The Rise of Android Banking Trojans

The Rise of Android Banking Trojans

Andrea Atzeni, Fernando Díaz, Francisco López, Andrea Marcelli, Antonio Sánchez, and Giovanni Squillero[1]

A. Atzeni, A. Marcelli and G. Squillero are with Politecnico di Torino — Department of Control and Computer Engineering, Corso Duca degli Abruzzi 24, 10129 Torino, Italy. E-mail: {shocked, andrea.marcelli, squillero}@polito.it
F. DíazA, F. López, and A. Sánche are with Hispasec Sistemas S.L., C/ Trinidad Grund 12, 1A-1B, 29001 Málaga, España. E-mail: {fdiaz, flopez, asanchez}@hispasec.com

As soon as banks started developing mobile applications to enable users to perform financial activities online, cybercriminals started developing ways to penetrate this new channel and perform illicit transactions. Indeed, for criminals it is easier to exploit the scarce end-user security awareness and attack individual clients' devices, rather than directly target banks portals.

Mobile applications are commonly called "apps", while malicious programs that hide their intention under an apparently legitimate behavior are known as "Trojans". *Banking Trojans* are written with the specific purpose of stealing confidential information from victims' bank accounts through online payment services. They are a threat so common among bank apps, that most sources simply refer to them as "bankers". Attackers commonly exploit social engineering techniques, inducing users to visit hostile websites and install malicious applications; alternatively, the malware can be spread both through official (i.e., Google Play) and unofficial app stores.

This article surveys the bankers' history, their evolution and mode of operation, and highlights the countermeasures eventually introduced to stop their spread.

History

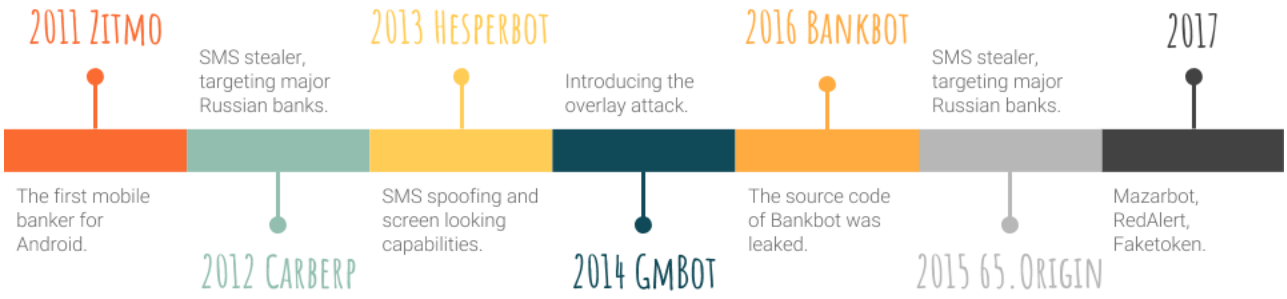

Figure 1. A timeline showing the evolution of Banking Trojans, from the first sample Zitmo in 2011, till the latest variants in 2017.

The ancestor of all banking Trojans is *Zeus*, a PC malware created in 2006, which managed to compromise over 3.5M devices in US and created one of the history largest internet-connected network of infected devices. *Zitmo*, an abbreviation for "Zeus in the mobile", was the first banker for Android: it emerged later, in 2010, and was devised to work in symbiosis with the desktop version and intercept two-factor authentication messages.

The year 2012 saw the rise of the *Carberp* family, a trojan devised to steal SMS messages and upload them to a remote server. In June 2013, the source code was leaked, and new variants of the malware targeted U.S.,

* This article is based upon work from COST Action CA15140 'Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO)' supported by COST (European Cooperation in Science and Technology).

Europe, and Latin America. Then, in 2013, the new banker *Hesperbot* appeared, targeting users in Turkey, Czech Republic, Portugal and the United Kingdom.

The forefather of a whole new generation of Android banking Trojans was *GMBot*, the first to exploit the *overlay attack*, tricking users into entering their access credentials into a fraudulent window and providing the attackers enough information for illicit money transfers out of their accounts. Later, in June 2015, *Android.Bankbot.65.Origin* was discovered in Russia; the attack reached 100,000 Sberbank users, and the state-owned company reported losses for over $35 million the same July.

In December 2016, the source code of *Bankbot* was released in a forum by a user named *Maza-in*, and it showed similarities with other well-known malware, such *MazarBot* and *RedAlert*. Since then, the number of its variants skyrocketed: in 2017, Trojans of the Bankbot variety, "Bankbots" for short, represented the most prominent threat in the Android ecosystem.

Bankbots steal users' credentials by accessing other apps private data or by displaying a fraudulent login page on top of legitimate banking apps, then they are uploaded to a server, which operates as an administrative panel, and it is used to constantly update the malware behavior. Later variants ascribable to the family included *Faketoken*, which is capable of running overlay attacks for about 2,000 financial applications.

At the same time, anti-detection techniques were growing in sophistication and effectiveness. For example, *Loapi* moved the malicious code outside the application, in a code module downloaded and executed at runtime, and delayed the execution waiting for commands from a remote server, thus eluding most detection tools. Other anti-malware detection techniques involve the identification of the running system and applications. Indeed, if malware gets aware of being in a sandbox, it can impair the analysis by not showing malicious behaviors. These adapting anti-detection tactics highlight the capability to circumvent fresh security countermeasures and call for the development of promptly evolving security strategies.

Banking Trojans' modus operandi

While hundreds of different banker variants exist, and new ones are developed daily, most of them share significant common traits. Figure 2 summarizes the three main phases of a banker attack: *the infection*, where the malware firstly infects the user's device, *the persistence*, to ensure seamless operation on the device, and finally *the attack*, where user's private bank information is stolen, and illicit payment transactions are carried out.

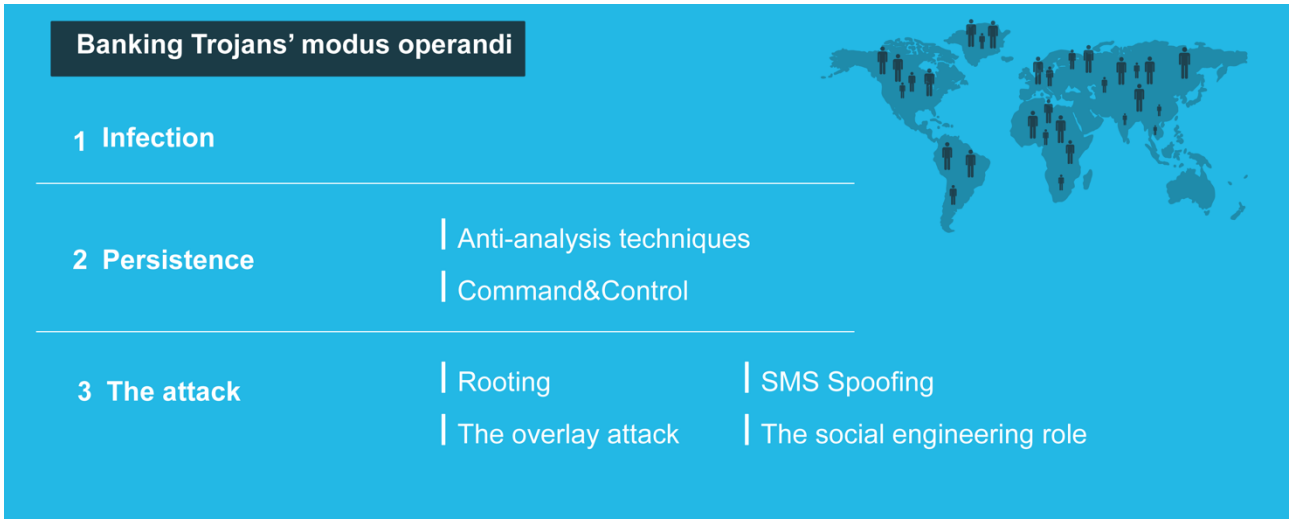

Figure 2. Schema of the three main phases of a banker trojan way of action: infection, persistence and the attack.

PHASE I: INFECTION

The most common vectors exploited to infect a device are malicious web pages and legitimate app stores. In the former case, social engineering attacks are used to cheat users to visit hostile websites, and then a malicious JavaScript code infects the device downloading the malware. In the latter case, the users' inexperience is exploited: users commonly perceive Android app stores, both the official and the third-parties ones, as trusted sources, despite most of them offer an "open market model" where applications are distributed with no preliminary check. In the past, even though bankers like *Acecard* and *Marcher* were removed within days after having been reported, they still managed to infect thousands of users.

The Android security model relies on the "least privilege principle", that is, each installed app is provided with the minimum capabilities to guarantee its functionalities, whereas the access to additional sensitive resources, is limited by the grant of specific permissions. In order to assist user granting the correct privileges, the operating system classifies permissions into four groups according to their dangerousness, but the ultimate decision depends on the user perception of risk [1]. Social engineering techniques and targeted vulnerabilities could be used to trick users for unintentional permission granting, trespassing the security model.

PHASE II: PERSISTENCE

Once installed, recent Trojans try to hide both from users and anti-virus, and to achieve persistence on the target device for operating seamlessly. While a variety of solutions are applied, a standard technique is to simply hide the application icon from the list of installed apps; alternatively, if the malware obtains administrative privileges, it may try to install the bankers as a system app, making its removal far more complex, as it would require administrative privileges as well.

In order to escape from antivirus (AV) detection, attackers commonly use anti-analysis tricks and exploit complex Command&Control infrastructures to remotely control their malware.

Anti-analysis techniques

Among the various anti-analysis techniques, *triggers* are very practical: the app shows a non-suspicious behavior until a specific condition, the *trigger*, is fired. Triggers are mainly used to mislead the detection during the AV analysis, which is usually limited to few minutes of runtime monitoring. Both "time bombs" (i.e., a timer) and device reboots are commonly used as triggers.

While older bankers naively embedded the list of target banks in plain text within the source code, nowadays the information is obfuscated and dynamically collected from remote servers. For instance, in October 2017, malware researchers discovered that *Tornado FlashLight* used the hash of the package names to identify the target banking apps.

Command&Control

To control their Trojans, attackers setup complex infrastructures, called *Command&Control* (C&C), where infected devices receive instructions (i.e., the *command*) from a central entity (i.e., the *control*). For instance, the banker family *Twitoor* exploits tweets published by a Twitter account, while samples from the *Charger* family exploit the Firebase Cloud Messages. Being able to communicate with the malware, attackers can change the targets of the attacks.

PHASE III: THE ATTACK

Bankbots could achieve their malicious behavior in different ways, mainly depending on the level of access rights. If the highest privileges have been granted, the malware could subvert the Android security model and steal confidential data directly from other applications key store, like login passwords and bank card details. Furthermore, it could maliciously tamper the web traffic and dynamically alter the web pages content to redirect users to fraudulent websites. In the most advanced cases, the malware can exploit unauthorized rooting capabilities to take full control over the device. On the other hand, if such access rights are not granted, overlay

attacks and screen recording could be applied. Furthermore, bankers commonly have the capability of SMS spoofing, that is to intercept text messages to bypass SMS-based 2-factor authentication and send SMS to activate paid services. Finally, almost all the malware exploit a combination of social engineering techniques in order to gather users' sensitive information.

The following sections will cover in detail the role of technical aspects such as *rooting*, the *overlay attack* and the *SMS spoofing*, as well as the *social engineering* techniques.

Rooting

The procedure of *rooting* allows users to take full control over their devices by obtaining *superuser* access rights, that is, the highest possible privileges. Although rooting is commonly seen as a way to extend the capabilities of an Android device, it has substantial security implications as it overcomes the Android basic security principle. Through rooting, malware can obtain administrative rights without the user agreement by exploiting a vulnerability in the operating system.

Some rooting techniques are quite ingenious, like the one used in DRAMMER [2] that exploits an issue with new generation DRAM chips. Although the original version of Bankbot did not exploit unauthorized rooting capabilities, most of the collected samples do include checks to verify if the device is rooted; then, they exploit administrative rights for directly stealing credentials in the key store of other apps or for tampering web traffic. On March 2016, researchers identified a Trojan named *Triada*, which gains unauthorized superuser privileges. The application behaves like a Dropper, using a remote server to get the list of new applications to download, eventually installing new Trojans. One of them, *Triada.p/o/q* tampers URLs loaded in a browser, especially targeting online banking platforms.

The Overlay Attack

The *overlay attack* is a common strategy adopted by malware authors and consists in drawing a fraudulent screen on top of a target application as soon as a victim clicks on a link on a legitimate site or launches a legitimate app. The ability to replicate high-quality user interfaces is essential for the success of an overlay attack. For instance, Figure 3 compares two login pages for the Skype application; although graphically different, they have the same look and feel of the original Skype and could easily mislead the user. Even for the most careful one, recognizing the legitimate version is hard due to the frequent graphical updates and variants that applications show.

Commonly Bankbots monitor which apps are installed on the infected device, then, if sensitive applications are detected (e.g., Online Banking, VPN, Social Networks) the malware redraw a similar login screen on top of the legitimate ones. In most of the cases, the user is not able to distinguish the fraudulent version from the original one and will insert its credentials which will uploaded on a C&C server. Figure 4 shows an example of overlay attack at the Google Play Store app: a malicious pop-up is displayed over the app, asking the credit card number.

The potential issues with overlay have been known since 2011, but the attack was led to fame only in 2017, when the *Cloak&Dagger* exploit showed how granting only two permissions (i.e., System Alert Window and Accessibility Services) could result in a very effective technique to steal user credentials [3]. The issue was targeted in Android *Mashmellow* (*v6.0*), and later Android *Oreo* (*v8.0*), which includes a visual notification whenever an overlay is displayed, allowing the user to easily dispose it. However, since the attack works seamlessly in all the Android versions prior to *Oreo*, the number of attackable devices is still huge.

Recent malware exploiting overlay attacks include *Charger*, a banking Trojan masquerade as a flashlight app removed from the Google Play Store after 5,000. On November 2017, *Smart AppLocker* was found on Google Play, posed as a legit app that secure device applications using a PIN code, but it was the first known malware to exploit the new *Toast overlay attack*; at the time of removal it has been installed between 100,000 and 500,000 times.

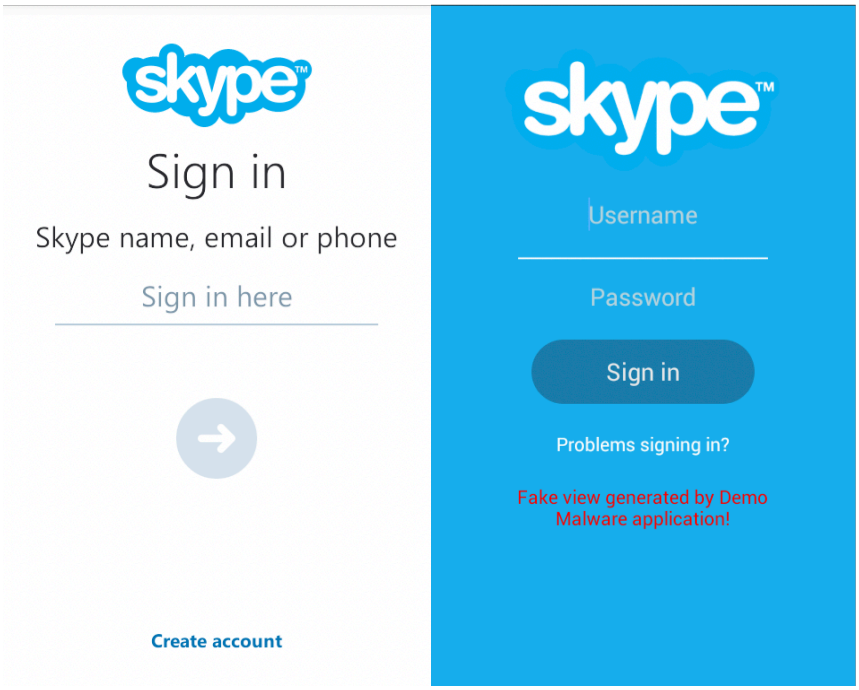

Figure 3. Comparison between the original Skype login page (on the left), and the one prompted during an overlay attack (on the right). Pictures are generated using the android-overlay-malware-example [4].

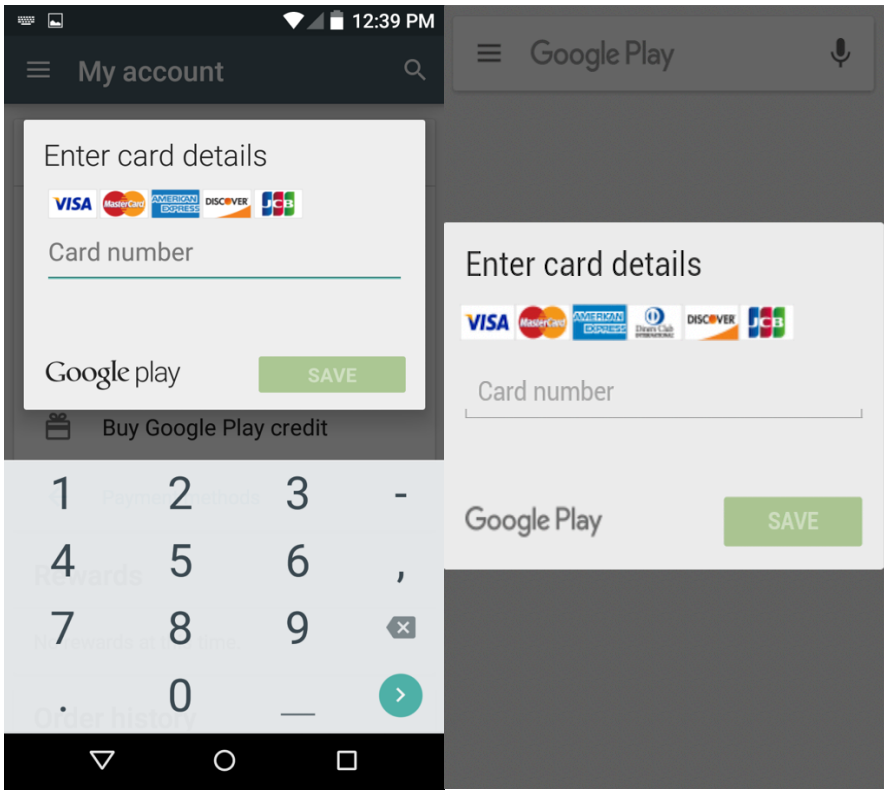

Figure 4. Overlay attack on the Google Play Store. Picture on the left shows the original user interface, while the one on the right shows the fraudulent pop-up displayed during an overlay attack. In this case the attackers were able to replicate the same look and feel of the original application.

SMS Spoofing

As online banking increased popularity, bank institutes adopted countermeasures to ensure the security of online operations and stop banking frauds. Since the adoption of the two-factor authentication (2FA), where for each operation a one-time password (OTP) is generated, attackers deployed features to steal OTPs. Since the first versions of Bankbot, Trojans do include SMS spoofing capabilities and are able to receive SMS and send the OTP to the C&C server.

The Social Engineering Role

The term "social engineering" refers to the psychological manipulation of users, tricking their confidence for wicked purposes, for example the gathering of sensitive information. Bankbots regularly exploit social engineering to circumvent the technical security measures imposed by the operating system, and more in general by the Android ecosystem: malware authors commonly deceive users to visit harmful web pages, grant risky permissions, and insert sensitive information in fraudulent login or payment forms.

Countering the threat

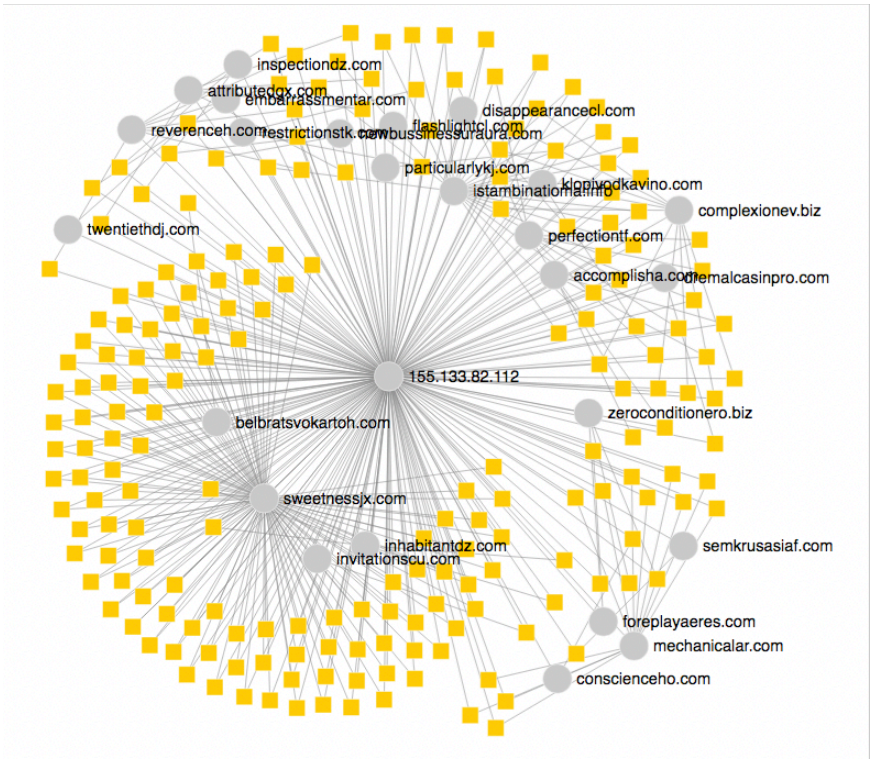

Figure 5. A graph created from the analysis of malicious domain (rounded nodes) contacted by a family of Banking Trojan (square nodes). Samples are collected from November 2016, until February 2017. In details, 25 domains resolve to one IP address (155.133.82.112), the C&C, which controls about 200 unique trojans.

Malware analysis is a typical example of the cat-and-mouse game: as new anti-virus techniques are developed, malware authors respond with new ones to hide their detection. Usually, AVs analyze applications using both static and dynamic techniques, getting an insight on the real application behavior. If a malicious pattern is shown, AVs use signatures to detect known samples. However, writing effective signatures that precisely identifies malware samples, without generating either false positives or negatives, is a challenging task. Recently, new detection methodologies based on machine learning techniques, like ensemble clustering methods and deep neural networks, gained popularity. Their effectiveness is directly related to the choice of the features (i.e., attributes which allow to transform a malware sample into a point in a multidimensional space), and the generality of samples used to train the models. Moreover, as malware constantly evolve, adversarial attacks are possible against such automated solutions [5]. Although machine learning cannot still

entirely replace the need of manual analysis, its adoption is a significant advantage and very useful in a number of cases [6].

To detect banking Trojans, AV companies deployed specific solutions to address this kind of threat: differently from other malware variants, Bankbots share many general features, like the widespread usage of overlays, and the need of a C&C infrastructure. Matter of factly, many of the known Bankbot samples share the same name of web pages used to contact the C&C server (i.e., *log.php*, *commands.php*, *config.php*), which can be detected by simple string matching. Moreover, applications with overlay capabilities are automatically tagged as suspicious, and further analyzed.

Despite new malware variants can be generated at a high pace, changing the C&C infrastructure is more challenging since devices already compromised should be updated to preserve the communication: malware authors commonly use *Domain Generation Algorithms* to keep their networks alive, making difficult for law enforcement to shut down them. Tracking the evolution of malicious domains, and the way in which applications interact with them, is an effective method to unveil new malware variants. For instance, Figure 3 shows an example of a Bankbot variant that contacts the very same Poland host, recognizable by its IP address, using tens of different domain names.

How to protect your Android device

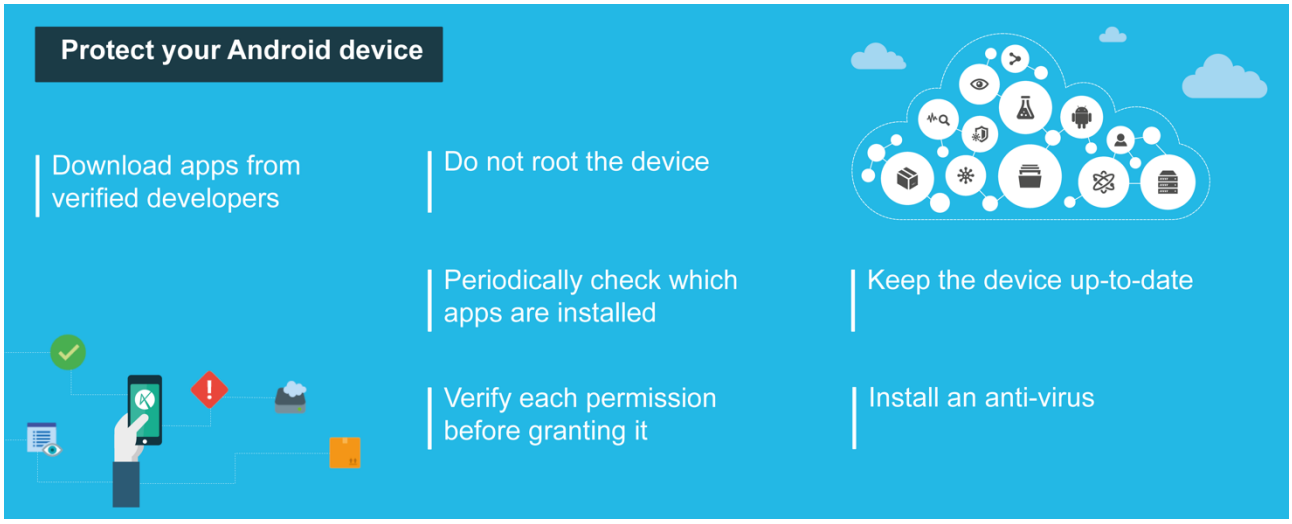

Figure 6. Infographics on how to "Protect your Android device".

Since several third-party app stores adopted a policy which foster fast applications availability over careful checking, users should avoid downloading apps from third party sources, using only the official Google Play Store to install new applications, and preferably **choosing only apps from verified developers**, with a high rating, and a significant number of downloads. Users should also be extremely cautious with applications downloaded directly from any website, especially being aware that malicious actors often exploit social engineering techniques to cheat their victims. In particular, they should never click on links embedded in SMS, MMS, or even email sent to the mobile phone: even if those messages look legitimate, it is better to go directly to the website and verify their validity. Once a new application has been installed, users should always **verify each permission before granting it**, and if an application is asking more than what is meant for, it is preferable avoiding its installation. Obviously, users should **not root their devices**, and should understand that administrative rights are remarkably powerful: granting them can permit full control over the device.

Users need to **periodically check which applications are currently installed**, removing non-necessary ones. Several malware try to hide them self, for example removing the main application icon: use the *Application Manager*, in the *Settings* menu to check which ones are installed. Whenever it is needed to remove an application with administrative rights, users need to use the *System update* options, in the *Security* menu.

Finally, all users should **install an anti-virus** able to detect and block known malware patterns, and always **keep** their **device up-to-date**.

Read more about it

1. A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android permissions: User attention, comprehension, and behavior," in *Proceedings of the eighth symposium on usable privacy and security*. ACM, 2012, p. 3.
2. V. van der Veen, Y. Fratantonio, M. Lindorfer, D. Gruss, C. Maurice, G. Vigna, H. Bos, K. Razavi, and C. Giuffrida, "Drammer: Deterministic rowhammer attacks on mobile platforms," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1675– 1689.
3. Y. Fratantonio, C. Qian, S. Chung, and W. Lee, "Cloak and Dagger: From Two Permissions to Complete Control of the UI Feedback Loop," in *Proceedings of the IEEE Symposium on Security and Privacy (Oakland)*, San Jose, CA, May 2017.
4. S. Paganoni. "Android Overlay Malware Example." Internet: https://github.com/geeksonsecurity/android-overlay-malware-example, Jun. 18, 2015 [Aug. 31, 2018]
5. K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," in *European Symposium on Research in Computer Security*. Springer, 2017, pp. 62–79.
6. D. Ucci, L. Aniello, and R. Baldoni, "Survey on the usage of machine learning techniques for malware analysis," *arXiv preprint arXiv:1710.08189*, 2017.

About the authors

Andrea Atzeni (shocked@polito.it) holds a MSc and a Ph.D. in Computer Engineering, both from Politecnico di Torino. He is currently Senior Research Assistant in the TORSEC Security Group at the Politecnico di Torino. In last fifteen years he contributed to a number of large-scale European research projects under the FP5, FP6 and FP7 and CIP programmes. He addressed, among the others, the definition of security requirements in multi-platform systems, mobile security, modelisation of user expectation on security and privacy, security specification, risk analysis and threat modeling for complex cross-domain architectures, development of cross-domain usable security, digital and cloud forensics, development and integration of cross-border authentication mechanisms, malware analysis and modelling.

Fernando Díaz (fdiaz@hispasec.com) is a malware analyst and software engineer. Currently he is a B.Sc. student in Health Engineering at the University of Malaga and he is working at Hispasec Sistemas as a Security Engineer. Daily work focuses on automated malware configuration extractions, distributed analysis environments, and developing software for the Koodous platform. His research includes analysis of new malware families and IoT malware.

Francisco López (flopez@hispasec.com) holds a Ms.C. in Computer Engineering from the University of Cádiz. Currently he is a System Engineer and Security Manager at Hispasec Sistemas, where he is responsible of the Koodous project. Overall, he has a 10 years' experience in the computer security industry, and his research interest includes new ways to to assist the automatic detection of Android malware and spam fraud campaigns.

Andrea Marcelli (andrea.marcelli@polito.it) received his M.Sc. degree in Computer Engineering from Politecnico of Torino, Italy, in 2015. Currently he is a Ph.D. student in Computer and Control Engineering at the same institute and member of the CAD group. His research interests include malware analysis, semisupervised modeling, machine learning and optimization problems, with main applications in computer security.

Antonio Sánchez (asanchez@hispasec.com) is malware analyst and research engineer. He received his M.Sc. in Computer Science at Universidad de Jan (Spain) in 2013 and since 2012 he is working as a security engineer at Hispasec Sistemas. His daily work focuses on the improvement of systems for the automatic detection and analysis of malware samples, while his research interests include new techniques for storage, recovering and correlation of big data.

Giovanni Squillero (squillero@polito.it) received his M.S. and Ph.D. in computer science in 1996 and 2001, respectively. He is an associate professor of computer science at Politecnico di Torino. His research mixes the whole spectrum of bio-inspired metaheuristics with computational intelligence, machine learning, and selected topics in electronic CAD, games, multi-agent systems. Other activities focus on the development of optimization techniques able to achieve acceptable solutions with limited amount of resources, mainly applied to industrial problems. Squillero is a member of the *IEEE Computational Intelligence Society Games Technical Committee*.