either not active or easy to bypass. Even installing forensics tools in the boot partition to read volatile memory would be possible given the right conditions [36].

3.2.2.1 *TrustZone TEE*

We mentioned in Section 2.2.2 that some OEMs provide a TEE implemented in TrustZone, in addition to their customized Android OS. Unfortunately, exploits have been found also for this “secure” solution. In 2013 the Motorola boot-loader was unlocked through a vulnerability in the TrustZone kernel⁴¹; the same year Thomas Roth showed how to create TrustZone based rootkits [37]; and Di Shen showed how he hacked the Huawei’s TEE at Black Hat 2015 [38]. Wrong implementation of secure functionalities using TrustZone has also been shown to be a real security problem [39].

3.2.3 iOS

As mentioned in the previous section, iOS has actually more documented vulnerabilities than Android, but despite this it has generally been considered more secure, and has been the platform of choice for the enterprise market [40]. The main reason lies probably in its closed nature that makes it much more difficult to allow for untrusted apps to be installed, and easier to keep it updated. iOS devices need, in fact, to be jail-broken in order to use third-party app-stores, and this in turn requires the exploitation of some vulnerability in the system. Still, around 8% of iOS devices globally are estimated to have been jailbroken [40], and it is mostly these devices that have been targeted and successfully infected by malicious code, at least until November 2014. Before that, in fact, only two cases were known of malicious apps that had bypassed security controls and had been published on iTunes (and could therefore be installed on non-jailbroken devices): the LTBM adware and the FindandCall worm. Then WireLurker was discovered [41], and Pawn Storm⁴² and Yispector⁴³ followed the next year. Another potential attack was also found in 2014 called Masque Attack⁴⁴. What all of these have in common, is the new attack vector used, namely ad-hoc (or in-house) provisioning through enterprise certificates [42]. A proof of concept showing how a similar attack could be performed

⁴⁰ <https://blog.zimperium.com/experts-found-a-unicorn-in-the-heart-of-android/>

⁴¹ <http://blog.azimuthsecurity.com/2013/04/unlocking-motorola-bootloader.html>

⁴² <http://blog.trendmicro.com/trendlabs-security-intelligence/pawn-storm-update-ios-espionage-app-found/>

⁴³ <http://researchcenter.paloaltonetworks.com/2015/10/yispector-first-ios-malware-attacks-non-jailbroken-ios-devices-by-abusing-private-apis/>

⁴⁴ <https://www.fireeye.com/blog/threat-research/2014/11/masque-attack-all-your-ios-apps-belong-to-us.html>

by connecting the device to a fake charger was also shown in 2013 [43], but since then some new security mechanisms have been implemented to prevent that specific attack.

Apple provides, in fact, an alternative and legitimate way to install apps on non-jailbroken devices that bypasses iTunes and all the security controls: enterprise provisioning. This method allows enterprises to install or push their own apps without going through iTunes, but in order to do that a special enterprise certificate must be used to sign them. Such certificates are harder to obtain than usual developer certificates⁴⁵, exactly because they might be misused in the way they did. Still, someone went to the trouble of setting up a fake company and paid to obtain such a certificate, in order to spread malware. Also, WireLurker could spread only through USB connection to an infected MAC workstation (where it is possible to install untrusted apps), while Yispector could also be directly downloaded and installed on the mobile device.

Thus, attacks on iOS are also possible and just as dangerous as on Android.

3.2.4 BlackBerry 10 and Windows 10 Mobile

These two mobile platforms have a very small share in the mobile market, which also means that they are not a particularly interesting target for most attackers. This may be one of the reasons why not many vulnerabilities and cases of exploitation are known for them. Windows in particular, just came out a couple of months ago, so there is no known exploited vulnerability as of yet. However, one observation we can make is that with “universal apps” there is a risk that vulnerabilities found on Windows Desktop applications may also apply to their mobile counterpart, and that the effort to give a unified experience to mobile and desktop users may also lead to common vulnerabilities.

As far as BlackBerry 10 is concerned, quoting from [19]: “One public jailbreak thus far has affected QNX-based BlackBerry devices — DingleBerry, released in November 2011⁴⁶. No jailbreaks have directly affected BlackBerry 10”. In general only few vulnerabilities are reported⁴⁷, and many seem to belong to the cross-platform type mentioned in Section 3.2.1. Being that BlackBerry is mostly used in governmental environments, it can also be the case that some successful targeted attacks have been performed given the high profile of the targets, but that they have been kept quiet or never discovered. Research is also very limited compared to Android or iOS, as we are aware of only 4 such works [44, 45, 46, 47, 48].

3.3 Discussion

One problem of the analysis in this chapter is the unbalance in the available data for the different mobile platforms. While for Android we have both various independent sources like anti-virus companies and Google itself that share their data, much less extensive statistics are

⁴⁵ <https://developer.apple.com/programs/enterprise/enroll/>

⁴⁶ <http://crackberry.com/so-you-want-rootjailbreak-your-blackberry-playbook-dingleberry-here%E2%80%99s-how-do-it>

⁴⁷ <http://support.blackberry.com/kb/articleSearch?language=English&keyword=vulnerability>

available for iOS and close to nothing for BlackBerry and Windows 10. Thus, one should not conclude that one platform is more or less vulnerable than others solely based on this data.

Nevertheless, one thing that we can conclude is that the preferred carrier for malware in mobile platforms seems to be malicious apps. Android is not surprisingly the most exposed due to its overwhelming market-share, the possibility of side-loading and the large number of third-party app-stores present on the Internet, but we saw how iOS also is affected thanks to a similar type of mechanism based on enterprise certificates, although at a smaller scale.

Ignoring intentionally rooted and jailbroken devices, or devices that must use side-loading because of the lack of official app-store availability in a certain geographical area, we are still left with a small percent of infected Android devices that apparently are used and configured correctly. In Section 3.2.2 we tried to explain what other attack vectors could have been used in these cases, but we cannot be certain, so we can only assume that despite much advances in security, mobile platforms still are vulnerable to various types of attacks, and that attack strategies evolve continuously in order to exploit any possible weakness in the security model. However, that so very few devices in percentage have been affected is an indication that security does work quite well in general, given that those numbers are indeed representative of the actual situation, and that the situation is much less dramatic than often depicted in the news. The better security is implemented at platform level and integrated with the infrastructure, the more difficult it becomes for the attackers to find easy attack vectors and for users to intentionally or accidentally disable security.

Still, perfect security does not exist. Some malicious apps will always slip through the official app-stores security controls, and some user will be tricked into installing it. Users will also often not like the limitations imposed by manufacturers and will try to break them exposing their devices to multiple threats. Social engineering will also be a prominent threat no matter what security is in place. However, most of these problems affect mostly the mass consumer market, where it is statistically more likely to meet all necessary conditions for a successful attack. In a more controlled and restricted environment, where it is possible to enforce centralized security policies, enhance security, limit risky activities and educate users, many of these threats could be eliminated or at least mitigated⁴⁸, although targeted attack may become more prominent. In the next chapter we look at existing approaches to achieve different level of controls on a mobile device.

⁴⁸ A report from Lookout seems to contradict this last statement [40], as they found various malware on devices associated with different enterprises, but they do not specify what type of device management solution they used and how they were configured.

4 Possible approaches to enhance security

The previous chapter emphasized how most security threats come from installing untrusted apps, either by using side-loading, or visiting malicious web-sites or not carefully reading warning messages. Rooting or jailbreaking a device makes things even worse by deactivating critical security mechanisms. If COTS mobile devices are to be adopted for security critical tasks, we need at the very least to eliminate these prominent threats, and try to mitigate the remaining ones.

The factors that can affect the trust we can have in a device and its security, are mainly four: its intrinsic security; the type of usage; the owner; and the user. With the first, we mean the platform security that is already built in the device, which gives us a baseline security: the software and firmware security mechanisms, the hardware security, and possibly its certifications. In other words, what we have seen in the first two chapters. The type of usage can include: the applications that are going to be installed on the device, its configuration, the type of maintenance it is subject to, the tasks it will perform, the data it will handle, and its physical protection. We distinguish then between owner and user as they may not necessarily be the same. The owner is the administrator, who has the highest privileges on the phone and can decide what can or cannot be installed and which configuration to enforce. The user is the one that has physical control of the device in practice and can potentially expose it to most threats by misusing it, losing it or intentionally bypassing the predefined configuration. In addition we should also trust the infrastructure, but this is a separate topic.

The relationship between owner and user is quite central in recent mobile devices, which offer native support for three types of ownership models. The reason is that steadily more enterprises understood that it can be advantageous to let their employees use their private devices also for work activities, but it has so far been challenging to find the right balance between the degree of control needed by the enterprise to trust the device and the users' right to freely use their devices. This is changing thanks to new security mechanisms which enterprises can leverage to enforce their security policies without necessarily taking complete control of the device. Still, in some environments with even higher security requirements like classified military network, even having complete control of the device may not suffice. In this case additional functionalities and assurance may have to be implemented in the platform itself, or dedicated solutions are to be developed.

In the next sections we discuss the different ownership models and what they can give in terms of security when applied to commercial devices, and how commercial and military actors are also trying to develop dedicated secure solutions by using commercial mobile platforms as a starting point.

4.1 Ownership models supported by COTS device

The latest releases of all mobile platforms we reviewed in the first two chapters come now with native support for BYOD solutions, which until now had to be especially developed as add-ons with limited capabilities. Mainly three ownership models are provided: 1) the user retains full control over the whole device, 2) the enterprise can control a part of the device and 3) the enterprise has full control. Each can be suitable for different types of scenarios, with varying degree of security.

4.1.1 User ownership

There are some situations when it may be desirable to let people use some service or application on their own devices, without any particular control or requirement. The reasons can be many. One scenario is emergency situations, where it may be essential to collect as much information as possible to create a complete picture of the situation. In this case any bystander or responder can just download an app on their phone and use it to take pictures, videos, and report different kind of events. Another is that a company may not have the resources to buy, configure or maintain some dedicated devices, and therefore are willing to accept the risk of letting their employees use their private devices for work related tasks. This use model implies, in fact, that the enterprise has virtually no control on the device, which may already be compromised by some malware, be rooted or generally not trustworthy.

In this model, the enterprise service comes as a *dedicated app(s)*, and all security would have to be implemented in the application itself, by leveraging either the platform security mechanisms, or custom security packaged within the app itself. One can potentially achieve a good level of security given that the app is well written, includes proper encryption, authentication and secure communication components, and that the underlying platform is not already compromised. The problem is that we cannot put much trust in the device itself as we have no reliable way to assess its trustworthiness. Hence, the risk of compromise is relatively high. Still, interesting use-cases, where a risk analysis may show that the benefits outweigh the potential risks, surely exist.

4.1.2 BYOD: shared ownership

A more advanced version of installing just some applications on a private device consists in a complete BYOD solution. Until now, the main limitation was that users had to completely give up control of their devices in order to install company software that enforced the company's security policies and monitored the status of the device. This was especially bothersome when the security policy required wiping the memory of the phone if a violation was detected, including the user's private data, or made the user unable to install their favorite apps. However, newer devices support natively a partitioning of the phone into a personal and a work profile, which are managed at system level to guarantee separation between the two. In this way, the employer can own the work profile, while the user can still control the device itself and the personal profile. Each profile would have its own applications, policies, dedicated encryption, authentication, and data. Deleting the work profile, would therefore not affect user data. We

have seen examples of such solutions earlier in the report: Samsung Knox, Android for Work, iOS for Enterprises and BlackBerry Balance.

It is important to remember that these solutions are still based on the assumption that the user cannot root or jailbreak the device without that being detected by the security mechanisms underlying the work profile, which can then be blocked for access. A malicious user may still be able to compromise the system before activating the work profile exploiting new vulnerabilities that can go undetected before the work profile is activated, and therefore potentially break company policies at a later moment. Still, the bar is much higher than for the previous ownership model, as these BYOD solutions also have effective mechanisms to detect whether a device has been tampered with, and lock the user out of the work environment. An example is the KNOX bit⁴⁹ used in Samsung devices with Knox, and the automatic wiping of the device in case the boot-loader is unlocked in order to install a new custom OS, which is present in most devices.

4.1.3 Enterprise ownership

Company owned devices differ from usual BYOD solutions in that the device is bought and configured by the company before being handed over to the employee. It can also be the case that no personal profile is set up at all, making the device at all effects, just a work tool. This guarantees better control to the company that knows the initial status of the device when the work profile is configured, and can also regularly recall devices for inspection and reconfiguration. In this case it is easier to monitor and detect compromise attempts, although there is always a chance of exploitation.

4.1.4 Management solutions

A BYOD solution needs some form of device and application management in order to define and enforce security policy when the device is both on-line and off-line. Mobile platforms provide built-in security functionalities for these purposes, but additional management software that builds on them is needed to achieve good usability and efficiency. There are mainly four types of tools one can use to administrate different aspects of the device:

- *Mobile Device Management (MDM)* deals with the administration and configuration of the device itself. For instance: enrolment; configuration; enforcement of password, protected connections and other policies; application and patch provisioning; and remote monitoring. These tools are implemented using application programming interfaces (APIs) released by mobile operating system providers⁵⁰.

⁴⁹ <https://www.samsungknox.com/en/faq/what-knox-warranty-bit-and-how-it-triggered>

⁵⁰ <http://asmarterplanet.com/mobile-enterprise/blog/2014/09/mdm-mam-emm.html>

-
-
- *Mobile Application Management (MAM)* provides individual app administration, provisioning and monitoring usually through a dedicated enterprise appstore. Some solutions provide also an SDK to add security extensions to applications [49].
 - *Mobile Information Management (MIM)* or *Mobile Content Management (MCM)* tools are used to protect data. This means encryption and strong authentication used to access data on the device or from the device. In some cases even secure containers where applications can run securely.
 - *Enterprise Mobility Management (EMM)* suits provide all or some of the above listed functionalities in a consistent framework.

Although it is possible to exercise a good deal of control on a device by using these tools, one must not forget that their functionalities rely on the assumption that the underlying security mechanisms offered by the OS and the hardware are working correctly. There is unfortunately no sure way of detecting whether a device has been rooted or jailbroken [50, 51]. An overview of commercial EMM can be found in [49].

4.2 Dedicated solutions

Although a managed enterprise owned device can already provide a degree of security adequate for many situations, there are situations when additional features at platform level or even higher security are required. In this case dedicated solutions are usually developed. In particular, here we are interested in commercial technology that has been hardened for military or governmental use, and in military projects experimenting with the possibility of adopting commercial mobile devices for tactical use. We do not consider dedicated products specifically developed from the ground up for higher assurance.

4.2.1 Commercial (based) products

In the U.S., the NSA (National Security Agency) has started a the “Commercial Solutions for Classified” (CSfC) program to accelerate the adoption of commercial products for use in governmental and military offices, including mobile devices, and redacted a guide that defines a layered approach to harden the security of mobile devices in order to handle sensitive information securely: the NSA’s Mobility Capability Package⁵¹. Devices that follow these guidelines and are approved according the Protection Profile for Mobile Device Fundamentals⁵² are listed in the “Commercial Solutions for Classified Program Components List”⁵³. Among them we find⁵⁴: the Boeing Black phone⁵⁵; Samsung Galaxy Note 4 with Android 5; Samsung Galaxy S6; Microsoft Windows 10; Samsung Galaxy Note 5 and Tab S2; and Apple iOS 9. However, whether they can only handle sensitive but unclassified information or also higher

⁵¹ https://www.nsa.gov/ia/_files/mobility_capability_pkg_vers_2_3.pdf

⁵² https://www.niap-ccevs.org/pp/pp_md_v2.0.pdf

⁵³ https://www.nsa.gov/ia/programs/csfc_program/component_list.shtml

⁵⁴ https://www.niap-ccevs.org/pp/PP_MD_v2.0/

⁵⁵ http://www.boeing.com/assets/pdf/defense-space/ic/black/boeing_black_smartphone_product_card.pdf

classification levels is unclear. Given that the guidelines mainly require standard security mechanisms like those presented in Chapter 2, we assume that it is the former case, unless additional security measures and a dedicated infrastructure are in place to form a complete secure solution as indicated in the Mobile Security Reference Architecture document by Federal Chief Information Officers (CIO) Council⁵⁶.

The Blackphone by Silent Circle⁵⁷, which may appear as an example of a particularly secure phone, focuses mostly on privacy rather high-level security. Still, removing third party services from Google and allowing the user to deactivate privacy sensitive peripherals, can indeed provide better security than most other commercial devices, but it is unlikely good enough for classified use. In the same category we find the Cryptophone 500i by GSMK, which also runs a hardened version of Android, with dedicated baseband protection and strong encryption, configurable security profiles and verifiable source code⁵⁸.

A commercial security solution that has been used in conjunction with other technologies to develop products that may be approved for higher classifications like RESTRICTED is Samsung Knox. The Green Hills high assurance separation kernel Integrity for instance has been integrated in the Samsungs Knox mobile enterprise family to offer strong isolation in Android⁵⁹. We speculate that this solution adds a layer of virtualization so that the Knox environment can run in a separate partition with its own kernel and therefore offer better isolation. It is not clear whether commercial Samsung devices actually adopt this solution. Another example is the Tiger/R from Sectra, a known producer of secure mobile phones for classifications as high as SECRET⁶⁰. This phone runs on a security enhanced version of the Knox OS and is designed for use at the RESTRICTED security classification level in Europe⁶¹. A similar level of certification was also announced for a specific BlackBerry solution in Germany⁶².

Finally, as part of the CSfC program, the Defence Information System Agency (DISA) announced that their Defense Mobile Classified Capability-Secret (DMCC-S) is fully operational. A mobile phone for use at the SECRET classification based on commercial technology is in fact to be expected later this year⁶³. Apparently it will be able to connect to SECRET networks through the Secret Internet Protocol Router Network (SIPRNet) and will not store data at rest, but provide email and secure voice communications via a secure Voice over Internet Protocol (VoIP) capability⁶⁴.

⁵⁶ <https://cio.gov/wp-content/uploads/downloads/2013/05/Mobile-Security-Reference-Architecture.pdf>

⁵⁷ <https://www.silentcircle.com/products-and-solutions/devices/silent-os/>

⁵⁸ <http://www.cryptophone.de/upload/files/46/original/CP500i-Brochure.pdf>

⁵⁹ <http://www.ghs.com/mobile/products/samsung-knox-hypervisor/>

⁶⁰ <http://communications.sectra.com/news-and-media/press-releases/eu-approves-new-model-the-sectra-tiger-secure-mobile-phone-secret>

⁶¹ <http://communications.sectra.com/news-and-media/press-releases/samsung-and-sectra-in-cooperation-secure-smartphone-european>

⁶² <http://press.blackberry.com/press/2013/blackberry-10-receives-nato-approval-for-restricted-communicatio.html>

⁶³ <http://www.c4isrnet.com/story/military-tech/mobile/2015/09/03/pentagon-top-secret-smartphone-expected--fall/71648428/>

⁶⁴ <http://www.disa.mil/enterprise-services/mobility/dmcc/secret>

4.2.2 Dedicated proprietary solutions

There are also mobile solutions that are not based on commercial technology found in COTS products, but are specifically developed from the ground up to achieve higher assurance and security. We will not list them all in this report because they are out of our scope and details are usually disclosed only under Non-disclosure Agreements (NDAs), but just to give an idea to the reader of what type of products we mean, we can mention the mobile version of Pike OS high-assurance kernel supporting TrustZone⁶⁵ and the Secunet mobile solution based on their SINA⁶⁶ kernel.

4.2.3 Military research projects

Given the advantages of commercial smartphones and tablets, it is a natural to wonder whether there is a real possibility to use them in a military setting, more specifically on the battlefield. In this case, devices are to be used in rough environments, where neither mobile nor Wi-Fi connectivity may be available or allowed, with opponents that have advanced cyber- or electronic warfare capabilities and where reliability and confidentiality are critical.

On one hand, most commercial devices are not designed to withstand intense, prolonged and rough use; it is not possible to interface them with tactical radios or satellite communications; their cryptographic suites do not support proprietary military grade algorithms; hardware keys and certificates are under the control of the manufacturer; and they would most likely not be possible to certify for a high enough assurance level as they are. On the other hand, much of the smart technology we are used to in our daily lives is not available in a military context, and could potentially bring huge advantages in many situations. It is not surprisingly then, that there are various other military research projects looking at how to adapt commercial mobile device for military needs. Most of these projects are naturally classified and it is not possible to gather information about them, but for at least two of them there is some documentation that can reflect what kind of solutions and approaches are being tested.

The first is the Transformative Apps or TransApps project from DARPA⁶⁷. The focus of this project has been mainly to develop a platform independent military app-store for tactical apps, and develop a new agile approach to app developing that includes real-time soldiers' feedback. However, in order to be able to test this in the field a platform with adequate security was needed, so COTS devices running Android "with custom multilayered security" have been used in practice. This refers probably to the Mobile Armour program that contracted a security enhanced Android to Invincea⁶⁸ in 2012. Details on these security enhancements are unfortunately not available, but among other things, they tested support of tactical radios as an alternative to mobile and wireless communication.

⁶⁵ <https://www.sysgo.com/news-events/press/press/details/article/sysgo-demonstrates-pikeosTM-and-androidTM-running-arms-trustzoneR/>

⁶⁶ <https://www.secunet.com/en/topics-solutions/high-security/sina/sina-tablet-s/>

⁶⁷ <http://www.darpa.mil/attachments/DrPrabhakar-26Mar14.pdf>

⁶⁸ <https://www.invincea.com/2012/06/defense-advanced-research-projects-agency-darpa-awards-invincea-21-4-million-contract-to-create-secure-android-smartphones-and-tablets-for-u-s-army/>

The other project we are aware of is the Dutch PROMISE [52]. PROMISE is an acronym for PROject Multi-touch Information System Experiment. Similarly to TransApps, this project aims at delivering a complete solution for tactical systems including a dedicated military app-stores and specialized apps, based on commercial mobile devices and possibly hardening of available apps. They also claim that by using MDM solutions and other commercial security measures it will be possible to certify this solution for “RESTRICTED” level. At the current stage their platform security relies on a customized Android, where the following security additions were made to the kernel: PLE (payload encryption solution) VPN for network security; Applocker, to only allow PROMISE apps on the smart devices; removing all Google API's and internet access; unlock patterns as authentication; stringent policies such as screen-lock timeout; and use of a dedicated app store (open source). However, this means that rooted devices were used, without any central management solution. These shortcomings are recognized by the project, and collaboration with manufacturer to implement the required changes in locked devices and secure MDM solutions are recommended as future work. Use of SIM cards as crypto controllers is also suggested as a possible solution to the key management problem.

4.3 Discussion

From the overview given in this chapter, it is clear that there are various solutions that can be adopted when using commercial mobile devices for security sensitive tasks. Each solution can provide a different level of security at the cost of flexibility, economic impact and amount of resources needed to deploy and maintain the solution. Table 4.1 summarizes the main aspects of the different ownership models. It is obvious that the more control the enterprise, in our example the Department of Defence, has over the device, the more trust we can have that the security mechanisms are active and working as expected. The reason is that, among other things, security enforcement can be extended from a single app to more layers of the device, so that it becomes more and more difficult to circumvent. Naturally, the more aspects of the device we need to control the more tools, expertise and resources we need, so that costs also will increase, including the acquisition cost of the devices themselves. At the same time, the more a solution is integrated with a specific platform, the less flexible it becomes in the event that devices need to be replaced or updated. In fact, while an app can be adapted relatively quickly for a new version of the OS or a new device, a customized OS would require much more work.

One other reason why controlling more aspects of the device gives better security is that this will reduce the number of different actors involved in the mobile platform. As we discussed in Chapter 2, the more actors in **Error! Reference source not found.** coincide with the same entity, the fewer adversaries we need to account for and the better the security mechanisms can be integrated across different layers of the platform. Table 4.2 shows an example of how the Department of Defence could assume steadily more roles in the different ownership models and therefore minimize the potential adversaries. The red boxes indicate actors that cannot be completely trusted because not directly under our control; dark green boxes are the actors we can fully trust; while light green boxes those that we can partly trust. We see how in the extreme case where a customized OS is installed on the devices, with a dedicated app-store, dedicated

apps and even a dedicated mobile network, what is left as potential untrusted actors besides external attackers are the manufactures and the users. The reason why the manufacturer box in a custom solution is colored both red and green is that manufacturers of some components may be trusted, but it is very difficult to control absolutely all components a mobile device is composed of.

OVERVIEW OF DIFFERENT OWNERSHIP MODELS				
	USER OWNERSHIP	SHARED OWNERSHIP	ENTERPRISE OWNERSHIP	CUSTOM SOLUTION
TRUST IN SECURITY MECHANISMS	Low	Low-Medium	Medium-High	High
SCOPE OF ENFORCED SECURITY	Single Application	Work Partition	Application Layer And Platform Configuration	Kernel And Above (possibly pre-boot)
ACQUISITION COST	Low	Low-Medium	Medium-High	High
MAINTENANCE COST	Low	Low-Medium	Medium-High	High
PORTABILITY	High	Medium-Low	Medium	Low

Table 4.1 Summary of the pros and cons of the different ownership models described in this chapter.

However, threats from external attackers could be mitigated by enforcing strict security policies that could eliminate many common attack vectors and reduce the attack surface. For instance: Internet access can be precluded and only proprietary encrypted local networks are used; no third-party apps are allowed to be installed besides those specifically developed and approved for the tasks at hand; no integrated services can monitor device activity and send potentially sensitive data to third-parties; mobile network is most likely not used, or a dedicated one managed by enterprise itself deployed. This leaves pretty much only supply chain attacks, insiders and targeted attacks as potential threats. On the other extreme instead, when we have

commercial devices completely owned by the user, any of the actors involved in the mobile platform can become a potential adversary against which is much more difficult to protect. Therefore, most common attack vectors which could be excluded in a controlled environment are still present.

HOW OWNERSHIP MODELS AFFECT SECURITY				
	User ownership	Shared ownership	Enterprise ownership	Enterprise ownership + custom OS
Administrator	User	Defence/User	Defence	Defence
Developers	Third party	Defence/ Third party	Defence	Defence
Marketplace provider	Third party	Defence/ Third party	Defence	Defence
Mobile operator	Third party	Third party	Defence/ Third party	Defence
Platform provider	Third party	Third party	Third party	Defence
Manufacturer	Third party	Third party	Third party	Third party
Users	Civilians	Civilians/Military personnel	Military personnel	Military personnel

Table 4.2 Assuming the Department of Defence as the enterprise that needs to trust the device, we can see how the more aspects of the platform are under its control, the more actors coincide with the enterprise itself. This leads to more consistent system and fewer adversaries to defend against.

Thus, commercial solutions can probably be made good enough for typical office use like e-mail and access to the enterprise network. In any case situations where sensitive information at a classification level not higher than RESTRICTED is to be handled, given that a sufficient level of control can be exercised. Management solutions are therefore necessary in order to correctly configure and manage the devices so that the possible attack vectors are reduced to a minimum.

Adequate infrastructure must also be in place, as indicated for instance in the Mobile Security Reference Architecture by the CIO Council⁶⁹. Approval for even higher classifications however, requires either high-assurance solutions which are not found in COTS products, or very strict policies like that all data is to be only stored in volatile memory, external secure elements must be used for authentication and key storage and only private encrypted local networks are to be used.

When it comes to tactical use, we have seen how some projects are trying to adapt commercial technology to more military-specific needs. A common trend seems to be the use of a hardened version of the Android OS installed on COTS hardware. This choice is quite natural since an open system is necessary in order to implement the additional capabilities required by a military tactical setting. However, this reduces significantly some of the benefits of using commercial products in the first place. Most of the security of COTS devices comes from a tight integration between hardware capabilities and software mechanisms that may be lost once the original OS is replaced with a customized one. For instance, in order to install a new OS, one would have to root the device and unlock the boot-loader, and therefore lose any type of device integrity verification. As a consequence, most security extensions like hardware-protected keys installed at manufacturing time, may become unavailable. The new OS would also have to be maintained by someone else than the device manufacturer, so that new drivers would have to be developed each time a new model comes to the market, and security of the custom components would depend entirely on those who developed them, rather than, for instance, Google or Apple, and the millions of users that test them every day. Resources needed for configuration, provisioning, and maintenance would also increase. Some of these problems seem to be avoidable in some cases, as big manufacturers like Samsung appears to be willing to lock the custom platform to their devices at manufacturing time [53], so that rooting would not be necessary, but they would not provide updates and service. This would also result in a vendor-lock. Nevertheless, in order to achieve approval for higher classification levels, such modifications are probably unavoidable, and it may still be more convenient to use commercial technology as a starting point rather than designing and developing a completely new device from scratch. Additionally one of the other main advantages of commercial smart technology are still preserved, namely: familiar interfaces which could significantly reduce training time and increase effectiveness of the solutions; faster development of new services; more flexibility; low hardware costs; and short time-to-market of the latest available technology.

⁶⁹ <https://cio.gov/wp-content/uploads/downloads/2013/05/Mobile-Security-Reference-Architecture.pdf>

5 Risk mitigations and challenges – scenario analysis

Despite all the security efforts, it is also clear that there are limitations to what a commercial mobile device can protect. Unlike stationary equipment, their mobility makes them more vulnerable to physical attacks as they are easier to be lost, stolen or left unattended. Being mostly based on technology that is publicly available, they can also be subject to reverse engineering or other types of analysis that can uncover new vulnerabilities. Besides, it is not always feasible to adopt the most restrictive and expensive approaches, so that only medium level security may be available. Therefore, as a further protective measure, it is important to understand what kind of data it is reasonable to let these devices handle. For instance, if we use a BYOD approach, where the user owns the device, we may want to be extra careful about letting sensitive data be stored on the device. Tightly controlled devices used for tactical purposes may instead be able to handle higher classified data, but only as long as it does not require long term protection. How to evaluate which information is critical or harmless, how long it is safe to store it, how to handle it, and so forth, is a case-by-case evaluation and depends on the specific scenario. Nevertheless, we argue that there are some types of data and activities that are particularly suited to be handled by mobile devices, without necessarily requiring high-grade security. There may be also other practical obstacles to the deployment of a solution based on commercial devices in a military setting, like for instance the difficulty of connecting systems with different classification levels. Here, we briefly consider a possible scenario to illustrate these issues and suggest some possible solutions. This chapter is based on a previous publication [2], but it elaborates the issues further in the context of this report.

5.1 The Common Operational Picture scenario

One thing mobile devices can do pretty well, is data collection. As mentioned earlier, they are in practice a platform packed with sensors, and the intuitive interfaces let users easily take pictures and videos, write messages, or record voice, while services in the background can collect a whole other spectrum of information like location, temperature, radio signals and much more. If we have a team of soldiers equipped with these devices, the individual observations can be sent to some central server that aggregates them into what is commonly referred to as a “Common Operational Picture” (COP) together with other information coming from additional source and intelligence. The problem is that while the data collected from a single device is not necessarily sensitive itself, the COP on the server can quickly become classified as it can potentially expose the overall capabilities, positions and movements of the whole unit. So, although we might argue that a commercial mobile device can be safely used to report some observations, there are other practical problems to consider. Figure 5.1 summarizes the scenario.

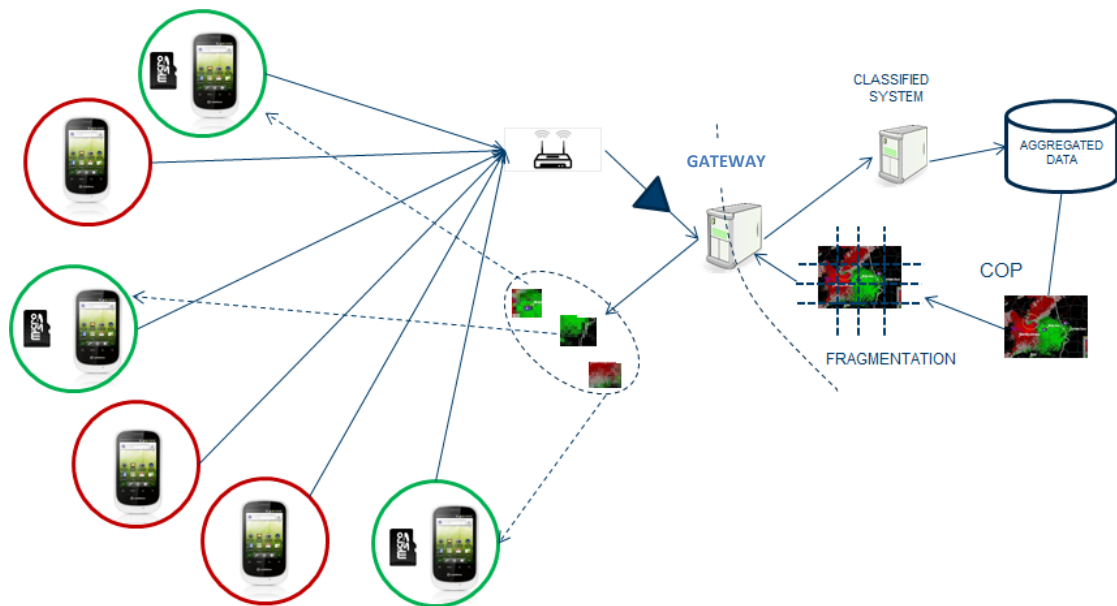


Figure 5.1 An approved gateway could allow information exchange between different security levels. Additionally, mobile devices may be discriminated based on their reliability and be given access to different information.

5.1.1 Initial risk assessment

First of all, let us give an example of how one may assess the risk of using mobile devices in this situation. In this case we consider commercial mobile devices “secure enough” for the task because we record observations that consist mostly of information that would be available to anyone else in the close vicinity or of public domain, and useful only in the short term. So, given that the devices can provide a secure communication channel, an attacker would have to gain physical access and “steal” the information right after it was collected in order to have any use for it. However, at that point it would be easier to just collect the same information instead of stealing it, making the security of the device sufficient to deter an attacker, and the potential loss of information acceptable.

5.1.2 Local aggregation and cryptographic material

While it can be argued that in most cases a single observation is not particularly sensitive, at least in the short term, if the device stores all observations made over a longer period of time, this might no longer be the case. For instance, in an aggregated form, the stored observations could indirectly reveal patterns and trends about the modus operandi of the user, and this could give an advantage to the attacker also in the near future. Since this information requires long-term protection, the device protection would become insufficient as more time-consuming attacks become feasible. In this case the problem could be mitigated simply by requiring that the observations were wiped away from the device right after they were reported to the server and the reception was acknowledged. A reliable way of doing so, however, could prove to be tricky in practice [2]. Similarly, the cryptographic keys used to establish a secure connection to the

server and possibly authenticate the user, should not be permanently stored on the phone because of similar security concerns. A possible mitigation we proposed previously [2] is to adopt smart-cards to provide tamper-proof key storage and strong authentication, which commercial mobile devices usually lack. We called this approach *ephemeral classification* to emphasize the short-lived nature of the sensitive data handled by the mobile device.

5.1.3 Reliability

The reliability of the observations themselves is dependent on the degree of trust we have in the device and the user. A compromised device can potentially report falsified or forged data. In a heterogeneous environment we could think of assigning different degrees of trust to different devices, based on some reliable parameters. Remote attestation could be one, but a simpler one, although not as reliable, could be the type of authentication. One could assign a higher degree of trust to those devices using stronger authentication. The underlying idea is that we can think that someone using a smart-card has also a somewhat more secure and reliable device (and training), than someone using, for instance, user-name and password or anonymous access. This could be the case in an emergency or disaster scenarios where military and civilians could cooperate to gather as much data as possible on the current status of the situation just by downloading an app and sending in whatever data they can collect [4, 54], but using different authentication methods based on the equipment they have.

5.1.4 Classified server

As mentioned at the beginning, the COP will most likely have to be stored on a server approved to handle classified information. Thus, if we wish to use commercial devices not approved to handle data of at least the same classification level, it may not be possible to connect them together to report the observations at all. The problem of connecting systems with different classification levels is a very current one, and not limited to the adoption of COTS products. We discuss this in detail in [2], but in general there are possibilities to allow a data flow from a lower classified system to a higher classified one, so that no data can leak from high to low. This however means also that it is very difficult to send any meaningful response to the device to confirm the successful reception of an observation, or maybe even a successful authentication. A gateway that ensures the separation between the security domains, while offering required functionalities like authentication and error handling, is one of the possible solutions that are under consideration [55]. Furthermore, there is always the risk that some malicious agent could find its way to the classified server through a less secure device and do some damages. Although information would not leak from it, its availability or integrity could still be compromised.

5.1.5 Information sharing

Even if we could successfully report all the observations to a classified server from a low classified device in a secure and reliable way, the resulting aggregated information would still be classified. Hence, it would not be possible to share it with the soldiers in the field if they were to use the same mobile devices. As it would not make much sense to have an additional classified device in the field (which could then be used for everything else instead), a possible strategy could be to share only parts of the COP with each device/soldier, based on their current position and information need. This could suffice to guarantee that the loss or compromise of one or few devices would not compromise the whole team or operation. The level or amount of information could also depend on the reliability of the device as described in Section 5.1.3. If more thorough declassification is needed, a manual solution may be adopted. In any case, additional control mechanisms might have to be in place to avoid that other classified information could leak from the server together with the declassified data.

5.1.6 Off-line

Although we argued that wiping of sensitive information could be a practical solution to minimize the risk of information loss, there may be situations when this approach is not possible, or might have to be delayed. Typically, if we do not have connectivity and cannot report to the server, we would save the data until we were back on-line. Similarly, if we received data from the server, we would like to store them for later use and avoid downloading them again, especially in situation with intermittent connectivity. This means that data cannot be kept in volatile memory only, and must be written to disk. This poses a significant threat if one were to lose the device, therefore strong encryption is the most intuitive solution here. However, full-disk encryption offered by most mobile devices may not work well for this particular application, and an ad-hoc solution, where keys were generated and stored on the smart-card, would be more suitable.

5.1.7 Civilian-Military cooperation

In a related scenario where the military may need to coordinate with civilian actors like police, fire-fighters or others, commercial mobile phones could be an easy and cheap solution to establish a common data and voice channel without the usual interoperability problems between proprietary systems. Given that in these situations only sensitive, but not highly classified information is shared and that most of it is generated on the fly and it is not retrieved from proprietary classified servers, standard security offered by commercial smartphones may suffice and we would not have many of the other problems mentioned in the previous sections. Dedicated servers managed by some trusted third party could provide the necessary infrastructure to swiftly set-up mission-specific profiles, so that anyone who can download the app and has a registered identity on the server can take part in the mission even with their private, but preferably managed, devices.

5.2 Discussion

In order to assess whether a given mobile device provides adequate security in a given scenario, we need to understand what type of information will be generated, consumed and exchanged in that situation and what kind of protection it needs. In general, commercial mobile devices can provide good short term protection, especially if used in a controlled environment, while in the long term they are more likely to be compromised. Therefore, we believe that they may be a viable choice in situations where information is not only unclassified, but also classified for a short-time or only in aggregated form. An example is data collected or generated by a user that is sensitive only in the current situation and is to be used immediately. Device or data loss would then not provide any particular advantage to an adversary, as he or she would not have time to act on it. Also information that may be classified could be fragmented so to still be useful to the individual users, but not particularly critical if lost. Of course, this is valid given the assumption that the device is not already compromised so that an attacker has real-time access to any information on it, but he/she needs to use a certain amount of time and resources to break its security.

So, even though commercial mobile devices may not provide high security or reliability in all situations, there are mitigations that could be put in place to allow their use in many interesting situations where they can provide an increased operational effect. Still, many technical challenges may arise in their actual integration and deployment in a real military setting, and possible solutions should be investigated.

6 Conclusions

Manufacturers and platform providers are increasingly interested in making mobile devices secure, because mobile devices are being used for security critical tasks like payments, remote management of houses and cars, and as identity tokens, and because corporate customers want secure and manageable devices to solve the BYOD problem. In addition, in order to appeal also to governmental and military actors, many mobile platforms seek also to be certified for standards like FIPS and Common Criteria, as shown by the NSA CSfC list⁷⁰. BlackBerry 10⁷¹ and a new phone born from the collaboration between Sectra and Samsung⁷² seem to have even been approved for use at RESTRICTED level, proving that commercial smartphones can in fact be made quite secure. This appears to be in contrast with the continuous stream of news about the increasing amount of newly discovered vulnerabilities and malicious applications in mobile devices. However, in this report we have seen how the presence of a vulnerability does not

⁷⁰ https://www.nsa.gov/ia/programs/csfc_program/component_list.shtml

⁷¹ <http://press.blackberry.com/press/2013/blackberry-10-receives-nato-approval-for-restricted-communicatio.html>

⁷² <http://communications.sectra.com/security-solutions/tigers-r>

directly imply its exploitability, as there are many security mechanisms in place to mitigate or prevent potential attacks. An analysis of various mobile security reports further confirms that devices that have not been tampered with and with the recommended security settings enabled are much less exposed to external threats. Globally, only around 0.1% of such Android devices have been found to have some kind of potential harmful application on them. The few iOS infections found on non-jailbroken devices can also be partly blamed on users ignoring security warnings.

This does not mean that commercial mobile devices should be trusted with handling high-grade military information out-of-the-box. The risk for a new vulnerability being discovered and exploited despite the existing security is always present, and users can intentionally or by mistake compromise the security of the device. Management tools can help achieving better control, but at the cost of the user's freedom. These tools are also not more secure than the platform on which they run, so they are rather a way to ease security management and increase the chance that policies are properly defined and enforced. The common vulnerabilities and attack vectors we have uncovered in this report are also taken from a consumer market setting, and while most of them could be avoided by stricter security policies, it is not clear what kind of new targeted attacks could emerge in a different setting like a military one and whether the same mitigations would be just as effective. This is however a problem that is common to any commercial product, not just mobile devices.

Still, by carefully analyzing the type of information one can expect to generate, receive and consume on these devices, it is possible to design "ad-hoc" solutions that can reduce the risk of sensitive information loss to an acceptable level. Thus, commercial devices can indeed be securely used in some scenarios where the benefits brought by their greater flexibility, ease of use and lower cost, outweigh the potential security risks. Example may be emergency or crisis situations where civil-military cooperation is needed; simple reporting of individual observations or non-security critical messages; local communication and information distribution in a closed network; and so on.

On the other hand, if such devices are to be used in mission critical and rough environments, where neither mobile nor Wi-Fi connectivity is available, with opponents that have advanced cyber-attack or electronic warfare capabilities and where reliability and confidentiality are critical, dedicated solutions would have to be adopted. In Chapter 4.2.3 we discussed this approach concluding that although we should talk of military devices built on commercial technology rather COTS products, using commercial technology as a foundation still gives considerable advantages. Supply-chain attacks, physical access and low reliability should however still be taken into account when evaluating the actual risk of using these devices.

Concluding, modern mobile devices can realistically be used in some situations where dedicated military equipment is not a desirable alternative because of the high costs, specialized expertise or clearance required, or interoperability issues. In these cases, having readily available and easy to use equipment that allows communicating and sharing information in a somewhat controlled way, is preferable to not having any means of communication at all or let users find even more insecure ways of exchanging information. Many projects are already looking at mobile apps as

the new way to bring innovative services to soldiers, and this trend is likely to grow. Thus, we should rather focus on how to support this development by adapting our security mindset to the new technology, rather than excluding the technology because it does not fit in our current security practices.

Appendix

A Secure mechanisms

Android (5.0-6.0)	iOS (9+)	BlackBerry 10	Windows 10
<i>INTEGRITY PROTECTION AT BOOT TIME</i>			
<p>Android supports verified boot through the optional device-mapper-verity (dm-verity) kernel feature⁷³. The device block integrity check is performed by the kernel comparing the calculated hash of each block to the stored reference hash tree. However, the device must implement a trusted boot that verifies the integrity of the kernel first [56]. Therefore it should be supported by a RSA key burned in the device.</p>	<p>From [18]: “Each step of the startup process contains components that are cryptographically signed by Apple to ensure integrity and that proceed only after verifying the chain of trust. This includes the bootloaders, kernel, kernel extensions, and baseband firmware.</p>	<p>At boot time the signatures of all loaded components, included the boot loader and the OS, are verified with a public certificate installed in the processor [21].</p>	<p>Windows phone must comply with UEFI specifications and implement a trusted boot, that includes a firmware TPM running in TrustZone [22]. Most likely as specified in [57].</p>
<i>SANDBOXING AND ISOLATION</i>			
<p>Android uses a Unix-style user separation of processes and file permissions, where each application is given a unique UID and runs in a separate process. s. All of the software above the</p>	<p>From [19]:”... each application is contained within its own unique directory on the filesystem and separation is maintained by the XNU Sandbox kernel extension... for an</p>	<p>“App sandboxing is primarily enforced through a combination of user and group filesystem permissions, separate operating system users and associated groups for each app, and PF</p>	<p>Apps are installed in a “Least Privilege Chamber” (LPC) allowing them to use only the capabilities defined in the “manifest” the app comes with. New privileges can be</p>

⁷³ <https://source.android.com/security/verifiedboot/index.html>

kernel (see Figure xx), including operating system libraries, application framework, application runtime, and all applications run within the Application Sandbox ⁷⁴ .	application to access things like media, the microphone, and the address book, it must request the relevant permissions from the user.”	firewall rules” [19]. Each app has a unique group ID, and its sandbox is defined by both this ID and the specific location in which it runs (work or personal space).	granted only by updating the app [22].
---	---	---	--

SECURE APPLICATION PROVISIONING

Android developers must register in order to be able to publish on the Google Play store, but no identity check is performed besides the credit card payment. Apps can be signed with self-signed certificates, but need to go through the Google Bouncer system where they are automatically checked for malware. However, apps from third party app-stores can be installed on the devices given the user consent. In this case, no security evaluation is guaranteed. Signatures are used to make sure that only apps signed with the same certificate can	Apple requires developers to register to their iOS Developer Program in order to issue a certificate that will be used to sign their apps. Developers are to be identified personally. Apps are reviewed by Apple to ensure they operate as described and don't contain obvious bugs or other problems. Companies can obtain special certificates to create in-house apps that can be installed through an MDM, bypassing the appstore [18]. Apps not signed by Apple can be installed only on jailbroken devices.	Applications for BlackBerry 10 are solely distributed via BlackBerry World [19]. Like for Apple, BlackBerry developers must enroll and get a certificate with which they sign their apps. BlackBerry states that apps submitted to be published in their appstore will be “aggressively scrutinized” and give some requisites they should satisfy ⁷⁵ . Keys, however, are stored by BlackBerry, so that it actually has a copy of the developers' keys ⁷⁶ .	Microsoft requires a developer account in order to submit apps for revision before publications. From their guidelines ⁷⁷ , it appears that apps are signed directly by Microsoft before being published on the app store.
--	---	--	---

⁷⁴ <https://source.android.com/security/overview/kernel-security.html#the-application-sandbox>

⁷⁵ <https://developer.blackberry.com/builtforblackberry/documentation/criteria/security.html>

⁷⁶ <http://devblog.blackberry.com/2013/08/code-signing-keys-be-gone-welcome-blackberry-id/>

⁷⁷ <https://msdn.microsoft.com/library/windows/apps/mt148554.aspx>

<p>communicate and be updated.</p> <p>System applications are signed with platform keys and are allowed to run with system privileges and share resources [56].</p>	<p>Signatures are also checked every time an app or some code is run [19].</p>		
<p><i>SECURE STORAGE OF SENSITIVE DATA</i></p>			
<p>Android offers full disk encryption by default from Android 5.0. It creates a 128-bit AES master encryption key the first time the device is booted, which is in turn encrypted with the user password or Pin and possibly signed by a private key. The resulting encryption key is then used by dm-crypt to encrypt everything that is written to disk.</p> <p>The key is protected in hardware by the TEE where this is available⁷⁸.</p>	<p>All iOS devices contain two unique AES keys stored in the dedicated cryptographic coprocessor (The Secure Enclave), which are used to generate other encryption keys, like the File System Key, created the first time the device is booted, which is used to decrypt the partition table and the system partition at boot time. In addition a “per-file” key is created through the Data Protection API from the user passcode and used to encrypt new files as they are written to disk [18] [19].</p>	<p>Blackberry encrypts both work, personal and media storage data with a key hierarchy rooted in a device-key embedded in the processor when the processor is manufactured. For each file a random AES-256 key is generated, which is encrypted with the domain key, in turn encrypted with the work or personal master key, which is stored in NVRAM and encrypted with the system master key stored in the replay protected memory block on the device, which is encrypted with the device-key [20].</p>	<p>Windows uses BitLocker to provide full-disk encryption, backed-up by a firmware TPM implemented in TrustZone. Keys are stored in the device eFuses⁷⁹. Company data are additionally encrypted using Enterprise Data Protection (EDP) [22].</p>

⁷⁸ <https://source.android.com/security/encryption/index.html>

⁷⁹ <https://businessmobilitycenter.microsoft.com/en/webinars/Pages/Webinar-Security-for-Lumia-with-Windows-Phone-8.aspx>

<i>CENTRALIZED AND FREQUENT SECURITY UPDATES</i>			
<p>Google maintains the Android code base and releases security updates, but each device manufacturer running a customized Android must release their own versions. Google promised however to increase the frequency of the updates to once a month⁸⁰, but device manufacturer and network operators will have to follow up.</p>	<p>Apple has full control on both device firmware and software, so they can push any update on all Apple devices whenever needed and centrally. They can also prevent downgrades that can be used to exploit old vulnerabilities by using a “System Authorization Software” [18].</p>	<p>BlackBerry has also control over all manufacturing process from hardware to applications and can centrally update all its devices.</p>	<p>Although originally shipped only on Nokia’s phones, Windows mobile opened to different OEM manufacturers last year⁸¹, potentially creating the same fragmentation problem Android has. However, with Windows 10 mobile, they intend to centralize the distribution of security updates despite different manufacturer and carriers⁸².</p>
<i>BYOD AND MDM</i>			
<p>Android support for MDM software has been usually limited to enforcing password policy, device wipe and encryption until the introduction of Android for Work with the Lollipop edition. Now a device can be partitioned into a personal and work space, where the work space owner has a much wider range of security policies they can create [15].</p>	<p>Apple gives also the possibility to configure a device for different scenarios like User-owned, Organization owned personally enabled, and Organization owned non-personalized. It offers wide support to MDM software and integration with Apple services [58].</p>	<p>BlackBerry Balance is the BlackBerry BYOD solution. It allows partitioning the device in a personal and work space, each encrypted with its own key. Again, the device can be configured for three different situations: Work and personal – Corporate (BYOD), Work and personal – Regulated, Work space only. [21]</p>	<p>Windows 10 offers also separation between personal and work data and apps with separate encryption and dedicated app stores for each. MDM support is also quite extensive and allow for wide range of policy options [22].</p>

⁸⁰ <http://officialandroid.blogspot.no/2015/08/an-update-to-nexus-devices.html>

⁸¹ http://www.engadget.com/2014/02/23/microsoft-lg-lenovo-windows-phone/?ncid=rss_truncated&utm_campaign=sf

⁸² <http://www.cnet.com/news/microsoft-to-control-software-updates-for-windows-10-mobile/>

B Abbreviations

AOSP	Android Open Source Project	NFC	Near Field Communication
API	A Platform Interface	NS	Non-Secure
BYOD	Bring Your Own Device	NSA	National Security Agency
C2IS	Command and Control Information System	NX	Non eXecutable
CIO	Chief Information Officers	OEM	Original Equipment Manufacturer
COP	Common Operational Picture	OS	Operating System
COTS	Commercial off the Shelf	OWASP	Open Web Application Security Project
CSfC	Commercial Solutions for Classified	PHA	Potential Harmful Application
CTS	Compatibility Test Suite	PKI	Public Key Infrastructure
CVE	Common Vulnerabilities and Exposures	PLE	Payload Encryption
CVSS	Common Vulnerability Scoring System	RTOS	Real-time Operating System
DAC	Discrete Access Control	SELinux	Security Enhanced Linux
DISA	Defence Information System Agency	SIM	Subscriber Identity Module
DMCC-S	Defense Mobile Classified Capability-Secret	SIPRNet	Secret Internet Protocol Router Network
DRM	Digital Rights Management	SMS	Short Message Service
EAL	Evaluation Assurance Level	SoC	System on Chip
EMM	Enterprise Mobile Management	SQL	Structured Query Language
FIPS	Federal Information Processing Standards	SSL	Secure Socket Layer

HTTPS	Hyper Text Transfer Protocol Secure	TEE	Trusted Execution Environment
IPC	Inter-process Communication	TLS	Transport Layer Security
IT	Information Technology	TPM	Trusted Platform Module
LTE	Long Term Evolution	UEFI	Unified Extensible Firmware Interface
MAC	Mandatory Access Control	USB	Universal Serial Bus
MAM	Mobile Application Management	VOIP	Voice Over IP
MDM	Mobile Device Management	VPN	Virtual Private Network
MIM	Mobile Information Management	XML	eXtensible Markup Language
MMS	Multimedia Messaging Service	XN	eXecutable Never
NDA	Non-Disclosure Agreement		

References

- [1] E. Demko, "Commercial-off-the shelf (COTS): a challenge to military equipment reliability," in *Proceedings of RAMS'96 - The Annual Reliability and Maintainability Symposium*, 1996.
- [2] F. Mancini and A. Fongen, "Ephemeral classification of mobile terminals," in *IMCIS'15 - International Conference on Military Communications and Information Systems, Cracow*, 2015.
- [3] J. H. Roa, D. E. Aubert, E. N. Eriksen, K. R. Karud, F. T. Johnsen, T. H. Bloebaum, M. R. Brannsten og B. K. Reitan, «KingsEye - plattformuavhengig situasjonsoversikt- FFI notat 2015/01718,» Norwegian Defence Research Establishment (FFI), Kjeller, 2015.
- [4] J. Hagen and H. Hafnor, "Med krisekommunikasjon i lomma - FFI report 2015/0196,"

Norwegian Defence Research Establishment, Kjeller, 2105.

- [5] B. K. Reitan, A.-K. Elstad and C. J. Gran, "En ny klasse kommando og kontroll informasjonssystemer - eksperimenter med smarttelefoner og samhandling - FFI report 2015/02298," Norwegian Defence Research Establishment (FFI), Kjeller, 2015.
- [6] K. Nomeland, "Teknisk målarkitektur og veikart for taktisk radio i Forsvaret," FLO/IKT/Nettverksavdeling/Radioseksjon, Kolsås, 2015.
- [7] Forsvarets sikkerhetstjeneste, «Sikkerhetskonsept for et nettverksbasert forsvar,» Forsvaret, 2011.
- [8] N. A. Nordbotten, F. Mancini, B. H. Farsund, R. Haakseth, A. M. Hegland og F. Lillevold, «Information sharing across security domains,» Norwegian Defence Research Establishment (FFI), Kjeller, NO, 2015.
- [9] N. Asokan, L. Davi, A. Dmitrienko, S. Heuser, K. Kostainen, E. Reshetova and A.-R. Sadeghi, *Mobile Platform Security*, Morgan & Claypool Publishers, 2014.
- [10] ETSI, *TS 131 111 - V11.5.0 - Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Universal Subscriber Identity Module (USIM) Application Toolkit (USAT) (3GPP TS 31.111 version 11.5.0 Release 11)*, CEDEX: ETSI, 2013.
- [11] ARM, "ARM Security Technology - Building a Secure System using TrustZone® Technology," April 2009. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf. [Accessed 16 December 2015].
- [12] F. Mancini, "Hardware-based trust and integrity: Trusted Platform Module (TPM) and Trusted Execution Environment (TEE) – possible building blocks for more secure systems?," FFI, Kjeller, 2014.
- [13] T. Alves and D. Felton, "TrustZone: Integrated Hardware and Software Security - Enabling Trusted Computing in Embedded Systems," *Information Quarterly*, vol. 3, no. 4, pp. 18-24, 2004.
- [14] S. Smalley and R. Craig, "Security Enhanced (SE) Android: Bringing Flexible MAC to Android," in *NDSS '13 - 20th Annual Network and Distributed System Security Symposium*, San Diego, CA United States, 2013.

-
-
- [15] Google, "Android for Work Security white paper," May 2015. [Online]. Available: <https://static.googleusercontent.com/media/www.google.com/no/NO/work/android/files/android-for-work-security-white-paper.pdf>. [Accessed 15 December 2015].
- [16] Samsung Electronics Co., Ltd, "White Paper: An Overview of the Samsung KNOX™ Platform," September 2015. [Online]. Available: https://www.samsungknox.com/en/system/files/whitepaper/files/An%20Overview%20of%20the%20Samsung%20KNOX%20Platform_V1.12_0.pdf. [Accessed December 2015].
- [17] Samsung Electronics Co., Ltd. , "In-Depth Look at Capabilities: Samsung KNOX and Android for Work," 2015. [Online]. Available: https://www.samsungknox.com/en/system/files/whitepaper/files/Samsung%20KNOX%20and%20Android%20for%20Work_2.pdf. [Accessed December 2015].
- [18] Apple, "iOS Security - iOS 9.0 or later," September 2015. [Online]. Available: https://www.apple.com/business/docs/iOS_Security_Guide.pdf. [Accessed 14 December 2015].
- [19] D. Chell, T. Erasmus, S. Colley and O. Whitehouse, *The Mobile Application Hacker's Handbook*, John Wiley & Sons, 2015.
- [20] BlackBerry, "BlackBerry Enterprise Service 10 - Security Technical Overview - BlackBerry Device Service Solution Version: 10.2," 10 September 2014. [Online]. Available: https://help.blackberry.com/nb/bes10/10.2/sto-pdf/BES10_v10.2_BDS_Security_Technical_Overview_en.pdf. [Accessed 15 December 2015].
- [21] BlackBerry, "Security Overview - BlackBerry 10," September 2015. [Online]. Available: <https://help.blackberry.com/en/blackberry-security-overview/latest/blackberry-security-overview-pdf/BlackBerry-10-latest-Security-Overview-en.pdf>. [Accessed 14 December 2015].
- [22] A. Meeus, "Windows 10 for mobile devices Secure by design," in *Microsoft Ignite*, Chicago, 2015.
- [23] TCG, "TCG Specification - TPM 2.0 Mobile Reference Architecture. Level 00 Revision 142," 16 December 2014. [Online]. [Accessed 16 December 2015].
- [24] R. Kissel, «NISTIR 7298 Revision 2 - Glossary of Key Information Security Terms,» NIST, Gaithersburg, MD, US, 2013.

-
- [25] M. Howard, J. Pincus og J. M. Wing, «Measuring Relative Attack Surfaces,» i *Computer Security in the 21st Centur*, Boston, MA, US, 2005.
- [26] NSA, "New Smartphones and the Risk Picture," April 2012. [Online]. Available: https://www.nsa.gov/ia/_files/factsheets/mobilerisks.pdf. [Accessed January 2016].
- [27] Google Inc., "Google Report - Android Security 2014 Year in Review," 2015. [Online]. Available: https://static.googleusercontent.com/media/source.android.com/en//devices/tech/security/reports/Google_Android_Security_2014_Report_Final.pdf. [Accessed Januar 2016].
- [28] R.-P. Weinmann, "Baseband Attacks: Remote Exploitation of Memory Corruptions in CellularProtocol Stacks," in *WOOT'12 - The 6th USENIX Workshop on Offensive Technologies*, Bellevue, WA, 2012.
- [29] F-Secure Labs, "MOBILE THREAT REPORT Q1 2014," April 2014. [Online]. Available: https://www.f-secure.com/documents/996508/1030743/Mobile_Threat_Report_Q1_2014.pdf. [Accessed January 2016].
- [30] Symantec, "Internet security threat report 2013," April 2013. [Online]. Available: http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v18_2012_21291018.en-us.pdf. [Accessed January 2016].
- [31] 360 Security, "An overview of malware and vulnerability landscape in 2nd quarter 2015," 28 July 2015. [Online]. Available: <file:///C:/Users/fma/Dropbox/mobile%20security%20docs/Q2%202015%20Android%20Malware%20and%20Vulnerability%20Report%20-.html>. [Accessed Januar 2016].
- [32] Kaspersky, "Kaspersky Security Bullettin 2014," 2015. [Online]. Available: <https://securelist.com/files/2014/12/Kaspersky-Security-Bulletin-2014-EN.pdf>. [Accessed January 2016].
- [33] Lookout, "2014 Mobile Threat Report," 2015. [Online]. Available: https://www.lookout.com/img/images/Consumer_Threat_Report_Final_ENGLISH_1.14.pdf. [Accessed Januar 2016].
- [34] FireEye Inc., "OUT OF POCKET: A Comprehensive Mobile Threat Assessment of 7 Million iOS and Android Apps," February 2015. [Online]. Available: <https://www2.fireeye.com/rs/fireeye/images/rpt-mobile-threat-assessment.pdf>. [Accessed Januar 2016].

-
-
- [35] Alcatel-Lucent, "Mobile malware: A network view," in *Black Hat Mobile Security Summit*, London, 2015.
- [36] T. Müller and M. Spreitzenbarth, "FROST - Forensic Recovery of Scrambled Telephones," in *ACNS 2013 - 11th International Conference on Applied Cryptography and Network Security*, Banff, AB, Canada, 2013.
- [37] T. Roth, "Next generation mobile rootkits," in *Hack in Paris 2013*, Paris, FR, 2013.
- [38] D. Shen, "Attacking your "Trusted Core" - Exploiting TrustZone on Android," in *Black Hat USA 15*, Las Vegas, NV, USA, 2015.
- [39] Y. Zhang, Z. Chen, H. Xue and T. Wei, "Fingerprints On Mobile Devices: Abusing and Leaking," in *Black Hat 15 USA*, Las Vegas, NV, USA, 2015.
- [40] Lookout, "Enterprise Mobile Threat Report - The State of iOS and Android Security Threats to Enterprise Mobility," 2015. [Online]. Available: https://www.lookout.com/docs/us/Enterprise_MTR.pdf. [Accessed Januar 2016].
- [41] C. Xiao, "WIRELURKER: A New Era in iOS and OS X Malware," November 2014. [Online]. Available: https://www.paloaltonetworks.com/content/dam/paloaltonetworks-com/en_US/assets/pdf/reports/Unit_42/unit42-wirelurker.pdf. [Accessed January 2016].
- [42] T. Wei, M. Zheng, H. Xue and D. Song, "Apple Without A Shell – ios Under Targeted Attack," in *Virus Bulletin Conference*, Seattle, USA, 2014.
- [43] B. Lau, Y. Jang og C. Song, «Mactans: Injecting Malware into iOS Devices via Malicious Chargers,» i *BalckHat USA 2013*, Las Vegas, NV, US, 2013.
- [44] A. Antukh, "BlackBerry 10 Research Primer - "Dissecting Blackberry 10 – An initial analysis", " SEC Consult Vulnerability Lab, Vienna, 2013.
- [45] D. M. Gomez and A. Davis, "BlackBerry PlayBook Security: Part one," NGS Secure , 2011.
- [46] G. Jones, "BlackBerry PlayBook Security: Part two," NGS Secure, 2011.
- [47] Z. Lanier and B. Nell, "No Apologies Required: Deconstructing Blackberry 10," in *CanSecWest 2014*, Vancouver, 2014.
- [48] R.-P. Weinmann, "BlackBerryOS 10 from a security perspective," in *Black Hat 2013 USA*, Las Vegas., 2013.

-
- [49] T. Cosgrove, R. Smith, C. Silva, J. Girard and B. Taylor, *Magic Quadrant for Enterprise Mobility Management Suites*, Gartner, 2015.
- [50] D. Brodie and M. Shaulov, "Practical Attacks against Mobile Device Management (MDM)," in *Black Hat 2013 Europe*, Amsterdam, NL, 2013.
- [51] T. Collyer, "Android, BYOD, and AirWatch MDM," SANS Institute, 2014.
- [52] TNO, "Promise 1.0 final report," Defence Materiel Organisation - Ministry of the Defence, The Netherlands, 2015.
- [53] F. T. Johnsen, "Reiserapport: Purple Nectar 2015," Norwegian Defence Research Establishment (FFI), Kjeller, NO, 2016.
- [54] J. Hagen, H. Hafnor, F. Mancini, A.-K. Elstad, B. Reitan, A.-L. Bjørnstad and C. Gran, "App teknologi som krisekommunikasjonsverktøy - FFI notat 2015/0280 Untatt Offentlighet," Norwegian Defence Research Establishment (FFI), Kjeller, 2015.
- [55] N. A. Nordbotten, F. Mancini, B. H. Farsund, R. Haakseth, A. M. Hegland and F. Lillevold, "Information sharing across security domains - FFI-report 2015/00456," Norwegian Defence Research Establishment (FFI), Kjeller, 2015.
- [56] N. Elenkov, *Android Security Internals - An In-Depth Guide to Android's Security Architecture*, San Francisco: No Starch Press, Inc., 2015.
- [57] TCG, "TCG PC Client Platform Firmware Profile Specification - Family "2.0", Level 00 Revision 00.21," 27 August 2015. [Online]. Available: https://www.trustedcomputinggroup.org/files/resource_files/71610DAD-1A4B-B294-D09C51583DB1A501/PC%20ClientSpecific_Platform_Profile_for_TPM_2p0_Systems_v21_Public%20Review.pdf. [Accessed 14 December 2015].
- [58] Apple, "iOS Deployment Overview for Enterprise," September 2015. [Online]. Available: https://www.apple.com/business/docs/iOS_Enterprise_Deployment_Overview.pdf. [Accessed 15 December 2015].
- [59] NIST, «FIPS PUB 200 - Minimum Security Requirements for Federal Information and Information Systems,» National Institute of Standards and Technology , Gaithersburg, MD, US, 2006.

About FFI

The Norwegian Defence Research Establishment (FFI) was founded 11th of April 1946. It is organised as an administrative agency subordinate to the Ministry of Defence.

FFI's MISSION

FFI is the prime institution responsible for defence related research in Norway. Its principal mission is to carry out research and development to meet the requirements of the Armed Forces. FFI has the role of chief adviser to the political and military leadership. In particular, the institute shall focus on aspects of the development in science and technology that can influence our security policy or defence planning.

FFI's VISION

FFI turns knowledge and ideas into an efficient defence.

FFI's CHARACTERISTICS

Creative, daring, broad-minded and responsible.

Om FFI

Forsvarets forskningsinstitutt ble etablert 11. april 1946. Instituttet er organisert som et forvaltningsorgan med særskilte fullmakter underlagt Forsvarsdepartementet.

FFIs FORMÅL

Forsvarets forskningsinstitutt er Forsvarets sentrale forskningsinstitusjon og har som formål å drive forskning og utvikling for Forsvarets behov. Videre er FFI rådgiver overfor Forsvarets strategiske ledelse. Spesielt skal instituttet følge opp trekk ved vitenskapelig og militærteknisk utvikling som kan påvirke forutsetningene for sikkerhetspolitikken eller forsvarsplanleggingen.

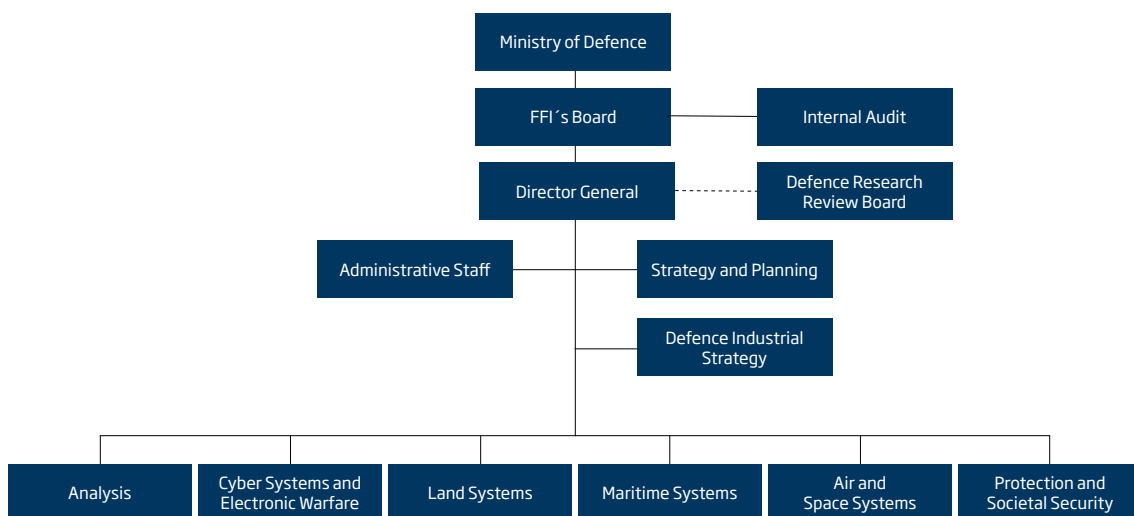
FFIs VISJON

FFI gjør kunnskap og ideer til et effektivt forsvar.

FFIs VERDIER

Skapende, drivende, vidsynt og ansvarlig.

FFI's organisation



Forsvarets forskningsinstitutt
Postboks 25
2027 Kjeller

Besøksadresse:
Instituttveien 20
2007 Kjeller

Telefon: 63 80 70 00
Telefaks: 63 80 71 15
Epost: ffi@ffi.no

Norwegian Defence Research Establishment (FFI)
P.O. Box 25
NO-2027 Kjeller

Office address:
Instituttveien 20
N-2007 Kjeller

Telephone: +47 63 80 70 00
Telefax: +47 63 80 71 15
Email: ffi@ffi.no