EasyChair Preprint

№ 9181

Security Risk Assessment Model for Cryptographic Algorithms Misuse in Mobile Payment Applications

Maharshi D'arunachala Zan, Franklin Tchakounté and Tiguiane Yélémou

October 28, 2022

Security risk assessment model due to cryptographic algorithms misuse in mobile payment applications

Maharshi d'Arunachala Zan[1], Franklin Tchakounté[2], and Tiguiane Yélémou[1]

[1] University Nazi BONI, (maharshizan,tyelemou)@gmail.com
[2] University of NGAOUNDÉRÉ, tchafros@gmail.com

Abstract. Applications that run on Android have more and more vulnerabilities that often lead to disclosures of personal information. Researchers have developed approaches to detect applications that are a source of vulnerabilities. We propose a model for risk evaluation. This highlights the high rate of cryptographic misuse in mobile payment. For us, detecting it is important to assess the risk associated with the use of these APIs because this evaluation allows sensitizing developers in the use of these different cryptographic APIs.
To carry out this work, we have proposed a vulnerability analysis model that allows us to quantitatively and qualitatively assess the risks related to these misuses. The experiment was conducted using the enjarify, bytecode viewer tools. Payment applications were downloaded from Apk repositories and made usable by converting them into java classes. Also, we used rules or criteria known to be vulnerable. So during the manual analysis, if one of the rules is found in an application, it is counted and so on until the list of rules is exhausted.
Finally, from this analysis, we calculate the risks based on a proposed formula. At the end, we have grouped payment applications into three (3) categories, payment solutions (PS), payment applications that interact with bank accounts (APCB), and those that do not require a bank account (APNCB). As result, we have security risk values ranging from 0.39 - 8.39 for APCB, 0.6 - 2.67 for NBCAA, and 0.22 - 4.39 for PS.

Keywords: Android · Cryptographic API · Mobile Payment Applications · Risk Assessment · Static Analysis

1 Introduction

With the advent of digital technology, the definition of what is called "payment" has evolved rapidly. From the use of traditional methods of payment such as coins, banknotes, and cheques, there has been a shift to the digitalization of these methods of payment.This is a new type of payment supported by mobile phone operators such as Orange Money [9], Mobicash [10] in Burkina Faso and M-Pesa [11] in Kenya. Mobile payments have been booming due to the low rate

of bank penetration combined with the increase in the number of mobile phones in cities and suburbs. This allowed the development of local activities using these means of payment accessible to all (possibility of sending small amounts). Unfortunately, malevolent people take advantage of the facilities to steal (scam) money from electronic purses. Moreover, developers who strengthening the security of payment applications omit , or misuse cryptographic APIs. Hence the need to measure the impacts of these applications through tests to propose its best practice solutions. The rest of our paper is organized as follows. In section 2 we present the related work. Next, in section 3 we propose our model. After that, in section 4 we carry out the tests and present the results we have achieved. Finally, the last part is dedicated to the conclusion.

2 Related work

For risk assessment, it is necessary, on the one hand, to highlight the cryptographic APIs misuse and, on the other hand, to use tools for risk calculation.

Egele et al. [9] has reported on several applications that use cryptographic means to ensure security. They assume that static analysis could give acceptable results based on a CryptoLint tool. For this, the analysis tool is based on security rules and Androguard. Thus, for Android applications collected between May and July 2012, CryptoLint has successfully analyzed fifteen thousand one hundred and thirty-four (15,134) applications; it has also recorded two thousand six hundred and seventy-four (2,614) failures related to the thirty (30) minute timeout and seven hundred and sixty-five (765) failures (a memory overflow). As a contribution, this tool makes it possible to analyze applications, even when the developers have well implemented the cryptographic APIs.

Alexia Chatzikonstantinou et al. [10] investigate the cryptographic methods used by developers to protect so-called sensitive information. Sensitive information includes short messages, passwords, and documents. For the analysis of cryptographic means, they recommend two combined approaches, the first consists of static analysis, and the second is dedicated to dynamic analysis. On a data set of forty-seven (47) Android applications collected from June to November 2014, they obtained forty-seven (47%) percent that misuse and fifty-three (53%) percent that do not use cryptographic methods as results.

Wickert et al. [11] ,to simplify their analysis, have made a collection of two hundred and one (201) misuse of Crypto APIs, which was done in three (3) steps: collect the projects, identify them manually or with CogniCrypSAST, and add them to the directory. It is made available on the GitHub platform (MUBench). At the end of the identification (Phase 2), one of the correct uses is provided to users, which is a great contribution to the results. One of these approaches' limitation is that it is based only on the JCA library and the projects that these authors manage (non-representative sample). An effort was made to categorize

vulnerabilities (encryption, hashing,...).

In addition,there are tools that allow us to calculate the risk. These tools are based on equations (there are used to calculate the risk) and metrics (there are used to categorized the level of the risk). A comparative analysis of the risk calculation tools revealed similarities in the types of assessment: quantitative and qualitative.
For example, we have CVSS(Common Vulnerability Scoring System) [4], CVE(Common Vulnerability and Exposures) [5] , CCEVS(Common Criteria Evaluation and Validation Scheme) [6], OWASP(Open Web Application Security Project) [7], OCTAVE(Operationally Critical Threat, Asset, Vulnerability and Evaluation) [8].

Based on the advantages of risk identification and risk calculation approaches by using equations and metrics, we propose a model.

3 Our risk assessment evaluation model

3.1 Description of the model

We call MAM4MA (Misuse Assessment Model for Mobile Applications) our risk assessment model due to the misuse of cryptographic algorithms implemented in mobile applications. It is used on mobile payment applications and identifies misuse of the APIs implemented. For example, a misuse of the cryptographic APIs in a payment application for Android, results in the following criterion:

- a misuse of a symmetric cryptographic algorithm to encrypt information is the EBC (Electronic codebook), because this encryption method does not use an initialization vector.

Using the diagram below, we have two (2) outputs (X and Y). These outputs materialize for X: the number of poorly implemented cryptographic algorithms; for Y: the number of breakable cryptographic algorithms.

We use the equation 4 to assess the risk associated with the use of a cryptographic algorithm for a given system (mobile payment applications). The result obtained from this calculation provides us with a value whose accuracy is limited to two (2) digits. This value obtained has a qualitative correspondence in the table 1.

3.2 Illustration of MAM4MA

The proposed model uses the security rules as well as the values of X and Y. The diagram in figure 1 gives us the values of X and Y.

3.3 Rules or criteria used

They represent the comparison criteria of the implemented cryptographic APIs in mobile payment applications. They are distributed in four (4) categories. They are : use of unsuitable algorithms (1), bad implementations (2), use of wrong keys (3), use of wrong cryptographic parameters (4) [12] .

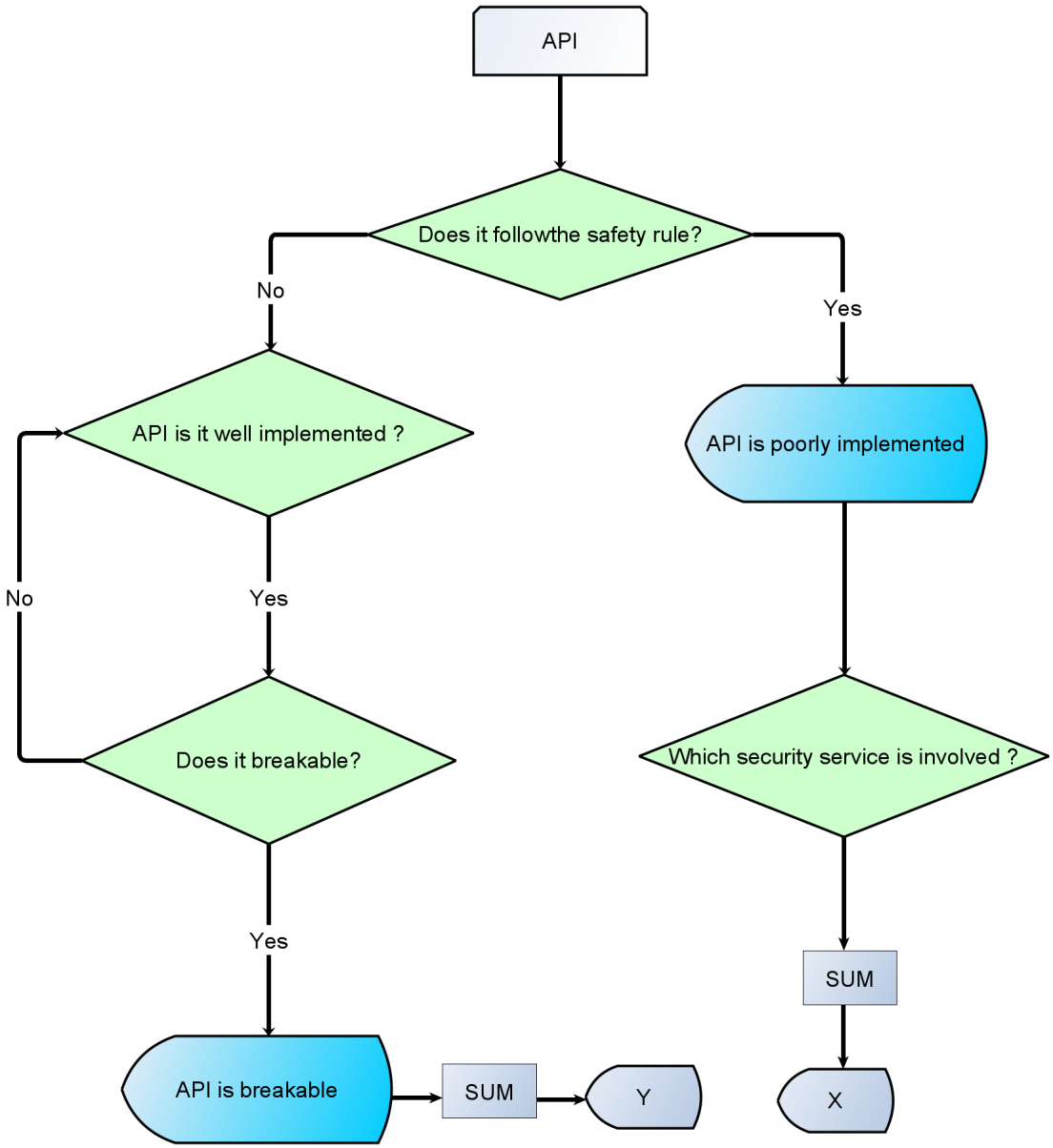

Fig. 1. Misuse Assessment Model for Mobile Applications (MAM4MA)

3.4 Mathematical justification of model

The model (MAM4MA) proposed and used for the analysis of vulnerabilities due to the misuse of different cryptographic algorithms. This model is based on equations and the diagram presented in fig 1. In the following we provide the equations used.

For n_1, n_2, X, Y $\in \mathbb{N}$, we have :

the risk associated with the use of a payment application A_m is equal to :

$$Risk \quad (A_m) = X\mathbb{P}_{X_i} + Y\mathbb{P}_{Q_i} \tag{1}$$

where

$\mathbb{P}_{Xi}$ denotes the probability that the relevant cryptographic API is implemented incorrectly and $\mathbb{P}_{Qi}$ designates the probability that the cryptographic API concerned is breakable.

In addition, we have :

$$\mathbb{P}_{X_i} = \sum_{i=1}^{6}(\frac{X_i}{n_1}) \quad \text{et} \quad \mathbb{P}_{Q_i} = \sum_{i=1}^{18}(\frac{Q_i}{n_2}) \tag{2}$$

and

$$X = \sum_{i=1}^{6} X_i \quad \text{et} \quad Y = \sum_{i=1}^{18} Q_i \tag{3}$$

where

X_i designates the criteria poorly implemented APIs and Q_i the criteria (C,K,P) of the breakable APIs with : n_1 , n_2 respectively the total number of badly implemented API criteria and the total number of breakable API criteria.

The calculated risk is therefore :

$$Risk \quad (A_m) = X\sum_{i=1}^{6}(\frac{X_i}{n_1}) + Y\sum_{i=1}^{18}(\frac{Q_i}{n_2}) \tag{4}$$

Table 1. Scale of risk according to MAM4MA

Score	Scale of risk
0	None
0.1 - 3.9	Low
4.0 - 6.9	Medium
7.0 - 8.9	High
9.0 - 10.0	Critical

4 Tests and results

After downloading twenty (20) mobile payment applications for Android randomly from deposits (Google Play, Aptoid, Apkpure), we copy them into a folder created for this purpose. Then we use the Enjarify and Bytecode Viewer tools. The first one is used to decompile and the second one to explore decompiled applications looking for misuse criteria.These steps as illustrated in fig 2.

We have in :

1. *Collection of mobile payment APK* :
   At this stage, we download the different mobile payment applications.There are taken randomly from different repositories.

2. *Formatting* :
   At this level we use appropriate tools to make the downloaded application usable (conversion into *.jar* format).

3. *Content Exploration* :
   At this time, we are looking for cases of misuse among the algorithms implemented in the applications.

4. *Risk assessment* :
   After having found bad implementation criteria, we calculate the risk linked to their presence in mobile payment applications.

Fig. 2. steps of test

The mobile payment applications for the study were grouped into three groups. The first group is the one containing mobile payment applications that are linked to a bank account (APCB); the second is the one that does not require a bank account (APNCB) and the last one contains payment solutions (SP). Initial observations point to several misuses of cryptographic algorithms.

Firstly, these are C1 (misuse of encryption algorithms or hash function) as detected in eighteen (18) applications (90%), C2 (use of password-based encryption) found in all twenty (20) applications. Then, we have C5 (use of CBC

block mode encryption combined with PKCS5 as stuffing algorithm), I6 (use of RSA combined with any other type of stuffing algorithm), and C3 (use of ECB, dictionary-based encryption mode).

Also, we note I1 (a re-implementation of the standardized version of AES) and P2 (use of CBC block mode encryption combined with a non-random initialization vector). Finally, we have, K3 (use of static or constant encryption key) and P4 (use of initialization vector embedded in the source code). By grouping these misuses into categories (4), we have the following result in the table. We find that most of the misuses that are made when securing mobile payment applications come from using bad algorithms.

Table 2. Grouping vulnerabilities by category

	APCB	**APNCB**	**SP**	**Total**
Use of unsuitable algorithms	15	22	20	57
Bad implementations	8	9	6	23
Use of wrong keys	4	1	4	9
Use of wrong cryptographic parameters	9	6	5	20

Legend:
SP: Payment solution
APCB: Payment application related to bank accounts
APNCB: Payment application not related to bank accounts

After presenting the vulnerabilities due to the misuse of cryptographic APIs, we associate security risks by level of risk (criticality) and by applications of payment. These differents elements are available in table 3. The mobile payment applications have been grouped by category (APCB, APNCB, SP). We note that the level of risk is higher for applications that interact with bank accounts, then means concerning payment solutions and low for payment applications that do not require a bank account for work.

In the first case, we explain the situation by the fact that the interaction with an account adds several APIs whose role is to ensure the security of the link (applications payment server - bank server). The second case is related to the inclusion of several platforms each adding their own vulnerabilities. The last one is related only to payment application vulnerabilities.

Table 3. A1 - Mobile payment applications and risk level

N	Origin	Risk	Level	Category
1	GP	***0,06***	low	APNCB
2	GP	***0,22***	low	SP
3	GP	***0,39***	low	APCB
4	GP	0,39	low	APNCB
5	GP	0,67	low	SP
6	APP	*0,89*	low	APNCB
7	GP	0,89	low	APNCB
8	APT	1,06	low	APCB
9	GP	1,06	low	APNCB
10	GP	1,06	low	SP
11	APP	1,31	low	SP
12	APT	1,39	low	SP
13	GP	1,56	low	APNCB
14	GP	2,00	low	APNCB
15	APP	2,67	low	APCB
16	APT	2,67	low	APNCB
17	GP	2,72	low	SP
18	APT	***4,39***	midle	SP
19	GP	***5,17***	midle	APCB
20	APT	***8,39***	high	APCB

Legend: **GP**: Google Play **APT**: Aptoïde **APP**:Apkpure

5 Conclusion

In short, it appears that very often developers use cryptographic APIs to ensure security. Unfortunately, these are poorly implemented or not updated, with the opposite effect. We propose a model that takes cases of cryptographic misuse in mobile payment. This is the case in this study, which showed a high proportion of misuse of encryption and hash APIs, which is an important aspect of security services (confidentiality, integrity, availability). Then we propose alternatives to developers to remedy this situation.

Thus, the future perspectives are first to increase the number of misuse criteria to widen the detection field of the proposed model; secondly, to adapt the model to take into account the applications on android except those treated in the framework of this work.

References

1. orange Homepage, https://www.orange.bf/particuliers/1/5/orange-money-2525.html. Last accessed 10 Jun 2020
2. onatel Homepage, http://www.onatel.bf/particulier/telephonie-mobile/produits-et-services/mobicash.aspx. Last accessed 10 Jun 2020
3. safaricom Homepage, https://www.safaricom.co.ke/personal/m-pesa. Last accessed 10 Jun 2020
4. CVSS Homepage, $https : //www.first.org/cvss/v3 - 1/cvss - v31 - user - guide_r1.pdf$. Last accessed 09 Aug 2020
5. CVE Homepage, https://cve.mitre.org/cve. Last accessed 09 Aug 2020
6. CCEVS Homepage, https://www.niap-ccevs.org/. Last accessed 09 Aug 2020
7. OWASP Homepage, https://www.owasp.org. Last accessed 09 Aug 2020
8. OCTAVE Homepage, https://study.com/academy/lesson/operationally-critical-threat-asset-vulnerability-evaluation-octave-definition-overview.html. Last accessed 09 Aug 2020
9. M. Egele, D. Brumley,M. Egele, D. Brumley, Y. Fratantonio, C. Kruegel.: An empirical study of cryptographic misuse in Android applications. In: Proceedings of the ACM, Conference on Computer and Communications Security 2013, pp. 73–83. https://doi.org/10.1145/2508859.2516693
10. A. K. Wickert, M. Reif, M. Eichberg, A. Dodhy, M. Mezini.: A dataset of parametric cryptographic misuses. In: IEEE International Working,Conference on Mining Software Repositories 2019, vol. 2019, pp. 96–100 https://doi.org/10.1109/MSR.2019.00023
11. A. Chatzikonstantinou, C. Ntantogian, G. Karopoulos, C. Xenakis.: Evaluation of cryptography usage in android applications. In:EAI International, Conference on Bio-inspired Information and Communications Technologies (BICT) 2015, https://doi.org/10.4108/eai.3-12-2015.2262471
12. David Lazar, Haogang Chen, Xi Wang, N. Zeldovich.: Why does cryptographic software fail ? A case study and open problems. In: Proceedings of 5th Asia-Pacific Workshop on Systems 2014, pp 1–7, https://doi.org/10.1145/3196494.3196538