riscure

Analyzing the security of Cloud-Based Payment apps on Android

Since Google enabled Host Card Emulation based solutions in the end of 2014, many banks have adopted this technology for Cloud Based Payments. HCE is an evolution of smart card payment technology as standardized by EMVCo. An HCE app on the mobile phone uses the NFC interface to emulate a payment card and utilizes the existing EMV payment network. Being a full software-based solution, it is possible to run HCE on any phone having an NFC interface, typically most Android phones.

Since the hardware security of the host CPU inside the smartphone is much weaker than that of a smart card, an HCE app running in the phone must provide additional security. HCE partially relies on protocol features enabled by the always-on network connection. An additional level of security is needed and can be achieved by hardening the app.

In this paper we study the use of security features at a large scale, by analyzing all HCE apps in the Google play store. Although we refrain from a security evaluation of individual app strengths, we can judge the overall state and priority of HCE security given by the payment industry.

The payment schemes have promoted and stimulated security by means of advisory and approval of payment apps. They proposed security guidelines and risk mitigation strategies. Most solution providers are using evaluation labs to challenge the strengths of their software components, and many banks verified the robustness of their payment wallets.

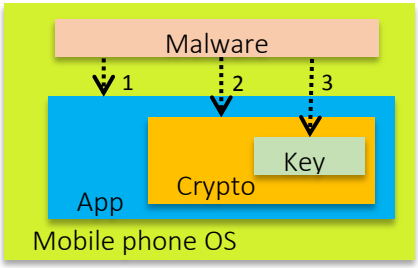
Although there is a wide range of attack possibilities for an attacker with access to the device, the most significant threat is a remote attack using malware or malicious applications (IANS, 2018). Operating system (Android) vulnerabilities are frequently discovered and experts agree that given enough time, all platforms can be rooted. This means that malware targeting a smartphone (e.g. through malicious attachments, infected web pages or malicious application in the app store) can gain root privileges (Avraham, 2015), (CheckPoint, 2016), (Artenstein, 2017) and manipulate the behavior of genuine apps. An attacker might also modify the original payment application and distribute it using phishing.

Remote malware threats

There are three increasingly dangerous threat scenarios for malware manipulating a payment app:

1. Malware manipulates transaction data and changes payment details. For instance, the user may be charged for another amount than displayed.

2. Malware uses the crypto engine to sign arbitrary transactions. In this scenario the user does not need to be involved, and the transaction can run in stealth mode.



3. Malware is able to lift, export, and clone the complete app, or extract and export the cryptographic keys or payment data. In this scenario transactions can be run from a remote entity mimicking the HCE app.

The biggest risk arises when malware is present and active on a large number of user phones, and many parallel fraudulent transactions are run simultanuously in a short time frame, hence limiting the effectiveness of back-end mitigations or fraud detection mechanisms.

Apart from unfocused scalable attacks with malware, phishing and other focused techniques can be used to access small targeted group users.

Software security features

Even on an open platform, where untrusted apps can be loaded, it is possible to protect such mobile apps against eavesdropping or manipulation by malware. To protect against threats, there are a number of security mechanisms that HCE mobile payment applications should implement. These mechanisms are often complementary to each other, meaning that security relies on the combination of such countermeasures. Layered security relying on multiple countermeasures can significantly extend the time required for developing a successful scalable attack.

1. Anti-analysis, such as native and Java code obfuscation, increases the effort required to reverse engineer the application, and understand sensitive information in an app. This is the first line of defense, and if done well it may slow down an attacker by several weeks.

2. Anti-rooting checks help detect rooting or locally installed rooting tools. If effective, it prevents the application from running on a rooted mobile device, and mitigates less sophisticated malware attacks.

3. Anti-instrumentation detects debugging and function hooking used to observe runtime behavior, and control the application during an attack. This feature slows down the analysis of an application and the development of an exploit.

4. Anti-tamper mitigations protect the code and data of the application against manipulation by an attacker. Changes to the program code would be detected and prevent the app from further execution, thereby preventing security degradation.

5. Anti-cloning prevents an application and its data from functioning correctly after being cloned to another device. This is done by binding the data to a specific device, and by preventing execution in another or an emulated environment.

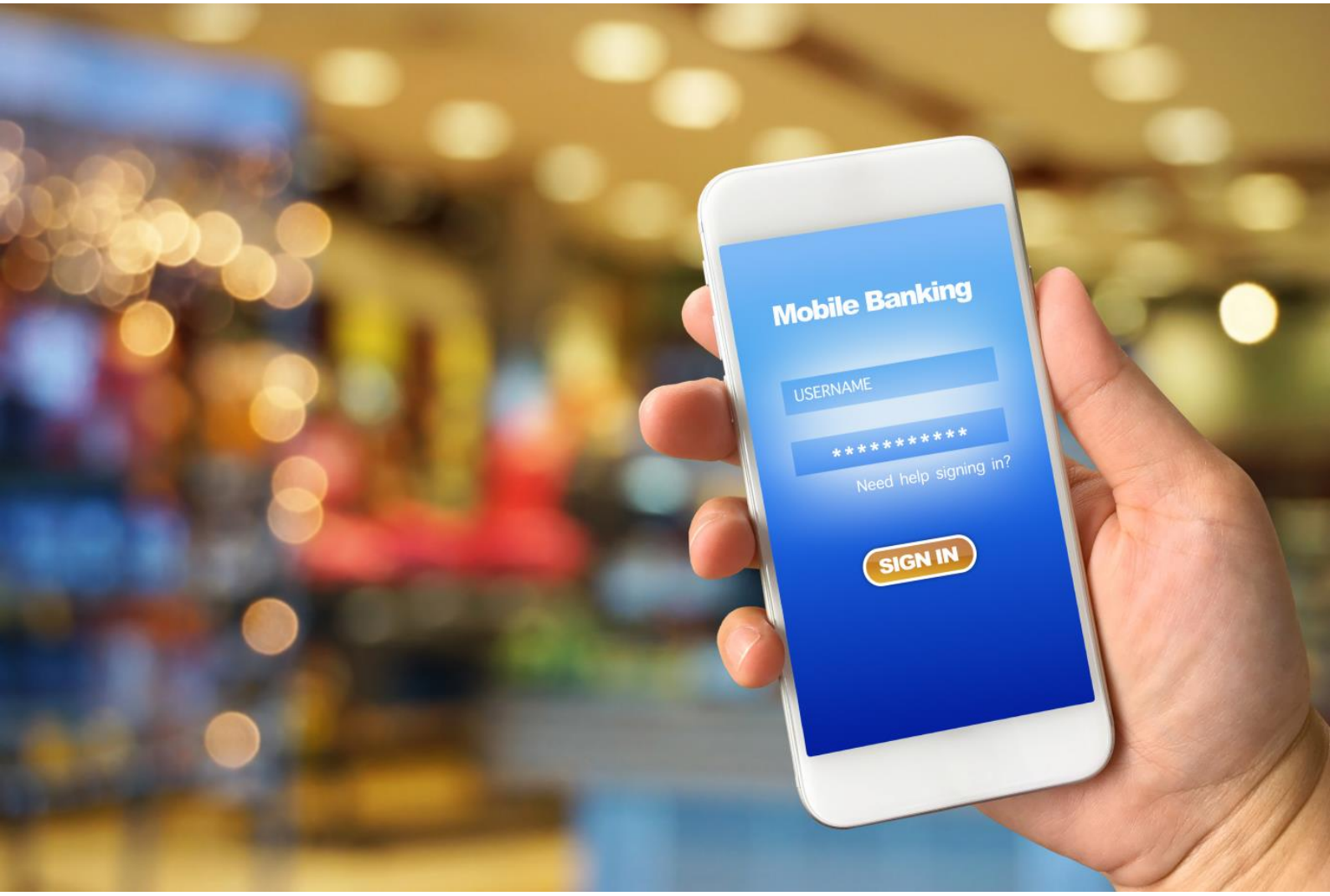
6. Anti-key-recovery hides the cryptographic keys. Typically, this is done with White-Box Cryptography, an implementation strategy spreading key bits over large portions of code, making it very hard to extract the keys by code inspection.

A strong application implements many or all of these features. On the other hand, missing features could render the effect of other features useless. None of the features alone can be sufficient to stop a sophisticated malware for too long. While Anti-analysis can slow down the attacker developing the malware, the effectiveness of the measure is not sufficient for stop a determined attacker. Additionally, it is possible that the application is not fully covered or some data is excluded from the obfuscation. The Anti-rooting prevents the use of the application on a rooted device; however anti-rooting checks for common root tool files can be circumvented by renaming the files. The Anti-tamper feature is used to prevent manipulation of code and data, however without Anti-analysis measures Anti-tamper can be identified and bypassed with patching or hooking. Lack of Anti-instrumentation feature enables full control of the application, no Anti-cloning allows for export of application and secret data and repetition of the attack and a lack of Anti-key-recovery might leave vulnerable keys in plain sight.

Besides the presence of these features it is important that their implementation is robust. Obviously a weak implementation can be more easily defeated. Security evaluation has proven to be an effective method to increase awareness, usage, and quality of security features. Indeed, security labs have observed improvements in the apps offered for evaluation while the market for mobile payment matured.

Apart from the software security features, solutions can be protected using security separation such as Trusted Execution Environment, or separated secure hardware. Also, risk mitigation in the back-end network can be a powerful mechnism to detect fraud. This feature is bank and payment network dependent, and can include remote attestation and statistical anomaly detection.

In this paper we focus only on the software security features as in practice such security separations are not readily available for banks but mainly for smartphone manufacturers.



Security feature investigation

We researched the level of support in existing operational HCE apps for the most essential security features. For this study we downloaded 426 HCE apps from the Android app store. These are selected by searching for all payment apps using the NFC API in Android. It is important to note that only a minority (estimated less than 100) of apps has actually undergone security approval testing, most apps are probably introduced without an independent analysis of security strengths and weaknesses.

In the past several years Riscure analyzed many HCE apps by manual inspection, and developed knowledge of applied security features, and filters to detect them. These filters check for known signatures of code performing specific actions. This enables Riscure to perform an automated analysis, at a large scale. As Android apps are publicly available through the apps store we were able to generalize, test, and improve our detection filters. While it is possible that features go undetected, we verified by random manual survey that features are detected and we are confident that our detection capabilities have a high degree of reliability.

We used the beta version of our upcoming **Touchstone** mobile software analyzer to determine which of the six security features are present in commercially available solutions. Figure 1 depicts the percentage of HCE applications with a given security feature.

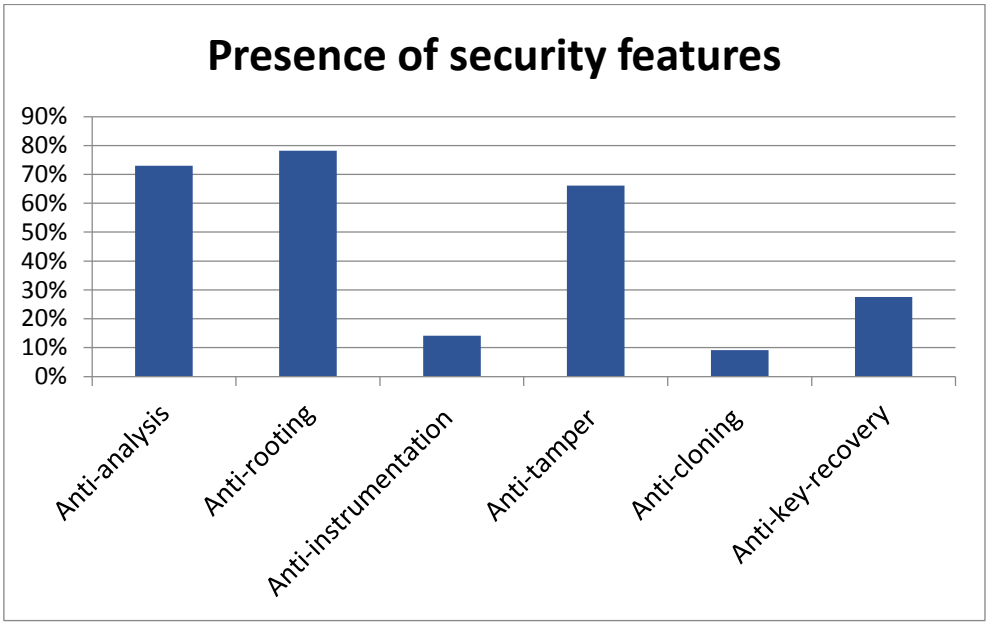


Figure 1 Presence of security features detected by Touchstone in 426 publicly available HCE solutions

The feature support ranges from 9% for **Anti-cloning** to 73% for **Anti-analysis**. As mentioned in the description of security features, **Anti-analysis** and **Anti-rooting** alone are not sufficient to protect the application from malware. It is worrisome that **Anti-cloning** and **Anti –instrumentation** are hardly used and the same applies to **Anti-key-recovery**.

The combination of security features is necessary for effective and robust security and therefore we analyzed the number of security features present in each app. The results are disturbing for financial institutions and users. Almost 10% of HCE apps do not implement any security feature whatsoever. More than half implement fewer than 3 features, and less than 1% implements all six features.

Even for apps that support multiple security features some caution is in order. There are various degrees of quality in the implementation of security features, and the robustness of each feature can only be confirmed after a proper security evaluation.

Based on our experience with mobile security testing, we know that apps with only few security features can be broken in a matter of days. This means that adversaries who are able to load malware on phones equipped with these weak HCE apps, can make unauthorized payments at the expense of the legitimate account holder.

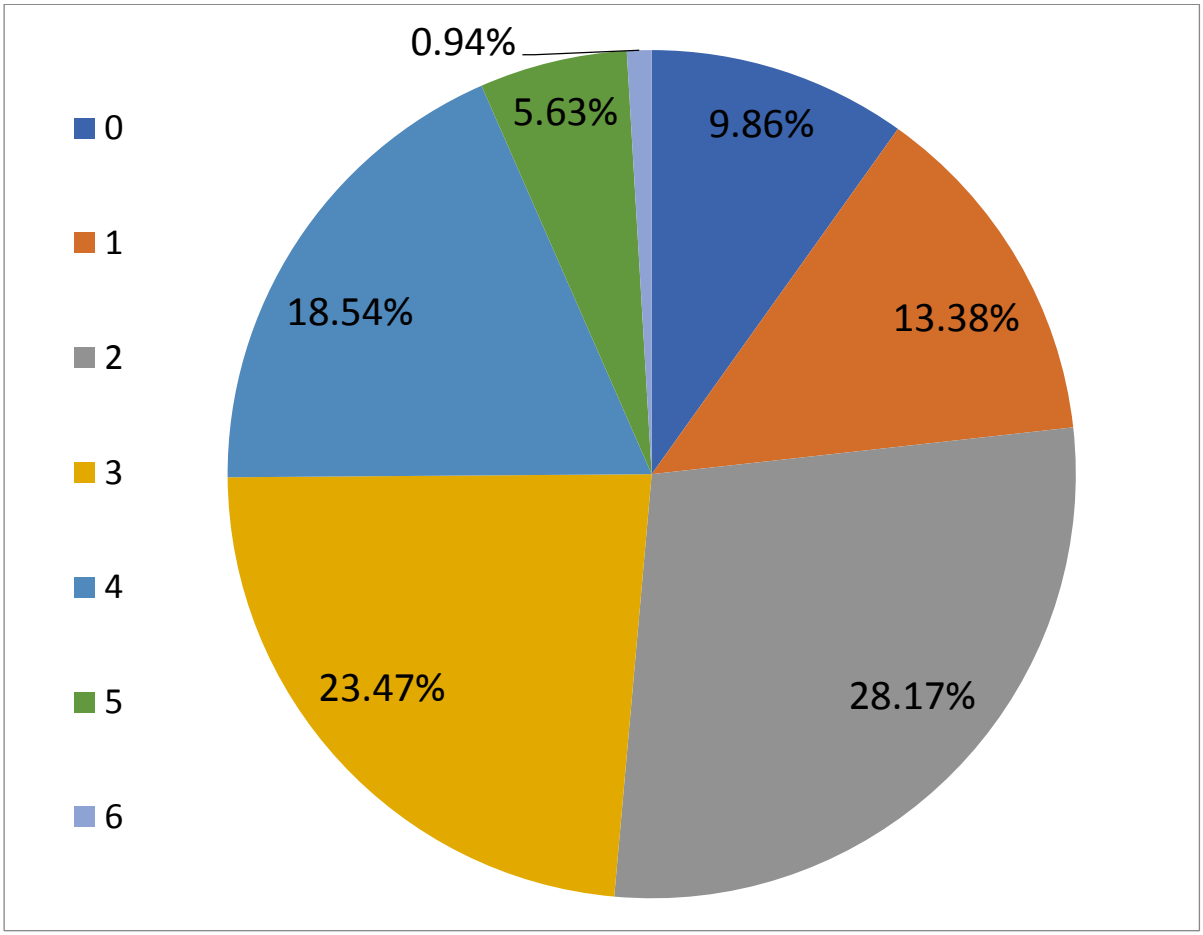


Figure 2 Percentages of HCE apps supporting from zero up to six security features

Conclusion and recommendations

Although security promotion and product approval testing accelerated secure software technology maturity, its application has not become widespread. With the exception of some well-scrutinized and hardened solutions, the general state of security feature support in current HCE apps is poor. Based on our analysis, we conclude that most of the deployed HCE apps today cannot be trusted to be secure. Account holders are at risk of fake transactions, and organisations are at risk of high cost and brand damage as a direct consequence of large scale attacks.

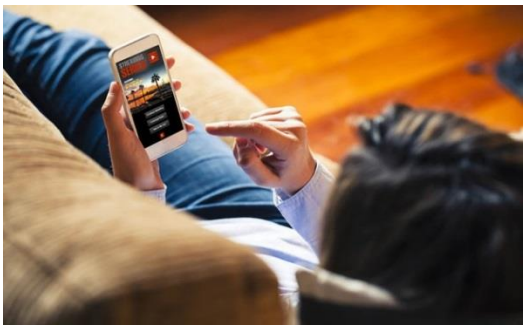
Fortunately, solutions to improve the security level are readily available. From our extensive experience with mobile application security, we provide the following advice about how to increase the security robustness and effectiveness of each of these features:

Anti-analysis can be improved by using strong obfuscation tools, which not only hide function and data names, but also encrypt strings, and reorder the control flow. Such tools are readily available, and can be effective if their configuration enables sufficient coverage.

Anti-rooting, Anti-instrumentation, Anti-tamper, and Anti-cloning become effective when applied repeatedly throughout the code. Adversaries can bypass these measures by removing them one by one, but this is largely manual work and can be labor intensive when multiple mechanisms are used with a wide coverage. Several commercial frameworks exist to harden code with these features.

Anti-key-recovery can be achieved by utilizing strong White Box Cryptography. This technology is under intense scientific research and has made significant progress in the past few years. However, the technology is sensitive to memory side channel and fault injection attacks during the execution of the cryptographic algorithms. The White Box Cryptographic implementation should contain countermeasures against these types of attacks. Several solutions are commercially available but testing of the final instances is recommended as the robustness against such side channel and fault injection attacks is solution and configuration dependent.

The security of the application can be significantly improved by security review and testing. A system level review of the design and configuration of security features, both in the phone as in the network, can help reduce conceptual risks, while testing the product provides assurance that the implementation is robust and no vulnerabilities were introduced in the software.



Riscure offers the best tools and expertise in the field of mobile security, and uses these capabilities to help developers as well as financial institutions, media and entertainment industry to improve security through advisory and evaluation.

Would you like to improve the security of your mobile application? Contact us at: inforequest@riscure.com.

Works Cited

[1] Artenstein, N. (2017). *BROADPWN: REMOTELY COMPROMISING ANDROID AND IOS VIA A BUG IN BROADCOM'S WI-FI CHIPSETS.* Exodus Inteligence.

[2] Avraham, Z. (2015). *Experts Found a Unicorn in the Heart of Android.* Zimperium.

[3] CheckPoint. (2016). *QuadRooter: New Android Vulnerabilities in Over 900 Million Devices.* CheckPoint.

[4] IANS. (2018). *Use an app for your banking? An Android Trojan could target you.* https://www.thenewsminute.com/article/use-app-your-banking-android-trojan-could-target-you-74206.

Riscure B.V.
Frontier Building, Delftechpark 49
2628 XJ Delft
The Netherlands
Phone: +31 15 251 40 90
www.riscure.com

Riscure North America
550 Kearny St., Suite 330
San Francisco, CA 94108 USA
Phone: +1 650 646 99 79
inforequest@riscure.com

Riscure China
Room 2030-31, No. 989, Changle Road, Shanghai 200031
China
Phone: +86 21 5117 5435
inforcn@riscure.com