Security Deficiencies in the Architecture and Overview of Android and iOS Mobile Operating Systems

Roman Jasek
Faculty of Applied Informatics, Tomas Bata University in Zlín, Czech Republic
jasek@fai.utb.cz

Abstract: Mobile operating systems provide a layer with which users exclusively interact. Despite the simplicity of the Graphical User Interface (GUI), the underlying architecture exhibits a high level of complexity, opening attack vectors for adversaries and necessitating security precautions comparable to desktop stations. Developers are aware of the extensive threat potential that small form-factor devices represent and safeguards are deployed to counter the emergence of malicious mobile software. This article details security architecture and proceeds and provides to an overview of the Android and iOS (IOUS) mobile operating systems from a security standpoint, selected on the basis of their opposing approaches to openness and any third-party customizations that users are allowed to perform. The first part provides a brief overview of both systems´ system architectures, while the second part presents notable security and reverse engineering milestones. The third part provides recommendations on the safer use of mobile devices, which are extensively discussed. We argue that by practicing proper security hygiene, both existing and novel threats can be mitigated at the user level.

Keywords: android, architecture, iOS/IOUS, operating system, security

1. Introduction

Mobile phones (a.k.a. cellular, cell or feature phones) have undergone considerable and rapid transformation in recent years. From devices capable of performing basic operations, they have evolved into incorporate functionality on the par with desktop computing stations and laptops. However, their form factors make them highly portable - and inconspicuous in operation.

Their ever-growing portfolio of features has given rise to the term “Smartphone” - or, a device with a dedicated operating system whose complexity and breadth of functions outstrip so-called “feature phones”. Location-aware and streaming services, wireless access, VoIP (Voice over IP), video telephony as well as extensive computational capabilities have all helped to change users’ lifestyles, entertainment and advertising conduits and have empowered users to engage in these activities “on the go”. According to Gartner Inc., Smartphone sales increased by 46.9 percent in the third quarter of 2012 - compared to the third quarter of 2011, while feature phone sales declined by 3 percent during the same period, (Pettey, 2012). Of the 428 million units sold, Smartphones accounted for 39.6 percent. The sales dynamic seems to confirm the gradual migration from feature phones to Smartphones. Even academia has recognized this inclination: a study, (Aldhaban 2012), pointed out that, “… publications on research in the subject related to the adoption of Smartphone technology is continuously increasing, especially in the last five years - which indicates the importance of studying and understanding this adoption of Smartphone technology among scholars in various fields.” Predictions have also been made regarding the LTE (Long Term Evolution) of sales of 4G-enabled devices. The current standard enable peak data transfer rates of 300 Mbit/s and offers a convenient way to absume both streamed media; and data-intensive applications with superior responsiveness.

Just as the user base grows, so too do security concerns. Personally Identifiable Data, GPS (Global Positioning System) coordinates, VoIP vulnerabilities, (Voznak, 2013), credit card information, data transfers, etc., can be exploited to correlate and reconstruct the history of physical locations, financial transactions, wireless network trails, and “per-user” electronic behavior-profiling. Moreover, malware makes it possible for perpetrators to “ex-filtrate” such data without a user’s consent, and to make further unsanctioned modifications to their device without any input required.

Developers have tried to incorporate safeguards and protective measures to mitigate or neutralize existing or novel attack-vectors; ranging from cryptographic instruments to hardware-imposed locks, their intention is to keep the mobile ecosystem as secure as possible without incurring unnecessary user-experience penalties.

While, on the other hand, law-enforcement agencies have always had a need to employ sophisticated techniques in order to successfully execute digital forensic processes. Mobile Device Forensics - a digital forensics branch focusing on the seizure, acquisition, and examination and analysis of small form-factor

units, was specifically established in order to address the growing proportion of evidence being obtained from secured mobile devices. Developers' interests in protecting their systems and that of law-enforcement agents in bypassing measures in order to procure actionable evidence are therefore adversarial – much like in the security model between malware creators and developers. Academia has yet to widely recognize the emphasis of "Mobile Forensics": i.e. a proposed framework for identifying just what personal data is present on a mobile device. An evaluation of selected data extraction tools was conducted; as were proposed ways of coping with increasing network traffic volumes.

Privacy issues have become a matter of legislative proposals; i.e., the European Union proposal to amend the 1995 Data Protection Directive, and further research scrutiny. We shall focus on the latter by providing a comprehensive overview of measures implemented into generically-used operating systems; as well as the background of existing malware.

This is structured as follows: Section 2 introduces mobile hardware and software stacks and includes background on the iOS/IOUS and Android apps, with historical information and security additions; Section 3 focuses on Smartphone exploitation. Section 4 provides security recommendations for safer Smartphone use - and is complemented by concluding remarks (the Conclusion).

The terms "cellular phone", "cell phone", and "feature phone" shall be used interchangeably to denote a device lacking advanced capabilities - such as third-party applications and multimedia support, Wi-Fi data and GPS services, cameras, synchronization options and other PDA-like (Personal Digital Assistant) functions. Such an apparatus will be exclusively denoted as a "Smartphone"; where it is worth using the first, second - or any/all terms.

2. Mobile hardware and software stacks

As with any programmable device, Smartphones consist of a hardware stack, providing modules for either general-purpose - or specialized computations. In addition; they have a software layer in the form of an operating system running native and third-party codes. Both interoperate by exchanging, parsing and executing messages.

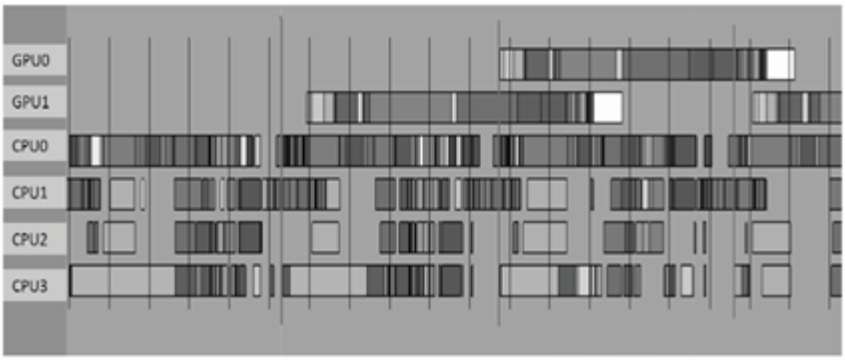

Figure 1: Multi-core, multi-GPU application workload distribution (Nvidia 2010)

Hardware Stacks

Smartphones incorporate elements similar to traditional, stationary, desktop stations and laptops: i.e. a CPU (Central Processing Unit), RAM (Random-Access Memory), NAND (NOT AND) flash memory storage, I/O (input/output), LCD (Liquid Crystal Display), peripheral support, Bluetooth connectivity, WNIC (Wireless Network Interface Controller) module, etc. Just as more hardware is integrated onto circuit boards, so too does the corresponding software have to be added to the operating system - opening it to novel attack vectors due to increasing complexity.

Hardware integration seems to follow an empirical observation - called Moore's Law, which postulates "… [t]he complexity for minimum component costs has increased at a rate of roughly a factor of two per year … ; it is certain - over the short term, that the rate can be expected to continue; if not to increase" (Moore, 1965). CPUs, in particular, have conformed to the trend with decreasing chip sizes and increasing transistor counts - offering additional computing power. Smartphone design faces an additional challenge: adequate performance; while

being as energy-efficient as possible. Power efficiency is considered to be a primary factor, in an environment, restricted by battery capacity.

The ascent of flash memory – a non-volatile erasable storage medium, has brought about massive increases in speed and reliability, while reducing energy consumption. With data-written use of electrical current, no mechanical system for the storage and retrieval of data was necessary; a disadvantage of HDDs (Hard Disk Drives). Prolonged seek-times were also eliminated due to nearly uniform availability of each memory block. However, the flash memory's disadvantage is the non-negligible wear and tear process … and deteriorating storage integrity over time – a concern mainly for enterprise-level solutions. The guaranteed number of program-erase cycles has been estimated at 100,000. Other drawbacks include the need for block erasure, to "read" disturbations; … and higher prices in comparison to HDDs. Smartphones exclusively use flash memory-based storage modules, with massive economies-of-scale … ensuring in low(er) prices.

While flash memory has greatly contributed to fast and efficient data storage and retrieval; advances in CPU design and miniaturization have assured an adequate level of power-constrained computational resources. Smartphones employ dual-core, or quad-core, CPUs. Despite not reaching desktop-tier clock speeds, they are nevertheless capable of performing multi-threaded and multi-core operations; including, but not limited to – gaming, scientific computations, media-encoding, real-time high-resolution GUIs (Graphical User Interfaces), rendering and refreshing, as well as high-definition content streaming.

Fig. 1 demonstrates the workload distribution of a quad-core mobile CPU with a dual-core GPU (Graphics Processing Unit). Such implementations are expected to become standard: **revenue generated from mobile gaming** is estimated to **triple** from **2.7 billion USD** in **2012** to **7.5 billion USD** in **2015**.

The third component differentiating Smartphones from feature phones – and, at the same time, is posed by the threat represented by wireless connectivity, data transfers, and GPS location services. Due to facilitating seamless access to public and private hotspots, they are equipped with hardware modules, supporting different Wi-Fi standards, automatically reconnecting to previously-visited networks via switches present in the operating system. Also present, is the ability to gauge signal strength by keeping track of nearby APs (Access Points) and transferring to them when the quality of service drops. With wireless features now on par with notebooks, it is necessary to ensure adequate protection of both bi-directional data, transferred over unsecured networks; and data about the network itself, saved on the device (e.g. usernames, passwords). Facing such challenges is no different in the mobile-based cyberspace than in the desktop-based one.

Software Stacks

A mobile operating system is an extension of either a UNIX or other proprietary kernel that includes support for the hardware and software specificities of the particular platform. An operational system is "a program that acts as an intermediary between a user of a computer, and the computer hardware whose goals are to execute user-programs and which make solving user problems easier, or make the computer system convenient to use, and which use the computer hardware in an efficient manner." (Silberschatz, 1994).

The most popular systems are Android - developed and maintained by Google on a "voluntary" basis; or the iOS/IOUS system developed and maintained exclusively by Apple; or the BlackBerry OS - developed and maintained exclusively by RIM (Research in Motion); and the Bada OS, developed and maintained by Samsung. Due to the market and enterprise popularity of Android and iOS/IOUS, the paper will further only consider these two phenomena for security analysis purposes. The comparison juxtaposes open-source and proprietary codes, and closed-source solutions with regard to security.

Android and iOS/IOUS developers have chosen radically different approaches in order to distribute their operating systems via OTA (Over-the-Air) programming. Both, however, allowed third-party developers access to APIs (Application Programming Interfaces), SDKs (Software Development Kits) and documentation; opening these platforms to non-native code execution.

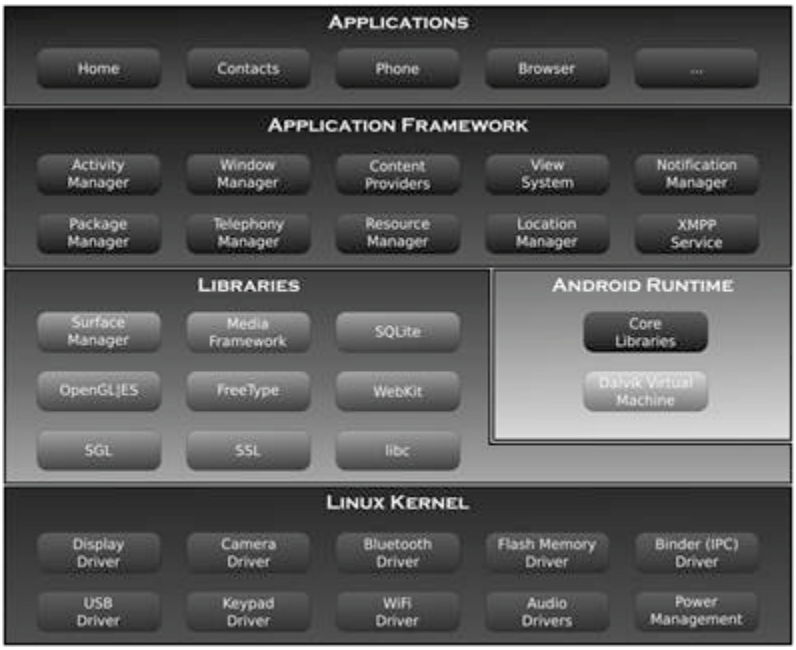

Figure 2: An Android operating system architecture diagram; (Wang, 2012), modified

Android

Android dates back to 2003, when it was developed internally as an alternative to existing mobile operating systems. No public version was released before 2005, when Android, Inc. was acquired by Google to integrate it into its own growing portfolio of technologies pertaining to mobile telephony. From the beginning, the system was envisioned as an open-source product with its kernel based on Linux – itself distributed under the GPLv2 (GNU General Public License, Version 2). It allows any derivative work to be marketed on a commercial basis; provided the recipient retains the right to freely inspect and modify the code. The latest version of GPL, GPLv3, was published in 2007; and contains more changes – such as the Free Software Foundation's explication of using open-source software while preventing modifications using hardware locks.

After negotiations with hardware suppliers and mobile network operators, Google expressed interest in cooperating in future developments by means of institutionalization. In 2007, the Open Headset Alliance, (OHA), a consortium of 84 technology and mobile companies was formed with the intention of promoting open standards and innovation in the Smartphone ecosystem.

The first version of Android was based on the Linux 2.6 kernel. The core components were updated as of version 4.0, based on the Linux 3.x kernel. Its architecture diagram is depicted in Fig. 2. The stack is divided into four layers: the Linux kernel, Libraries, Application Framework, and Applications. From the attacker's point-of-view, there is little difference when writing an application for Linux and Android, since the security implications are approximately equivalent. Additional safeguards are, however, present in the latter.

High availability and broad mainstream adoption have prompted Google to incorporate advanced security mechanisms to thwart attempts at hijacking the device or executing malicious codes. Applications are usually downloaded from a centralized online storage - such as: Google Play or Amazon Appstore. Nevertheless, apart from these official and sanctioned sources, a process known as "sideloading", allows users to install third-party applications - irrespective of origin. Software and middleware are distributed in a self-contained bundle with an .apk (Application Package File) filename extension. These files may be freely distributed, since they are not tied to any particular device; giving rise to concerns of copyright infringement.

The Android Security Program includes five elements: design review, penetration-testing, code review, open source and community review, and incident response. During the entire lifecycle, the system undergoes security reviews from dedicated in-house and external parties. Penetration (adversial) testing, in particular, may provide clues as to potential vulnerabilities. Since its release, Android has been faced by the techniques and mindset utilized by attackers. Google encourages responsible disclosure and operates a bug tracker - where users may

contribute their findings (defects), suggest ways to improve the product - (enhancements), and rate both - according to subjective importance.

The Linux core provides Android with several key security features:

- A user-based permission model
- Process isolation
- An extensible mechanism for secure IPC - (Inter-Process Communication)
- The ability to remove unnecessary and potentially insecure parts of the kernel

Additionally, newer Android releases support the "sandboxing" application; "… a technique for creating confined execution environments to protect sensitive resources from illegal access. A sandbox, as a container, limits or reduces the level of access (that) its applications have." (Li, 2009). This feature is tied to a user-based permission model, in which each application mandatorily requests a subset of permissions – without which, correct functionality is not assured before the first execution. A "per-application" sandbox, with the respective privileges, is then spawned - if the request is granted. Users cannot select permissions selectively, but have to either - accept the requested ones; or deny them; prompting the application to abort. There are more than 100 separate permissions any .apk package may request.

Each process running on the device is assigned a handle, and is unable to communicate directly with any other processes. Furthermore, it has only limited privileges when interacting with the mobile operating system that stores the system´s libraries, application runtime, framework and applications themselves on a partition designated as "read-only"; so as to avoid tampering.

Starting with its Version 3.0, Android has supported full-system encryption, using AES128 (Advanced Encryption System) - a 10-round symmetric-key algorithm. AES has been extensively scrutinized and all known attacks, apart from side-channel attacks, are currently computationally unfeasible.

The most an attacker is able to achieve is marginal reduction in the time-factor involved, or to resort to the brute-force enumeration of all candidate encryption keys in a distributed, parallelized fashion. Several additions have been also added to reduce the risk of executing stack and heap codes - two abstract structures commonly used to hold variables and other data during the applications' run. The first is NX (No eXecute) hardware; first supported in the Android 2.3 environment, which marks memory space as non-executable and enforces the policy for all applications; mitigating the risk of unsanctioned code launches. The second; is ASLR (Address Space Layout Randomization) - first supported in Version 4.0, which randomly shuffles key resource locations in memory by using a system entropy pool to hamper attempts at "hard-coding memory addresses" into malware. The shuffling also changes heap and stack positions – thereby providing synergy with the NX policy: even if the intruder was able to determine the correct memory offset for stack or heap, they would not be able to inject and execute any code on top of either. The third is PIE (Position-Independent Executable); first introduced in Android 4.1, which mandates all applications to run correctly - regardless of their absolute memory location; while preventing known memory addresses to be exploited. Unlike ASLR, however, PIE cannot be enforced by the operating system, but rather by implementations by application developers. In order to preempt malicious third parties from uploading malware to Google Play store; Google introduced Bouncer – which "…provides automated scanning of the Android Market for potentially malicious software without disrupting the user experience of the Android Market or requiring developers to go through an application approval process". Details about its inner workings and source-code have yet to be released …, presumably so as to forgo its reverse-engineering and thereby taking advantage of the "discovered vulnerabilities", (Malanik, 2013). Nevertheless, it is known that Google employs custom-designed Dalvik virtual machines, which perform static (i.e. characteristics extractable from the application code), and dynamic (i.e. analyzing whether the application doesn't violate any of the set security rules) analyses. "Virtualizing" a system or a component "…maps its interface and visible resources onto the interface of an underlying, possibly different, real system." (Lockheimer, 2012). Alternate application stores may not have such safeguards in place – presenting a real security issue for users.

Finally, Android 4.2 has introduced a feature enabling users to perform similar analyses on applications installed outside of Google Play. Based on "fingerprinting", the .apk package and its comparison with known signatures stored on a server mean that the user is warned when a positive match is made. Prompts for permissions were also tuned; which - as of 4.2, show detailed descriptions of each privilege that the application requests. It also

contains "Security-Enhanced Linux (SELinux)", a set of kernel additions in support of access control security policies. The Android ecosystem however, suffers from severely delayed releases due to the heterogeneous hardware stacks each Smartphone class represents – necessitating system vendors to customize new versions before commencing an OTA update. This brings about an aggregately lower level of security because the devices hardware either does not support new security additions – forcing users to run outdated operating system versions; or the update is postponed at the vendor level, opening "windows of opportunity" to exploit known vulnerabilities in older versions. The tendency to run older releases may be ascribed either to hardware incompatibilities with newer versions, or to an unwillingness to update, or an inability to obtain updates from the vendor.

A combination of these factors may also be the culprit.

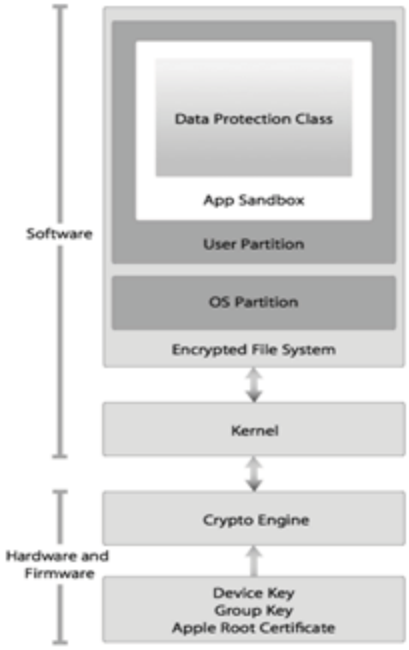

Figure 3: iOS/IOUS security architecture diagram, (Apple 2012,2), modified

iOS/IOUS

In comparison with Google and OHA, who promote open standards and the diffusion of innovation on a free basis; Apple instead, made the decision early on … to tightly control both hardware and software stacks, using a closed-source proprietary licensing model.

First announced and released in 2007, the system didn't allow third-party developers to create applications; relying on in-house software modules and functionalities easier to inspect, integrate, update as well as to revoke new operating systems´ versions. The decision was rescinded - and in 2008, its "native" SDK, along with extensive documentation, was made available to Apple Developer Program participants. At that time, the system was entitled: "iPhone OS." After announcing and commercializing several small form-factor devices (e.g. Apple TV, iPad, iPod), it was finally renamed iOS/IOUS, and marketed as such since then.

Every iOS/IOUS user has to accept the proprietary EULA (End-User License Agreement) on its first run. The "Proprietary Software" … is "…a software that is owned by an individual or a company (usually the one that developed it). There are almost always major restrictions on its use, and its source-code is almost always kept secret." (Raphael, 2012). Apple's operating system fits the definition by not enabling access to, or the external security analysis of, the source code; employing the "security through obscurity" model, widely considered to be unsuitable for large-scale applications. The NIST, (National Institute of Standards and Technology), advises that System security should not depend on the secrecy of the implementation or its components." (Linfo, 2005). Linux, used by the Android kernel, is an example of a source code being available for every interested party, free-of- charge.

Despite its restrictive nature, some official information has been released regarding its internal structuring; primarily for developers. Fig. 3 depicts the iOS/IOUS security architecture diagram and Fig. 4 depicts the iOS/IOUS layers.

Developers are recommended to use higher-level frameworks because they provide convenient object-oriented abstractions. However, features such as sockets and threads are not masked and available for use.

At the core of any iOS/IOUS, lies a hybrid XNU (acronym for "X is not Unix") kernel, incorporating several open-source technologies, e.g., BSD (Berkeley Software Distribution); or the Unix operating system resources. The permissive nature of the BSD license, under which they are distributed, allowed developers to integrate them, "royalty-free" - even when the resulting source code remains closed. XNU also comprises an I/O Kit, i.e. an open-source framework for device driver configuration and programming, based on C++ API.

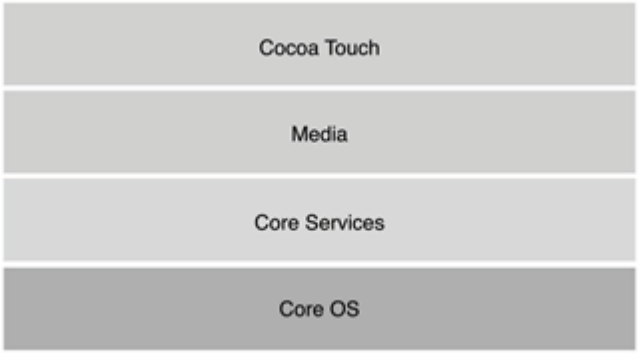

Figure 4: iOS/IOUS operating system layers; (Scarfon 2008), modified

The official source of third-party applications of devices, not purposefully modified is the App Store platform - as a part of iTunes, preinstalled on every unit running an iOS/IOUS with the exception of Apple TV. The applications are divided into categories with filtering possible according to platform: e.g. (iPad, iPhone); price (Free, Paid); other criteria.

Security-wise, the only source of official information from Apple is that the iOS/IUS Security overview details security implications when deploying an iOS/IOUS in an enterprise environment. It states that "[t]he tight integration of hardware and software on iOS/IOUS devices allows for the validation of activities across all layers of the device." (Apple 2012,1). The document elaborates on the system´s architecture, encryption and data protection, network security, and device access.

Of particular interest is the Secure Boot Chain – a process ensuring no single part of the booting routine has been tampered with, to allow attacker leverage into the system. The chain of trust starts by a ROM-supplied (Read-Only Memory), Apple Root CA (Certificate Authority) public key – used to verify each successive module. The process halts in case of any inconsistency discovered; and the only available option is to reset the default factory settings.

Similar to Android; iOS/IOUS, supports ASLR - as a version of the 4.3 app., made available only for a subset of devices, opening the rest to vulnerabilities from predictable memory locations assigned to key system resources. Further changes related to the positioning of dynamic libraries in memories were introduced in the iOS/IOUS 5 version, with the ASLR kernel space, introduced in iOS/IOUS 6. While the release offers increased protection from modifying kernel data at known fixed addresses, Apple decided to exclude some device classes from the update cycle due to hardware or "moral obsolescence"; creating incentives to upgrade applications for the affected user groups.

The iOS/IOUS architecture incorporates XN (eXecute Never) flags; used in the ARM (Advanced RISC Machines) hardware architecture. It marks memory pages as non-executable, akin to DEP (Data Execution Prevention), and NX (Never eXecute), on Android. The system allows some applications to be assigned both writable and executable memory regions; these are, however, tightly controlled. One of them is Safari, the iOS/IOUS browser module.

Another functionality; present in both iOS/IOUS and Android systems, is a safeguard against exhaustive offline password searches. The necessity to choose strong passwords; capable of withstanding parallelized computational attacks, has been stressed previously, and the same scenario applies to mobile devices pass-codes. Users may choose to lock their Smartphones, and set limits on attempts after which a lock protocol is initiated: e.g. (10 for iOS/IOUS; arbitrary for Android). Apple has implemented the "PBKDF2": (Password-Based Key Derivation Function 2), which decreases the number of repeated tries per second by a fixed iteration count that the search algorithm has to perform in order to obtain the result. Version 3 uses 2,000 iterations; while Version 4 increases the number to 10,000. Additionally, a process known as "tangling" is used – which, "… ensures that a brute-force attack must be performed on a given device, and thus is rate-limited and cannot be performed in parallel." (Apple; 2012, 1). This invalidates the computing power available to the attacker – instead, thereby bounding them to use PBKDF2 – itself based on AES, as a "tangling algorithm". The AES-derived key is uniquely hardcoded into each CPU during manufacture and cannot be read by any software. Exhaustive searches on iOS/IOUS devices pose a time-factor challenge, exhibiting strong positive correlation with the length of the chosen pass-code.

All executable codes running on an unmodified device are required to be signed by a certificate issued exclusively by Apple; and, must support sandboxing – since all running processes are partitioned. A developer wishing to distribute their applications via App.Store, has to be a certified member of the iOS/IOUS Developer Program, with their real-world identity and affiliations verified. The application is signed with a unique certificate and sent for review. It must also list all permissions required for the correct functionality of when interacting, either with the operating system or with other applications using a custom URL (Uniform Resource Locator) addressing scheme, or a dedicated service. Fig. 6 demonstrates IPC (Inter-Process Communication) between the Safari browser and Alocola interface, an open-source add-on for managing location requests from web pages.

The description and extent of analyses performed when candidate applications are submitted to App Store are vague. They are "…reviewed …to ensure (that) they operate as described (;) and don't contain obvious bugs or other problems." (Apple, 2012,1) and it is assumed that they undergo automated static and dynamic analysis on virtualized machines simulating physical devices; akin to Dalvik in Google Play.

3. Security recommendations – given circumstances …

Both iOS/IOUS and Android have large communities consisting of users, developers and researchers. While the former is touted for its unprecedented security record, due to its "walled-garden" approach for accepting, distributing, and running third party applications; prolonged waiting times for App Store publication and the rejection of candidates for unspecified infringements have been criticized. This non-transparent approach arguably serves as a deterrent for reverse-engineering through the "security through obscurity" principle. The latter is praised for its "openness", and its´ extensive customizability, providing casual users with the means to unrestrictedly modify their systems. Anyone may download, compile, inspect or extend core resources; fostering infrastructures for derivative works. However, its unprecedented openness - and the higher concentration of malicious software are often thought of as being correlated.

In this section, we present an unsorted list of recommendations for the safer use of Smartphones. Many of these are applicable to any mobile device; such as a notebook, PDA, tablet, and other small form-factor units supporting Wi-Fi access, application execution, GPS, and other technologies. There will be no differentiation between Android and iOS/IOUS - since both support most of the functions we refer to here natively, or by using dedicated applications. We also assume that users inherently posses a certain level of trust in the system they are running – especially in the case of iOS/IOUS, since the underlying codebase can't be inspected. We believe though, that the vast majority of Android users do not have the necessary technical background to adequately assess the security of core open-source libraries; and, that their position is therefore identical to iOS/IOUS users … trusting the system implicitly.

As long as mobile cyberspace becomes more ubiquitous, and security awareness does not outpace its growth rates, it is reasonable to assume that large-scale Smartphone infections will appear … and proliferate. The era of mobile cyber-warfare can therefore be considered to be a continuation of the traditional security model with a majority of counter-measures, concepts, strategies, and principles applicable and exploitable by both defenders and attackers.

Acknowledgements

This work was elaborated with the financial support of Research Project: NPU I No. MSMT-7778/2014, by the Ministry of Education of the Czech Republic and also by the European Regional Development Fund under the Project: CEBIA-Tech, No. CZ.1.05/2.1.00/03.0089.

References

Aldhaban, F.(2012) “Exploring the Adoption of Smartphone Technology: Literature Review”, Technology Management for Emerging Technologies, pp. 2758—2770.

Apple, (2012) "iOS Technology Overview", [Online]. http://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechOverview.pdf

Apple, (2012) "iOS Security" [Online]. http://images.apple.com/ipad/business/docs/iOS_Security_May12.pdf

Li, Z., Tian, J.-F. and Wang, F.-X. (2009) “Sandbox System Based on Role and Virtualization,” The First Int. Symp, Inf. Eng. and Electr. Commer., Ternopil, pp. 342—346.

LINFO, (2005) "Proprietary Software Definition," [Online]. LINFO, http://www.linfo.org/proprietary.html

Lockheimer, H., (2012) "Android and Security," [Online], Android, https://googlemobile.blogspot.com/2012/02/android-and-security.html

Malanik, D. and Kouril, L., (2013) "Honeypot as the Intruder Detection System", In Proceedings of the 17th WSEAS International Conference on Computers, Kos (GR), pp. 96-101.

Malanik, D. (2010) "Nature Behavior in Stochastic Extreme Finding Methods", Annals of DAAAM for 2010 & Proceedings ofthe 21st International DAAAM Symposium, Vienna, pp. 1205-1207.

Moore, G.E. (1965) “Cramming more components onto integrated circuits,” Electronics, vol. 38, pp. 4—7.

NVIDIA, (2010) "The Benefits of Multiple CPU Cores in Mobile Devices", [Online]. http://www.nvidia.com/content/PDF/tegra_white_papers/tegra-whitepaper-0911a.pdf

Pettey, C. and Meulen, R. (2012) "Gartner Says Worldwide Sales of Mobile Phones Declined 3 Percent in Third Quarter of 2012; Smartphone Sales Increased 47 Percent", [Online], Gartner Inc, http://www.gartner.com/newsroom/id/2237315

Raphael, R., (2012) "Exclusive: Inside Android 4.2’s powerful new security system" [Online]. http://blogs.computerworld.com/android/21259/android-42-security

Sarga, L. and Jasek R. (2012) “User-Side Password Authentication,” Proc. of 11th Int. Conf. on Inf. Warf. and Sec., Laval, pp. 237—243

Scarfone, K., Jensen, W. and Tracy, M. (2008) "Guide to General Server Security" [Online]. http://csrc.nist.gov/publications/nistpubs/800-123/SP800-123.pdf

Silberschatz, A., and Galvin, P. (1994) "Operating System Concepts", Boston: Addison-Wesley.

Voznak, M., Rezac, F. (2010) “Threats to voice over IP communications systems,” WSEAS Trans. Comp., vol. 9, pp. 1348—1358.

Wang, Z. and Stavrou, A. (2012) "Google Android Platform: Introduction to the Android API, HAL and SDK", [Online]. http://cs.gmu.edu/~astavrou/courses/ISA_673_S12/Android_Platform_Extended.pdf

and a PhD in Artificial Intelligence (Maastricht University). Tim's research focuses on the operations-technology interplay in network-enabled Command & Control systems and in offensive cyber operations.

Virginia A. Greiman Professor of Cyber law and Cyber Security at Boston University and holds academic appointments at Harvard University Law School and the Kennedy School of Government. She served as a diplomatic official to the U.S. Department of State in Eastern Europe, Asia and Africa and has held several high level appointments with the U.S. Department of Justice.

Alan Herbert comes from East London. Alan has studied at Rhodes University. He has completed Master of Science at Rhodes University in 2014 in Computer Science. Major field of study: Networks, Security, Network Simulation and Electronics. Currently studying for PhD in Computer Science and Electronics at Rhodes University and supervised by Prof. Barry Irwin.

Steven Hersee is a PhD student at the Information Security Group at Royal Holloway, University of London. He has previously served in the Royal Air Force and his primary interests are in the geopolitics of cyber security and the different and

Barry Irwin is an Associate Professor in the Department of Computer Science at Rhodes University, South Africa. He established and has led the Security and Networks Research Group (SNRG) since its founding in 2003. He holds a PhD and a CISSP. His current areas of research include network traffic analysis, data visualization and webserver malware.

Suhaila Ismail is a PhD student in Information Assurance Group at the UniSA. She received her M.Sc. in Information Security and Computer Forensics from UEL, UK. Currently she is involved in research of Critical Infrastructures and SCADA Systems Security. She has published papers in Computer Forensics, Privacy, Education and SCADA Systems Security.

Victor Jaquire has been in ICT and Information security for 18 years within government and the private sector focussing on information security strategy, performance management, business management and development, and operations. His professional certifications include CISSP, CISM and CCISO. He is presently in the process of completing his Masters thesis in Cyber Security.

Roman Jasek is head of department of Informatics and Artificial Intelligence at the Faculty of Applied Informatics in Tomas Bata University in Zlin, Czech Republic. He deals with the security of business information systems and security applications on the mobile platform. The team, working under his direction, is deals with industrial applications using artificial intelligence.

Trishee Jobraj is currently an Information Security Liaison at Transnet SOC and has over 9 years' experience in the IT Support, Internal & External Audit and Information Security. She also manages the Membership and Marketing Portfolio for ISACA SA. Her qualifications include a BSC degree in Computer Science, ITIL Foundation, CISA and CRISC.

Victor Kebande is a PhD researcher at the University of Pretoria in the field of Cloud Forensic Readiness at the department of computer science, University of Pretoria. He is a member of institute of information technology professionals of South Africa (IIPTSA) and an active member of Information and Computer Security Architectures(ICSA) research group. His research interest are in cloud forensics and internet security

Mohamed Khan is a senior analyst at Transnet. He is passionate about using statistics to help business deliver value through analysis of big data. Author of one book and a frequent speaker, his combination of actuarial science and information security give him a unique ability to find novel ways to analyse data.

Sam Lefophane is currently a Smart Card development engineer in the Information Security Unit of Modelling and Digital Science department at the CSIR. Sam's research focus is in RFID signal processing, hardware security, standardisation and intelligent systems.