Security for Mobile Device Assets: A Survey

António Lima[1], Bruno Sousa[2], Tiago Cruz[1], Paulo Simões[1]
[1]Department of Informatics Engineering of the University of Coimbra, Coimbra, Portugal
[2]OneSource, Consultoria Informática Lda.
aclima@student.dei.uc.pt
bmsousa@onesource.pt
tjcruz@dei.uc.pt
psimoes@dei.uc.pt

Abstract: Organizations are often faced with the need to manage large numbers of mobile device assets, including tight control over aspects such as usage profiles, customization, applications and security. Moreover, the raise of the Bring Your Own Device (BYOD) paradigm has further contributed to hamper these requirements, making it difficult to strike a balance between corporate regulations and freedom of usage.

In this scope, security is one of the main requirements both for individual and corporate usage. Device and information protection on mobile ecosystems is quite different from securing other assets such as laptops or desktops, due to specific characteristics and restrictions. For instance, the resource consumption overhead of security mechanisms, which is less relevant for desktop/laptop environments, is critical for mobile devices which frequently have less computing power and must keep power consumption as low as possible.

Security mechanisms for mobile devices combine preventive tools (e.g. Trusted Execution Environments and sandboxed applications), monitoring solutions and reactive and mitigation techniques. In this paper we discuss these security solutions, presenting a survey on the technologies, frameworks and use cases for mobile device security monitoring and management, with an emphasis on the associated open challenges and benefits, from both the end-user and the corporate points-of-view.

Keywords: mobile devices, security, monitoring, management, detection, prevention

1. Introduction

Unlike other forms of computing (such as desktop or laptops or certain specialized embedded systems), mobile devices can be characterized by several specific traits such as dimensions and weight, connectivity, human-machine interface capabilities or autonomy. Moreover, usage models differ from traditional desktop computing, as interaction tends to occur over short time windows, rather than in a continuous manner.

Most modern mobile devices have an embedded display screen (or can be connected to one), receiving input from physical or virtual buttons and keyboards (using touch-sensitive displays in the latter case). Additionally, some devices also support audio input, namely voice recognition. Several types of sensors and data capture devices can also be embedded or attached to the mobile device. Typical examples include accelerometers, compasses, magnetometers and gyroscopes (allowing for detection of orientation and motion), as well as barcode, RFID, fingerprint and smart card readers. On top of all the already mentioned extras, the most prominently used features are related to connectivity and usually comprise Wi-Fi, Bluetooth, NFC (Near Field Communications) and GPS capabilities.

The first mobile devices didn't have much to offer. Devices like the IBM Simon (released in 1994, regarded as the first smartphone) were very "dumb" by today's standards, having very limited electronics and processing power. For the sake of context, Table 1 compares several representative mobile device models released between 1994 and 2016: an iPhone 6, a Samsung Galaxy S 1st Generation, an iPhone 1st Generation (commonly referred by many as one of the first examples of the current smartphone paradigm) and the IBM Simon.

Mobile devices (and, particularly, smartphones) have been gradually acquiring computing, communications and sensory capabilities at an exponential rate. For this reason, such devices have evolved beyond their natural role of mobile assistants, gradually assuming roles previously associated with traditional computing devices, such as desktop or laptop PCs. With the result of this sustained trend, mobile devices have become important tools both for individual users and organizations. Company-provided mobile devices became as commonplace as laptops, prompting the need for the development of asset management systems for these ecosystems.

Table 1 – Comparison of iconic mobile device (smartphone) specifications

	IBM Simon	iPhone (1st Gen)	Samsung Galaxy S	iPhone 6
Release Date	August 1994	June 2007	June 2010	September 2014
Dimensions	200 x 64 x 38 mm	115 x 61 x 11.6 mm	122.4 x 64.2 x (9.9 - 14) mm	138.1 x 67 x 6.9 mm
Weight	510 g	135 g	118 - 155 g	129 g
CPU	16-bit, 16 MHz, x86-compatible	32-bit, 412 to 620 MHz, ARM	1 GHz single-core, ARM	64-bit, 1.4 GHz dual-core, ARM
GPU	None	PowerVR MBX Lite 3D GPU	200 MHz PowerVR SGX540	PowerVR Series 6 GX6450 (quad-core)
Memory	1 MB	128 MB DRAM	512 MB RAM	1 GB LPDDR3 RAM
Storage	Internal: 1 MB	Internal: 4, 8, or 16 GB	**Internal:** 2 - 16 GB **Removable:** microSD up to 64 GB	Internal: 16, 64 or 128 GB
Display	4.5 in × 1.4 in 160px x 293px monochrome backlit LCD	3.5 in diagonal 320x480 px resolution at 163 ppi, 2:3 aspect ratio, 18-bit (262, 144-color) LCD	4.0 in diagonal Super AMOLED with RBGB-Matrix (Pentile) 480x800 px WVGA (233 ppi)	4.7 in diagonal 1334x750, LED-backlit IPS LCD, 326 ppi (128 px/cm) pixel density 16:9 aspect ratio
Data Inputs	Microphone Touchscreen with stylus	Multi-touch touchscreen 3-axis accelerometer, Proximity sensor, Ambient light sensor, Microphone	Multi-touch capacitive touchscreen display, Ambient light sensor, microphone, 3-axis Magnetometer (Compass), aGPS, 3-axis accelerometer	Multi-touch touchscreen, Triple microphone, 3-axis gyroscope, 3-axis accelerometer, Digital compass, iBeacon, Proximity sensor, Ambient light sensor, fingerprint reader, Barometer
Connectivity	2400-bps Hayes-compatible modem 33-pin connector 9600-bps group 3 fax, PCMCIA type 2 port	Quad-band GSM/GPRS/EDGE, Wi-Fi (802.11 b/g), Bluetooth 2.0	Dual band CDMA2000/EV-DO Rev. A, WiMAX, 2.5G (GSM/GPRS/EDGE), 3G, Wi-Fi (802.11b/g/n), DLNA, Bluetooth 3.0	UMTS/HSPA+/DC-HSDPA, CDMA EV-DO Rev. A and Rev. B, GSM/EDGE, Wi-Fi, Bluetooth 4.2, NFC, GPS & GLONASS
Camera	None	Rear: 2 MP	**Rear:** 5 MP, 720p, HD video, panorama **Front:** VGA	**Rear:** 8 MP, 1080p HD video recording, Slow-motion video, Panorama **Front:** 1.2 MP (1280×960 pixel max.), 720p video recording

Eventually, the consumer and enterprise perspectives on mobile device management hit a crossroad, with the emergence of the "Bring Your Own Device" (BYOD) paradigm, blending the enterprise and consumer device ecosystem. There are several types of BYOD schemes available for enterprise usage, with the key distinctive factor being Mobile Device Management (MDM) policies, detailing which devices are admissible and what type of control is held over them (accessible data, applications and functionalities). Figure 1 illustrates the trade-off between control and number of devices in the environment.

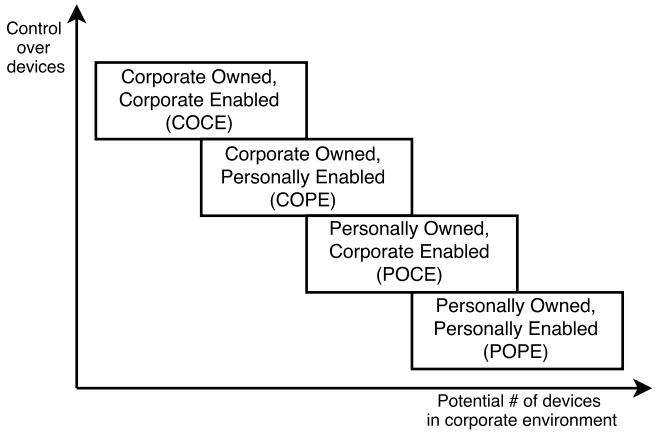

Figure 1 – Enterprise mobility schemes

BYOD constitutes a departure from "Corporate Owned Corporate Enabled" (COCE) models, moving towards schemes such as "Corporate Owned Personally Enabled" (COPE) and "Personally Owned Corporate Enabled" (POCE). Naturally, this has an impact in terms of security (which was one of the main concerns for delayed adoption of such policies at corporate level), as companies want to protect sensitive data and applications.

When it comes to security, mobile devices have very specific characteristics that separate them from other devices. The most noteworthy distinction is the limited battery life, inherent to mobile devices, as it is also the root of most of other differences. Since the battery is finite (unless recharged) the devices' computing power can only be pushed thus far, to the point where its continuous usage does not consume too much energy – battery life is also the reason why traditional agent-based security systems must be carefully designed and implemented, to minimize the overhead introduced by these agents. Compact form factors also pose concerns when it comes to cooling or increasing storage space. Another concern is the increased risk of misplacement or theft inherent to mobile devices, which grants a malicious user the freedom for a completely different suite of attacks or paths to the data stored in the device. With such accentuated differences, it is only inevitable that security assurance in mobile devices faces a different set of challenges from conventional computers' security risks and defence strategies, working with more limited resources.

This paper focuses on surveying mobile device management solutions for security problems, ranging from prevention to monitoring and mitigation of threats at various points in the system's and its applications lifecycles, identifying the best and worst practices for enterprise environments.

The rest of the paper is organized as follows. Section 2 delves into the aspects of Prevention in mobile devices, overviewing the most common used methods of each strategy, their benefits and shortcomings. Section 3 presents a similar analysis for mobile device security monitoring. Section 4 examines reactive and mitigation techniques employed in mobile devices. Section 5 concludes the paper with the final remarks.

2. Prevention

Currently there is an apparent emphasis on mobile device data security through preventive methods, when compared with monitoring or detection approaches. The most often cited cause for such discrepancy has to do with the less intrusive nature of prevention methods, which are also characterized by lower computation requirements and battery overhead. This section details the most common prevention methods found in mobile devices, along with their respective pros and cons.

The adoption of preventive methods is mostly driven by the need to protect devices from being compromised, by monitoring or minimizing the possibility of exploiting specific attack vectors. In the latter case this may correspond to the adoption of mechanisms or policies that reduce or eliminate the attack surface exposure. Most commonly mechanisms include:

- **Sandboxing** – this approach creates a separate virtual space for the untrusted application or code to run with limited to no interaction with other apps and under tight restrictions in the underlying host system, thus diminishing the possible attack vector. Most modern mobile device Operating Systems (OS), such as Android, iOS and Windows Phone, run their applications in sandboxed environments.
- **Personalized Approval** – this consists on a detailed analysis of each application's conformity to security policies before being approved for distribution. The most common example of this technique is that of Apple's App Review system wherein each submitted application goes through a series of meticulous review steps before being available for download on the App Store. Microsoft has a similar system in place the Windows Phone App Store. By contrast, Google does not have a similar policy for apps delivered to the Google Play Store (Google 2016a), relying on its users' capacity to identify and report malicious applications or on Trusted Party Management (which is covered later).
- **Code and Asset Assessment** – there are several well-known types of malware that can be identified by distinct signatures. Most anti-virus protection mechanisms rely on databases of such cases to identify potential threats to the system. Another way of assessing an app is to check which system calls, frameworks and methods it uses and for what reasons. Both iOS (Apple 2016a) and Windows Phone (Microsoft 2016) enforce this behaviour through their respective official application stores, for which each submitted application undergoes scrutinous inspection before being available for download.

- **Compartmentalization/isolation** – the basic premise for this approach is based on the complete separation of the user's personal space from the work space within the device, aiming to avoid unnecessary contact among both, while barring threats coming from the personal space. The workspace is usually subject to much stricter security and behaviour restrictions and might even not allow the installation or running of new scripts apart from the default assets. As an example, since Android API level 21, devices can have user accounts that are managed only by the main account, which can have a more limited set of permissions.
- **Trusted Party Management** – this method relies on the verification of trusted certificates, keys and signatures to manage which entities can interact with the system or which applications can be installed. For instance, only a single trusted computer might be able to write to the mobile devices storage (aside from the device itself) or only apps approved and properly signed by a company beforehand might be installed on the device. Android, iOS and Windows Phone all have the capability to install and manage custom trusted enterprise applications in the devices, be it through direct installation or certified third party application stores (Weiss 2013).
- **Permission Systems** – one way of circumventing the ordeal of assessing applications' functionalities or used tools is to leave it to the user to either accept or reject a list of granted permissions. Android and iPhone apps, for instance, must explicitly detail which system features or hardware capabilities they (might) need access to. For Android devices, this is a mandatory step performed before the app is installed unto the device (Google 2016b) whilst for iPhone devices, each individual permission must be asked during runtime when needed for the first time (Apple 2016b). In both scenarios, this information can be checked before installing the app and it is up to the user to evaluate and assume the risk of granting access to sensitive features such as access to the camera or SD card storage.
- **Authentication and Encryption Schemes** – these make use of encryption standards, with the possibility of depending on the existence of an external super user. The idea is to lock system management behind complex password/PIN mechanisms before any (or even every) write or read action is performed on the system. Modern mobile OS such as Android, iOS and Windows Phone also come with the option to encrypt the phone together with a locking mechanism, usually a PIN code or a fingerprint, without which the contents of the phone cannot be accessed.
- **Limited Time Access** – this technique makes use of ephemeral fixed length sessions, after which all the services are interrupted or aborted when the allocated time slot expires. Not available by default, this method is usually achieved at application level, using timed access tokens.

Most of these preventive methods have been imported from the desktop computing domain, and each one has its own advantages and disadvantages. Most of them try to delegate risk assessment to a more powerful entity (server or human), while also trying to impose a minimal resource overhead on the device. However, as can be seen, most of these methods are fallible for similar reasons. Previously unknown intrusion methods could simply pass by these security precautions, exploring undisclosed attack vectors allowing to circumventing them. Also, leaving it up to end user to figure out if a list of permissions is unacceptable or a potential hazard – it relies too much on common-sense and is subjective to the specific users' aptitudes and skills. In fact, one of the most common pitfalls of such methods is the need for the user to frequently deal with intrusive verification procedures required for application or content installation – users tend to simply forego.

Table 2 analyses at which moment of an application's lifecycle the already mentioned prevention strategies act, and for which mobile Operating Systems they are available or can be implemented.

Table 2 – Comparison of key moment of action for preventive methods

	Pre-Installation	Installation	Runtime	OS Availability
Sandboxing			x	Android, iOS, Windows Phone
Personalized Approval	x			iOS, Windows Phone
Code and Assets Assessment	x			iOS, Windows Phone
Compartmentalization			x	Android
Trusted Party Management		x		Android, iOS, Windows Phone
Permission Systems		x		Android, iOS, Windows Phone
Authentication Schemes			x	Android, iOS, Windows Phone
Limited Time Access			x	Android, iOS, Windows Phone

As we can see, many prevention strategies are effective at runtime, something that does not change with the emergence of ahead-of-time compiler techniques for some platforms (such as the ART/Android Runtime, which replaces the Dalvik VM (Bornstein 2008)) which could, in theory, provide anticipated analysis of application payloads. This is mainly due to the fact that it is harder to find suspicious payloads by solely resorting to binary pattern signatures or other static methods, rather than relying on continuous behaviour analysis.

3. Monitoring

Monitoring is crucial feature for security purposes, as it constitutes a key cyber-detection capability. It faces a different set of challenges from the prevention mechanisms already described as it is intended to detect and (if possible) halt malicious activity within the system at runtime. Mobile device monitoring and detection techniques may use different approaches, searching for signature patterns or deviations from the established normal behaviour (or fingerprints) not only at the device-level, but also involving several different devices, in a cooperative or distributed fashion. For this purpose, they may use system or device state and usage information, such as hardware characteristics, base station registration data, IP addresses, CPU usage, aggregated data transfers or TCP/IP stack state, together with application and service workload profiles, built using features such as app versions, data traffic, CPU usage and battery consumption. These techniques provide the means to detect issues such as device infection, hijacking and application permission misuse both at device-level and even involving several different devices, in a cooperative or distributed fashion. The most common monitoring and detection approaches are:

- **Behaviour Analysis** – this method is focused on what is and isn't expected from each type of application. For instance, apps or code can be scrutinized at runtime, using managed execution environments or specific instrumentation to detect any anomalous or suspicious activity, with the purpose of detecting infrequent activity patterns or access to sensitive information in the device. This process can be coupled with a training phase to establish a baseline behaviour model for known applications, or with Log Analysis, the next monitoring method on this list (Bose 2008; Burguera 2011).
- **Log Analysis** – the host system logs information regarding indicators such as memory and battery usage, storage accesses, data transfers and services used (Bluetooth, Wi-Fi, Mobile Data, etc.) to record app activity. The logs are then periodically sent to a trusted server to delegate the heavier and more resource intensive analysis burden out of the mobile device, albeit some minor pre-processing and analysis can still be performed on site. Log integrity measures should be considered on both endpoints for this method to work properly (Becher 2008; Mazumdar 2011; Halilovic 2012).
- **System Call Hooking (SCH)** – the act adding instrumentation to the host system for more fine-grained logging of the system's function calls (for native APIs and other frameworks), with the goal of easing the monitoring task (Willems 2007). This can help detect unnecessary use of mobile device features such as the camera or microphone. For instance, the dynamic shared library mechanism in Android makes use of a structure – the Procedure Linkage Table – that is used to implement the resolving mechanism for the dynamic linker (PLTs allow to implement lazy binding, providing stub functions which perform memory load operations on the Global Offset Table to retrieve the functions' real address). PLT patching can be used to implement code instrumentation mechanisms to detect anomalous situations, pretty much in line with the concept shown on Figure 2.
- **Runtime Permission Checks** – access to specific sensitive hardware or software features require apps to explicitly request permissions at runtime. For example, camera access can be managed using this technique: an app should request permission to use the device's camera every time to stop a malicious app from stealthily collecting unwanted pictures of the user or his surroundings. Android and iOS have runtime permission verifications, allowing for the user to toggle or opt out permissions after installing the application (Google 2016c; Mulligan 2014).

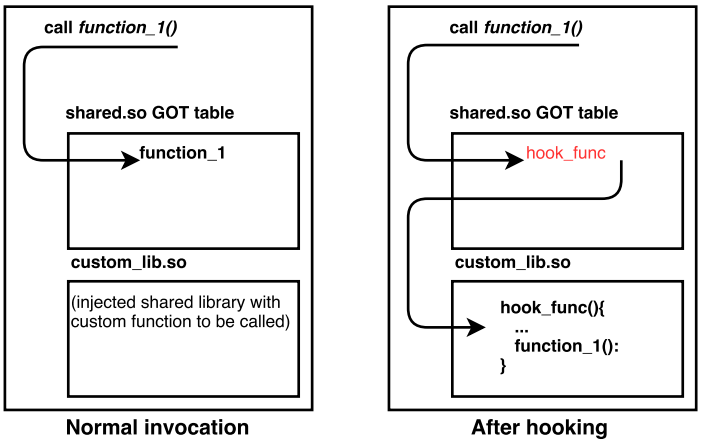

Figure 2 – Call hooking example (for interception of an exported function belonging to a shared library)

The main advantage of monitoring security strategies is a better and more refined assessment of the sensitive data and what actions or applications could be compromising it (hopefully) in time for intervention. They also enable a stricter control of what is acceptable behaviour through a more detailed logs and variable limitations. Given the more limited computing power and battery life, some of these methods can easily become an overkill for the device or rely too heavily on consistent and systematic connection with secure networks and servers, which might not be a possibility all together depending on the use case scenario. Another disadvantage is the risk margin for potential interventions in "near real time" which, depending on network status, system resources, cryptographic complexity and connectivity issues might result in an unacceptable latency.

It is worth mentioning that some monitoring methods cannot be implemented out of the box. The perfect example is System Call Hooking, which is very hard to perform or even impossible on most mobile operating systems without rooting or running a custom modified version of the original OS. Android is the most commonly adopted OS for this kind of barred monitoring techniques, because it is based on the Android Open Source Project – it can be more easily altered at core level to accommodate the desired changes, and it can run on a variety of devices and hardware, unlike iOS and Windows Phone.

Table 3 examines which threats each monitoring method can help prevent. Apart from Runtime Permission Checks, all methods seem equally effective against most threats. However, SCH is usually not readily available on most mobile devices and Behaviour Analysis requires a priori training, making Log Analysis the most effective out-of-the-box technique.

Table 3 – Comparison of prevented threats by monitoring method

	Privilege Escalation	Worms, Trojans, Viruses	Resource/ Data Misuse	Spyware	Theft, Device Cloning	OS Availability
Behaviour Analysis		x	x	x	x	Android, iOS, Windows Phone
Log Analysis		x	x	x	x	Android, iOS, Windows Phone
System Call Hooking	x	x	x	x		Android
Runtime Permission Checks			x			Android, iOS, Windows Phone

Although prevention and monitoring can drastically improve security, there will always be new malware or attack vectors against which they are ineffective. The best examples of this kind of malware are zero day, undisclosed and undetectable exploits, ranging from target specific applications such as Windows' Stuxnet (O'Murchu and Falliere 2011) to global pandemics such as Android's Stagefright (Wassermann 2015). Mitigation mechanisms, discussed next, address such threats.

4. Reactive and mitigation strategies: from isolated devices to organizational assets

Despite the existence of embedded security monitoring and preventive mechanisms within several mobile OS frameworks, which can be further complemented with other techniques and external or third-party components, it is impossible to guarantee complete and fail-proof protection from any risk. Therefore, any comprehensive approach towards mobile security must also encompass situations were protections have been overcome and the device has been compromised. Reaction and mitigation techniques are important for such situations, providing a way to minimize or contain the potential risk arising from successful intrusion attempts. Beyond the context of the mobile OS, such mechanisms also make sense from an organizational perspective, promoting the development of good practices for reaction and mitigation of security incidents, which must be closely coupled with asset and lifecycle management strategies.

Some of the preventive and monitoring techniques already described in the previous Sections can be considered for attack vector or vulnerability mitigation techniques on their own, but there are other means to try and repair a compromised device, or to at least minimize the damage it can do:

- **Security Policies** – written documents with organizational-wide rules and practices that must be enforced. For instance, these documents must identify the devices that can connect to the organizational infrastructure, what types of data they should contain, what constitutes as sensitive data, a list of supported operating systems and applications, and terms of acceptable use and penalties for misuse.
- **Documentation** – documents dedicated to scrutinize successful and unsuccessful aspects of a project, application or system, along with identified threats, how they were handled and how they could have been avoided. Usually performed at the end or at pre-defined iterative stages of the lifecycle, a Post Mortem document is a common example of this strategy.
- **Patch Support** – the ability to submit updates or patches for an application or system without having to physically access the device can be sufficient to fix newly discovered vulnerabilities or disable attack vectors.
- **Long Term Support (LTS)** – after distributing the system/application there is a time frame for which the developer agrees to provide support for security updates. Can be crucial to close exploited vulnerabilities or render them unusable.
- **Remote Control** – if a device becomes compromised it can be wiped, reset, shutdown or locked remotely, removing the possibility for an escalated compromise or continued/recurrent access to sensitive data.
- **Location Tracking** – if a mobile device is stolen and ends up in a conspicuous location it can be tracked down or simply located after the theft is reported.
- **Recall for Analysis** – the potentially targeted devices must be periodically recalled for a temporary system and application assessment. This method can be invasive to personal data if the device is also used for personal purposes or outside of the enterprise environment.

User Feedback – a means for the user of the device to report any suspicious or anomalous behaviour, no matter how ridiculous it might seem. This must be enforced with a sense of trust between the user and coordinators/feedback-receivers.

Table 4 analyses the moment at which each of the addressed strategies can help handle a potential threat: before, during or after it is detected or does damage. The first stage should not be confused with prevention, since it is assumed that a successful intrusion or attack has already occurred.

Table 4 – Strategy comparison accordingly with threat stage

	Before being active	During	After
Security Policies	x		
Patch Support	x		x
Long Term Support		x	x
Remote Control		x	x
Location Tracking			x
Recall for Analysis	x	x	x
User Feedback		x	
Documentation			x

As expected from the definition of "mitigation", there is great emphasis mitigating a threat after it has done damage more than the preventive and monitoring aspects that we have already discussed, which have more pre-emptive and persistent characteristics. The more detailed mitigation strategies and security countermeasures are adopted, the easier it can become to overcome threats. Figure 3 exemplifies how all the identified preventive, monitoring and mitigation methods can be combined into a security lifecycle model for mobile asset management, identifying the three main phases at which security episodes can be identified and, possibly, prevented.

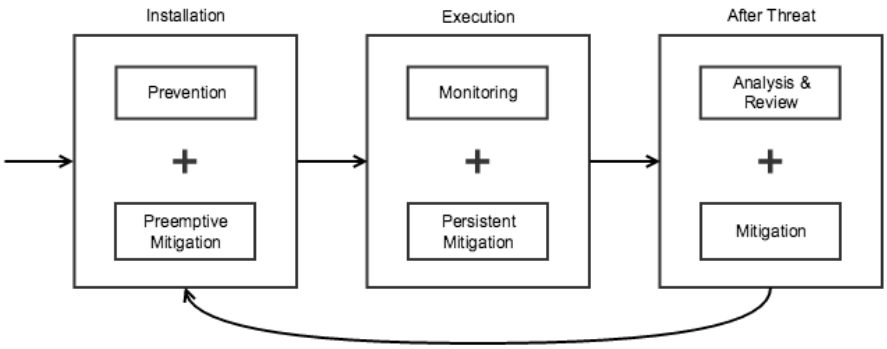

Figure 3 – Security lifecycle for mobile asset management

Figure 3 is reminiscent of a simpler Observe-Orient-Decide-Act (OODA) loop (Boyd 1995), a common strategic approach to military and commercial operations and learning processes. It focuses on agility when dealing with a threat, in order to mitigate its advances and damage at any given point in time, learning from any form of input and feedback obtained. Figure 4 portrays a simplified version of the Boyd's OODA loop, representing all these aspects.

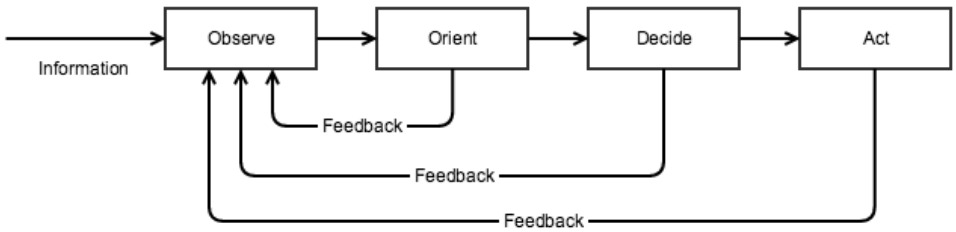

Figure 4 - Simplified OODA loop

If should be stressed that the observation stage can be replaced by a warning or notification from a security organism, such as a CERT. For instance, if a researcher/developer finds vulnerabilities in an application/system he should contact the developer about his findings and, if given permission (in case of coordinated disclosure), make them available as soon as possible, or simply make them public as soon as he is certain of them (in case of full disclosure). Accordingly with the timing and nature of the announcement, the reaction time between the notification and the deployment of adequate security measures can be quite short, prompting the mobile device management teams to enforce conservative security mechanisms as a preventive measure.

5. Final Remarks

The aim of this survey was to provide an analysis about the available security techniques that can be used to maximize the overall safety of enterprise and personal data and applications stored and used on mobile devices. While almost all the described technologies have their specific advantages and shortcomings, this paper's objective to show how they may work together to create a more robust mobile security environment.
Also, this paper provided an overview about mitigation and reaction strategies that is mostly focused on the organizational perspective, also being aligned with asset and app management practices. Finally, an OODA-inspired model for managing security events is also proposed, which brings together all the discussed categories of security mechanisms (prevention, monitoring, reaction/mitigation) within a common lifecycle management strategy.

Acknowledgements

This work was carried out with the support of the Centro 2020 Project MobiTrust (CENTRO-01-0247-FEDER-003343), which is part of the CATRENE project MobiTrust (CA208 – MobiTrust). The authors would like to thank the other consortium partners involved in these projects for the valuable discussions and contributions to the work presented in this paper.

References

Alfalqi, K., Alghamdi, R. and Waqdan, M. (2015) Android Platform Malware Analysis., 6(1), pp.140–146.

Apple App Center (2016a) App Store Review Guidelines. Available at: https://developer.apple.com/app-store/review/guidelines/ (last accessed October 21, 2016).

Apple Developers (2016b) About Info.plist Keys and Values. Available at: https://developer.apple.com/library/content/documentation/General/Reference/InfoPlistKeyReference/Introduction/Introduction.html (last accessed October 21, 2016).

Armando, A. et al. (2012) Changing user attitudes to security in bring your own device (BYOD) & the cloud. Network Security, 2012(3), pp.5–8.

Becher, M. and Freiling, F.C. (2008) Towards Dynamic Malware Analysis to Increase Mobile Device Security. In Proc. of SICHERHEIT, P-128, pp.423–433. Available at: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84873743297&partnerID=40&md5=568aa830f6fbf59ca90683d4c5c34d23.

Bornstein, Dan. "Presentation of Dalvik VM Internals" (PDF). Google. https://sites.google.com/site/io/dalvik-vm-internals/2008-05-29-Presentation-Of-Dalvik-VM-Internals.pdf (last accessed on November 11, 2016).

Bose, A. et al. (2008) Behavioral detection of malware on mobile handsets. Proceeding of the 6th international conference on Mobile systems, applications, and services - MobiSys '08, p.225.

Boyd, J. R. (1995) The Essence of Winning and Losing. A Discourse on Winning and Losing.

Boyd, J.R. (1976) Destruction and Creation. A Discourse on Winning and Losing, (September), pp.3–9.

Burguera, I., Zurutuza, U. and Nadjm-Tehrani, S. (2011) Crowdroid: Behavior-Based Malware Detection System for Android. Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices - SPSM '11, p.15.

Distefano, A., et al. (2011) SecureMyDroid: Enforcing Security in the Mobile Devices Lifecycle.

Eslahi, M. et al. (2015) BYOD: Current state and security challenges. ISCAIE 2014 - 2014 IEEE Symposium on Computer Applications and Industrial Electronics, (April 2014), pp.189–192.

FireEye (2015) Zero-Day Danger: A Survey of Zero-Day Attacks and What They Say About the Traditional Security Model., p.16.

Frei, D. (2012) Conducting a Risk Assessment for Mobile Devices.

Google Android Developers (2016a) App Manifest. Available at: https://developer.android.com/guide/topics/manifest/manifest-intro.html (last accessed October 21, 2016).

Google Android Developers (2016b) Launch Checklist. Available at: https://developer.android.com/distribute/tools/launch-checklist.html (last accessed October 21, 2016).

Google Android Developers (2016c) Requesting Permissions at Run Time. Available at: https://developer.android.com/training/permissions/requesting.html (last accessed October 21, 2016).

Halilovic M., A. Subasi (2012) Intrusion Detection on Smartphones.

Harris, M.A. and Patten, K.P. (2015) Mobile Device Security Issues Within the U.S. Disadvantaged Business Enterprise Program. Journal of Information Technology Management, XXVI(1), pp.46–57.

Jaatun, M.G., Jaatun, E.A.A. and Moser, R. (2014) Security considerations for tablet-based eHealth applications. CEUR Workshop Proceedings, 1251(Pahi 2014), pp.27–36.

Jaramillo, D., B. Furht, and A. Agarwal (2014) Mobile Virtualization Technologies. In: Virtualization Techniques for Mobile Systems, 14th ed., Vol 73, pp. 5-20.

Kocher, P. et al. (2004) Security as a new dimension in embedded system design. DAC '04: Proceedings of the 41st annual Design Automation Conference, pp.753–760.

Lebek, B., Degirmenci, K. and Breitner, M.H. (2013) Investigating the Influence of Security, Privacy, and Legal Concerns on Employees' Intention to Use BYOD Mobile Devices. Amcis, (2008), pp.1–8.

Lee, J., T. Kim, and J. Kim (2009) “Energy-efficient Run-time Detection of Malware-infected Executables and Dynamic Libraries on Mobile Devices”, Software Technologies for Future Dependable Distributed Systems.

Li, Q. and Clark, G. (2013) Mobile Security: A Look Ahead. IEEE Security & Privacy, 11(1), pp.78–81

Luo, J., and m. Kang (2011) “Application Lockbox for Mobile Device Security”, Eighth International Conference on Information Technology: New Generations.

Mazumdar, S. and Paturi, A. (2011) Tamper-resistant database logging on mobile devices. Internet Security (WorldCIS), 2011 World Congress on, pp.165–170.

Microsoft Windows Dev Center (2016) The app certification process. Available at: https://msdn.microsoft.com/windows/uwp/publish/the-app-certification-process (last accessed October 21, 2016).

Miettinen, M., and P. Halonen. (2006) “Host-Based Intrusion Detection for Advanced Mobile Devices”, 20th International Conference on Advanced Information Networking and Applications (AINA'06), Vol 1.

Mulligan, B. (2014) The Right Way To Ask Users For iOS Permissions, Techcrunch. Available at: https://techcrunch.com/2014/04/04/the-right-way-to-ask-users-for-ios-permissions/ (last accessed October 21, 2016).

Nabi, R.M. and Mohammed, R.A. (2015) Smartphones Platform Security a Comparison Study., 5(11), pp.44–48.

Olalere, M., et al. (2015) “A Review of Bring Your Own Device on Security Issues”, SAGE Open Apr 2015, 5 (2) 2158244015580372; DOI: 10.1177/2158244015580372.

O’Murchu, L., Falliere, N. (2011) “W32.Stuxnet dossier”, Symantec White Paper, February 2011.

Rhee, K., Jeon, W. and Won, D. (2012) “Security requirements of a mobile device management system”, International Journal of Security and its Applications, 6(2), pp.353–358.

Scarfo, A. (2012) New security perspectives around BYOD. Proceedings - 2012 7th International Conference on Broadband, Wireless Computing, Communication and Applications, BWCCA 2012, pp.446–451.

Schmidt, A.-D., et al. (2009) “Static Analysis of Executables for Collaborative Malware Detection on Android.”, 2009 IEEE International Conference on Communications, doi:10.1109/icc.2009.5199486.

Shabtai, A. U. Kanonov, and Y. Elovici (2010) “Intrusion Detection for Mobile Devices Using the Knowledge-Based, Temporal Abstraction Method”, Journal of Systems and Software, Vol 83, No. 8, pp. 1524–1537.

Skovoroda, A. and Gamayunov, D. (2015) Securing mobile devices: Malware mitigation methods. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 6(2), pp.78–97.

Souppaya, M. and Scarfone, K. (2013) NIST Special Publication 800-124 Guidelines for Managing the Security of Mobile Devices in the Enterprise., p.30.

Team, U.-C.C.R. (2010) Technical Information Paper-TIP-10-105-01 Cyber Threats to Mobile Devices. US-CERT Technical Information Paper, pp.1–16.

Venugopal, D., and G. Hu. (2008) “Efficient Signature Based Malware Detection on Mobile Devices”, Mobile Information Systems, Vol 4, No. 1, pp. 33–49.

Väisänen, T. et al. (2015) Defending mobile devices for high level officials and decision-makers, pp.1–107.

Wassermann, G. (2015), “Vulnerability Note VU#924951 – Android Stagefright contains multiple vulnerabilities”. CERT. Available at: https://www.kb.cert.org/vuls/id/924951 (last accessed November 2, 2016).

Weiss, A. (2013) How to Deploy Enterprise Mobile Apps. Enterprise Apps Today, Available at: http://www.enterpriseappstoday.com/crm/how-to-deploy-enterprise-mobile-apps.html (last accessed October 21, 2016).

Willems, C., T. Holz, and F. Freiling (2007) “Toward Automated Dynamic Malware Analysis Using CWSandbox”, IEEE Security and Privacy, Vol 5, No. 2, pp. 32-39, March.

Wu, F. and Chen, C., (2014) Sensitive Data Protection on Mobile Devices., 5(9), pp.38–41.