Open Archive Toulouse Archive Ouverte

OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible

This is an author's version published in: http://oatao.univ-toulouse.fr/24820

Official URL:

https://doi.org/10.1016/j.future.2018.11.051

Any correspondence concerning this service should be sent to the repository administrator: tech-oatao@listes-diff.inp-toulouse.fr

Preventing the propagation of a new kind of illegitimate apps

Lavoisier Wapet [a], Alain Tchana [a], Giang Son Tran [b], Daniel Hagimont [a,*]

[a] IRIT, University of Toulouse, France
[b] University of Science and Technology of Hanoi, VAST, Viet Nam

HIGHLIGHTS

- Attacks with fake apps for well-known companies that did not publish a mobile app yet.
- Existing solutions (Androguard and FsQuadra) cannot addressed this issue.
- A service which is able to successfully detect and prevent such fake apps.
- Evaluation with more than 5000 apps, thus demonstrating its accuracy.
- Able to daily validate all apps deployed on Google Play or Apple App Store.

ABSTRACT

A significant amount of apps submitted to mobile market places (MMP) are illegitimate, resulting in a negative publicity for these MMPs. To our knowledge, all scanning solutions in this domain only focus on the detection of illegitimate apps which mimic existing ones. However, recent attack analysis reveal the appearance of a new category of victims: enterprises which did not yet publish their app on the MMP. Thereby, an attacker may be one step ahead and publish a malicious app using the graphic identity of a trusted enterprise. Famous enterprises such as Blackberry, Netflix, and Niantic (Pokemon Go) have been subject of such attacks. We designed and implemented a security check system called IMAD (Illegitimate Mobile App Detector) which is able to limit aforementioned attacks. The evaluation results show that IMAD can protect companies from such attacks with an acceptable error rate and at a low cost for MMPs.

1. Introduction

Personal computers changed our world a lot in the last decades. Computer applications (apps) automated many of the complex and challenging tasks previously accomplished by human brain. However, not all computer apps are beneficial. Among them, there is also an important number of malicious apps (the recent WannaCry virus has infected thousands and thousands of computers across the world [1]). For several reasons, the attackers are increasingly attracted by mobile platforms. First, in comparison with a desktop machine, a mobile device has more peripherals (e.g. accelerometer, GPS, proximity sensor, camera, etc.) which generate more interesting data for an attacker. Second, a mobile device has access to a mobile network where overtaxed numbers may be exploited in order to increase the attacker's revenue.

Nowadays, the vast majority of mobile apps are made available to users through digital distribution platforms called mobile market places (MMP) such as Google Play for Android and App Store for Apple. These main MMPs host a tremendous amount of apps. For example, Google Play has more than 3.3 million [2] apps and over 50 billion downloads [3]. Considering this plethora of MMP apps, it becomes fairly difficult for a malicious app to be visible enough to be downloaded. Thereby, they try to associate themselves with well-known public entities. A study from a security company [4] revealed that around 77% of the most downloaded apps have at least one illegitimate version. To cope with them, several app scan solutions have been investigated and integrated into MMPs (e.g. Bouncer at Google). To our knowledge, scanning solutions for illegitimate apps only focus on the detection of apps which mimic **existing ones**. However, recent attack analysis reveal the arrival of a new victim category. Attackers develop and publish mobile apps for well-known companies **which did not publish a mobile app yet.** In 2013, the Blackberry messenger has been a victim of such an attack where an illegitimate app has been published on Google Play and downloaded about 100k times before its removal. More recently (September 14, 2016), Pokemon Go has also been attacked in the same way; see the following post [5]: "*A few days ago we reported to Google the existence of a new malicious app in the Google Play Store. The Trojan presented itself as the "Guide for Pokemon Go". According to the Google Play Store it has been downloaded more than 500,000 times... Kaspersky Lab products detect the Trojan as*

---

* Corresponding author.
E-mail address: hagimont@enseeiht.fr (D. Hagimont).

HEUR:Trojan.AndroidOS.Ztorg.ad. At least one other version of this particular app was available through Google Play in July 2016.". The French Telecommunication company Orange has announced the development of a mobile money service called Orange Bank [6] which will be available throughout a mobile app this summer. We have successfully experimented the publication of an illegitimate version of that app on a popular MMP and several downloads have been observed.[1]

This paper presents IMAD (Illegitimate Mobile App Detector), a solution for detecting this new category of illegitimate apps at submission time (when the developer uploads the app in the MMP). To our knowledge, this is the first research work which investigates this issue. The main principle is to identify, from visible characteristics of the app (e.g., name or logo), the trusted entity (e.g., a company) associated with these characteristics. This trusted entity is either the submitter or the one the submitter wants to mimic. This identified entity is then contacted by email to validate the app submission. The implementation of IMAD raises several challenges. The most important among them is the following: how to identify the trusted entity associated with an app regarding the number of worldwide trusted entities? IMAD answers this question by relying on the biggest database in the world which is Google (its search engine). Our basic idea is to combine several standard text and image similarity checking in order to find from the internet the legitimate and trusted entity behind each submitted app (its visible characteristics). Although this idea appears simple to label, its implementation is not easy. The evaluation of IMAD with more than 5000 apps (from AndroZoo [7] and Contagio [8], among other data sets) demonstrates the effectiveness of our approach, with an acceptable margin of error: almost nil on legitimate apps and less than 20% on illegitimate apps. Overall, this paper makes the following contributions:

(1) We highlighted a new security problem which affects all MMPs: the apps presented under the image of a well-known public entity which does not have a mobile app yet.
(2) We presented an algorithm which is able to successfully detect and prevent the above problematic situation. We provide IMAD, a prototype which is easy to exploit and to integrate with existing MMPs.
(3) We evaluated IMAD with more than 5000 apps, covering all enterprise categories (geographical location, activity, etc.). The evaluation results show that IMAD can protect both big, small and medium-sized companies from such attacks. In addition, our system is able to validate all apps deployed within a day on Google Play or Apple App Store for a minimal cost (about $1,755).
(4) We compared IMAD with existing solutions (namely Androguard [9] and FsQuadra [10]) which confirmed that the studied issue cannot be addressed using current approaches. We showed that IMAD is also able to detect illegitimate situations handled by existing solutions.

The rest of the article is organized as follows. A review of the related work is presented in Section 2. Section 3 defines the concepts used in this work. It also presents the motivations. Section 4 presents our contributions while Section 5 presents evaluation results. Finally, we present our conclusion in Section 6.

2. Related work

Many studies contributed to the issue of detecting illegitimate apps. The proposed approaches can be classified according to three main criteria:

---

[1] This experiment was validated by the ethics commission from our laboratory.

Where: the **place** where detection takes place. Detection can be initiated in the MMP where the app is deployed, or on the mobile terminal where the app is installed.

When: the **time** when detection is performed. Detection can be performed statically: an analysis of the app's installation files. It can also be performed dynamically when the app is launched (either in the MMP or on the mobile terminal).

How: the employed detection **method**. We classify these methods into two groups: those which attempt to detect **internal** abnormal or suspicious characteristics within the app (e.g., abnormal communications), and those which attempt to detect fake apps through **similarities** with legitimate apps (e.g., based on images or logos).

These criteria logically lead (according to the place/time/ method) to the following classes of solutions:

MMP/static/internal. Solutions in this class rely on the analysis of app installation files when apps are published in the MMP. An example is described in [11] where they analyse the control flow in the app code in order to detect malicious behaviours.

MMP/static/similarities. Solutions in this class aim at detecting similarities between suspicious apps (fake apps) and legitimate apps already published in the MMP. Such similarities may be detected from document files (text, images) packed with the app [12] or from its code [13–15].

MMP/dynamic/internal. In this class, solutions rely on a dynamic analysis, i.e., they execute the app before effectively publishing it in the MMP. This execution is a means to observe the internal behaviour of the app. For instance in [16], they observe runtime communications in order to detect connections with malicious sites.

MMP/dynamic/similarities. In this class, solutions are looking for similarities with existing apps in the MMP to detect fake apps, but dynamically before publishing. In [17], they analyse at runtime apps' GUI (inside the MMP) in order to classify apps and detect similarities.

Mobile/static/internal. Solutions in this class rely on static detections when the app is installed on the device, as does Android bouncer [18] with electronic signatures.

Mobile/dynamic/internal. Dynamic solutions are also proposed on the device. For instance, [19] verifies that the name of the app (captured from its graphical user interface at runtime) is consistent with the communication endpoints (URLs) used by the app. Also, several solutions [11,20,21] introduce indicators (for instance an image) chosen by users when a (known) legitimate app is installed. If an app imitates a legitimate app without presenting the indicator (in its GUI), the user knows the app is illegitimate.

All these contributions aim at detecting illegitimate apps in an MMP or on a mobile terminal. The detection may be performed statically by analysing the installation files of the app or dynamically by observing the app's behaviour. As presented in Table 1 The detection either identifies a malicious behaviour within the app or identifies a similarity with a legitimate app. Detecting malicious behaviours within apps is limited because it is difficult to cover all attacks (and avoid false negatives). Similarity detection appears to be more promising.

Our solution falls into this latter category. However, it does not rely on the detection of pre-identified characteristics (e.g., from the GUI) from already published apps, which would limit its coverage. It detects all attacks, including those targeting apps which do not yet have a mobile version published in an MMP.

Table 1
Drawbacks and advantages of related work solutions compared to IMAD.

How	Similarities		Internal		IMAD
Where	MMP	Mobile	MMP	Mobile	
Independent of store data	✓	✓	✗	✗	✓
Independent of store applications	✗	✗	✓	✓	✓
Independent of the location	✗	✗	✗	✗	✓
Independent of the attack mecanism	✓	✓	✗	✗	✓

3. Definitions and motivations

3.1. Definitions

Trusted entity. We define a trusted entity as an enterprise or an institution which is recognized by a government authority (generally through a unique identification number). In this paper, we are interested in apps which belong to trusted entities, not to private holders. Let us call $\mathcal{T}$ the set of existing trusted entities.

Graphic identity (noted GI). The GI of a trusted entity T (respectively an app A) is the set of visual elements which refer to T (respectively A) without confusion. Most of the time, the GI of an app is included within the GI of the trusted entity which possesses the app. The GI of a trusted entity may be its name, logo or any image which refers to one of its services.

Trusted developer and the attacker. Let us consider an app A, implemented by a developer D. The latter is said to be a trusted developer if the trusted entity T which is behind the GI of A recognizes D, otherwise D is considered to be an attacker. Knowing that in some cases (which are extremely rare because entities generally try to define GI so that they are unique) several entities may have similar GI, we consider that the trusted entity behind a GI is the most popular one.

Legitimate and illegitimate app (respectively noted L and I). An app is said to be legitimate if it has been published by a trusted developer, otherwise the app is illegitimate. Notice that the illegitimacy of an app is independent from the developer intent. It means that an illegitimate app is not necessarily dangerous from the user point of view. However, it is from the point of view of the trusted entity because it is steering its users. For instance the Orange Cache Cleaner app (a small tool used for clearing application cached files) could be considered illegitimate because its GI refers to the famous enterprise Orange (the French telecommunication company), especially its service Orange Cash. From this definition, one can easily understand why legitimate/illegitimate app detection systems mainly rely on GI analysis.

Malware and Safe app (respectively noted M and S). An app is said to be a malware if it does actions without the initial approbation of the user, otherwise the app is said to be safe. Subsequently, malware detection systems mainly study the *behaviour* of the app.

3.2. Research scope

According to the above definitions, apps can be classified into four categories[2]: $L \cap S$, $L \cap M$, $I \cap S$, and $I \cap M$. Except the former category, all the others consist of what we call bad apps (towards the user or a trusted entity). MMP operators try to avoid the publication of bad apps on their platforms. Therefore, before being published, each submitted app is subject to several security checks that can be synthesized in two steps (see Fig. 1): GI analysis (for detecting illegitimate apps) and behaviour checking (for

detecting malware). Apps which fail one of these controls are kept within a dedicated storage for further studies (e.g. for improving the detection systems). An app is published only if it satisfies all the security checks. In this paper, we only focus on the detection of illegitimate apps, which are the basis of phishing attacks [16]. Thus, malware detection is out of scope for this paper. The next section summarizes the current state of the research in this domain in order to highlight our specific contribution. For illustration, we consider Android apps, although our contribution can be applied to other app types.

3.3. Problematic

Let us note $\mathcal{A}$ the set of downloadable/published apps in the MMP. Let us note A_{pub} an app under the submission process. Existing illegitimate app detection solutions can be organized into three classes:

(1) source code analysis [11,14]: they check if there is an app in $\mathcal{A}$ whose implementation structure (especially its graphical user interface) is similar to the one of A_{pub}.
(2) image analysis [12,15]: they check if there is an app in $\mathcal{A}$ which uses the similar images as A_{pub}.
(3) and app name analysis [20]: they check if there is an app in $\mathcal{A}$ whose name is similar to A_{pub}.

As we can see, all these solutions only focus on detecting if A_{pub} is similar to **an existing app**. More formally, they answer the following question: *(Q1) $\exists A \in \mathcal{A}$ (within the MMP), such that A_{pub}'s GI is close to A's GI?*

We claim that answering (Q1) does not allow to cover all illegitimate apps at submission time. Consider the situation where the attacker implements A_{pub} as a service of a trusted entity **which has not yet published** a mobile version of its service. The fraudulent nature of A_{pub} will not be detected by current solutions. Fig. 2.a presents the list of apps suggested by Google play when the user is looking for an IEEE app. This suggestion list could have been the one presented in Fig. 2.b, which includes two illegitimate apps: EasyChair (faking the legitimate conference management system EasyChair [22]) and ACM EuroSys (faking the EuroSys conference management system). This situation may occur in current MMPs because the legitimate organization behind EasyChair for example has not yet published a mobile app version of the system.[3] For example, the attacker could obtain the *username* and the *password* of the conference reviewers. Therefore, their reviews could be the subject to Man-in-the-Middle attacks. Another illegitimate situation we experimented concerns the french telecommunication company Orange. The latter has announced the arrival of a mobile money app called Orange Bank [6] this year. Our team has developed and successfully published on a popular MMP[4] an app which purports to be Orange Bank. This situation could have been very problematic for the legitimate company and its clients in the case of a real attacker. Notice that we have unpublished the app after one month so as to avoid exposure to legal proceedings.

Considering the large success of smartphones combined with the trend of converting computer applications to mobile apps, this problem is crucial. Therefore, illegitimate app detection systems should not limit their checks to the GIs within the MMP. More formally, instead of answering (Q1) as current researchers do, we answer the following question: *(Q2) $\exists T \in \mathcal{T}$ (worldwide), such that A_{pub}'s GI is close to T's GI?* This is a very difficult problem. This paper presents (for the first time) a solution to this issue.

---

[2] For instance, $L \cap S$ means the intersection of Legitimate and Safe apps.

[3] Notice that it totally makes sense to have the mobile version of these apps. It would be useful for conference organizers.

[4] The name of the MMP is not revealed in order to avoid negative publicity.
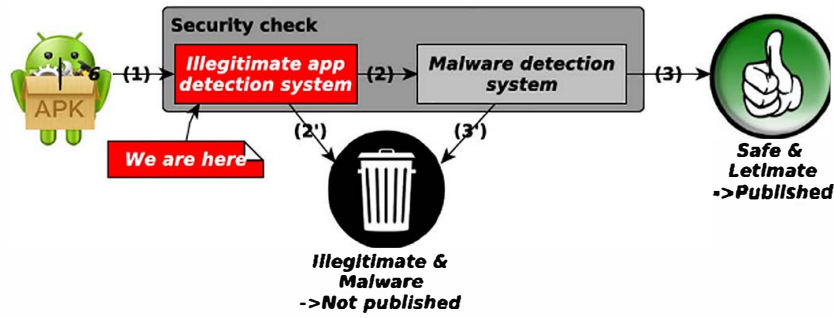
Fig. 1. Synthetic app submission workflow, from the security check point for view.
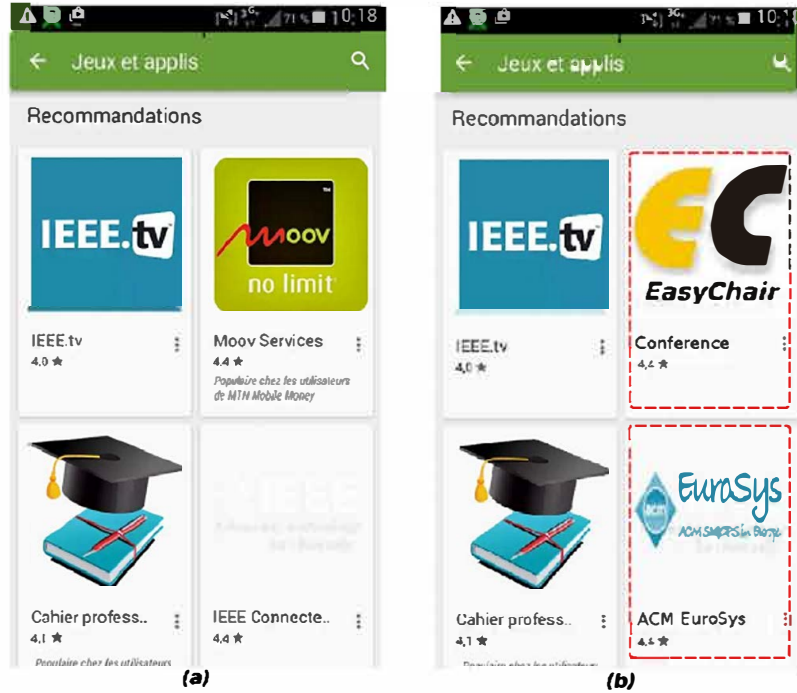

Fig. 2. Example of attack.

4. IMAD: Illegitimate Mobile App Detector

4.1. Overall system design

This paper presents IMAD, a solution for detecting illegitimate apps at publication time. Its works as follows, summarized in Fig. 3. Once the app is uploaded, IMAD builds its GI, noted GI_A_{pub}. Then it performs a set of web searches (on the internet), analysed with standard machine learning techniques using each element in GI_A_{pub} in order to find the trusted entity behind GI_A_{pub}. This is the core of our solution. We assume that any trusted entity can be found on the web.[5] If the result of the previous step reveals the presence of at least one trusted entity (noted T), a validation email is sent to it and a countdown is armed. The app is considered to be illegitimate if IMAD does not receive a validation email before the end of the countdown. Notice that if the developer is legitimate, thus the trusted entity will waiting for the validation email sent by IMAD.

IMAD can be deployed in two manners: directly within a specific MMP or deployed as an independent service (IMAD as a Service or IMADaaS). We consider this latter case because it is the most generic one. Therefore, once an MMP is registered as an IMADaaS customer, it can automatically forward all received apk (Android Package Kit − we mainly experimented with Android apps) to IMADaaS for checking. Upon receiving the checking result, the MMP can apply its internal checking system (see Fig. 1).

The implementation of IMAD raises several challenges. The most important among them are: (1) Trusted entity determination: how to cover all trusted entities which exist throughout the world? How to obtain their GI, knowing that there is no database which includes them? (2) GI comparison: how to identify, with as less errors as possible, the proximity between GIs? How to minimize both the false positive and false negative rates? (3) Scalability: the time, the amount of resources as well as the cost required for exploiting IMAD should be acceptable. The next sections details each IMAD's component while tackling the above challenges. To facilitate reading, difficult concepts are introduced (when needed) and followed by illustrations. The latter are based on the illegitimate EasyChair app presented in the previous section.

4.2. Graphic identity (GI) construction

IMAD considers the following elements as part of GI_A_{pub}: the name of the app (noted *appName*) and its logo. These elements are

---

[5] One may ask why not just getting the contact of T from the submitter in order to accelerate searches. This would not be secure because our system does not trust the developer, who could be an attacker.
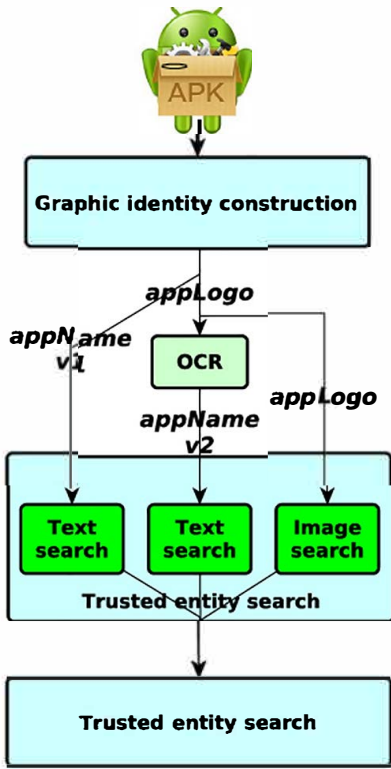
Fig. 3. IMAD general functioning.

chosen because they are those which mainly influence the user's decision in the process of associating an app with a trusted entity. Concerning the name, we consider two versions: the one given by the developer (called *appName v1* in Fig. 3), and the other overlaid onto the logo (called *appName v2* in Fig. 3). This second version is important because sometimes the attacker can provide a bizarre name, knowing that the relevant one is well visible on the logo. The extraction of all these elements is straightforward. We use *apktool* [23] to extract both *appName v1* and the logo from the apk. Then we use tesseract-ocr [24], an optical character reader system (OCR in Fig. 3), to extract *appName v2*. Applied to our illustrative example, *appName v1* could be "BXdFcGKfp1" while *appName v2* is "EC EasyChair" (refer to the logo in Fig. 2.b).

4.3. Trusted entity search

After the construction of the submitted app's GI (noted *GI_A$_{pub}$*), IMAD has to find the trusted entity (if exists) which is behind the name or the logo of the app. Our basic idea is to rely on the web (especially the Google custom search engine) in order to consider all trusted entities. In fact, we assume that attackers only target trusted entities which are known by a significant number of persons, and we are betting that such entities are indexed by Google. Each element of *GI_A$_{pub}$* is used as a search criteria, all searches being performed in parallel. Therefore, we distinguish two search types:

- text search: performed on *appName v1* and *appName v2*, see Sections 4.4–4.9.
- image search: performed on the logo, see Section 4.10. This step exploits almost the same algorithms as the text search.

4.4. Text search (based on appName)

The main difficulty is to filter from the search results all pages which are not directly related to the official website of the trusted

entity behind *appName*. To this end, we use a five-step algorithm, summarized in Fig. 4:

- (1) collection of web pages which relate to *appName*;
- (2) organization of web pages into clusters according to the topic they deal with;
- (3) elimination of clusters whose topic does not relate to a trusted entity, the rest are merged;
- (4) elimination of pages which are not directly related to *appName* (non official pages, youtube pages, press articles, etc.);
- (5) extraction of the contact (email) of the trusted entity.

4.5. Web page collection

We rely on the Google Custom Search framework [25] to perform this task. Google's APIs can be used for programming custom Google searches. The result of a search is a list of items which represent web pages. Each returned item is composed (among others) of a title, a brief description of the web page, and a link to its HTML content. In the case of IMAD, we have experimentally seen that the first 20 items are sufficient (see the evaluation section). From each item, IMAD builds what we call a *document* (noted *doc*) by concatenating the name, the description and the HTML content. Each document then goes through few changes (such as conversion to lower-case, elimination of HTML tags and stop words, etc.) in order to facilitate the next steps. The obtained set of documents is called the *corpus*. Notice that documents which contain very few information (such as 404 pages) are removed from the corpus. Fig. 5 presents the corpus built from our illustrative example.

4.6. Clustering

Intuitively, clustering consists in gathering documents from the *corpus* which nearly have the same group of words (the application of this step to our illustrative example is presented in Fig. 6). To this end, we used the k-means clustering algorithm. K-means works on vectors while we deal with a corpus. Several studies have investigated the issue of corpus vectorization. IMAD uses Vector Space Model [26], a widely used solution.

A dictionary. This solution is based on a dictionary (noted $\mathcal{D} = \{w_1, \ldots, w_n\}$) which includes the words from the corpus. Naively, we could use all the words that appear at least once in the whole corpus. This would result in a very large dictionary which could impact the execution of the k-means algorithm. In IMAD, we only consider words that appear at least once in all document titles and descriptions. The HTML content, which is the largest part of a document is ignored. This solution is acceptable because the relevant words of a web page are either in its title or its description.

Vectorization. The basic idea is to transform each document doc_j ($1 \le j \le m$, m is the number of documents in the corpus) into a n-sized vector (noted vec_doc_j), n be the size of the dictionary. The ith coefficient of vec_doc_j is also called the coefficient of w_i in doc_j. It is noted $Coef_{i,j}$ and it evaluates the importance of the word for characterizing the document.

$Coef_{i,j}$ is computed using the TF–IDF (Term Frequency–Inverse Document Frequency) standard, as follows:

$$Coef_{i,j} = tf_{i,j} \times idf_i \tag{1}$$

where $tf_{i,j}$ is the occurrence frequency of w_i in doc_j, and $idf_i = log(\frac{m}{m_i})$, with m_i be the number of documents containing at least once w_i. Roughly, the higher the occurrence frequency of the word in the document, the higher its coefficient. However, the word is
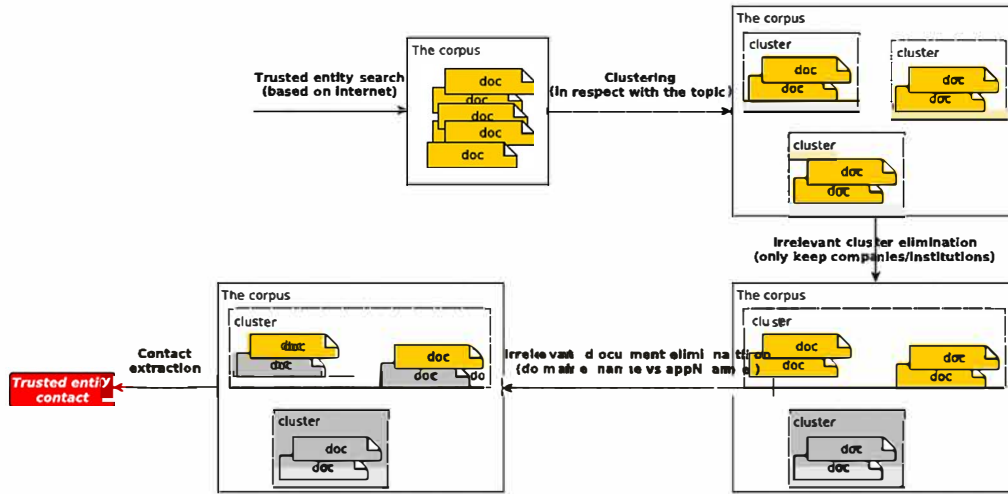
Fig. 4. The main steps of our solution.

Corpus built from *appName* v1="BXdFcGKfp1" search results

No document !

Corpus built from *appName* v2="EC EasyChair" search results

```
doc1  http://easychair.org/
doc2  https://en.wikipedia.org/wiki/EasyChair
doc3  http://voronkov.com/easychair.cgi
doc4  http://www.magisdesign.com/fr/elenco_prodotti/easy  family/easy  chair/
doc5  http://www.ikea.com/gb/en/products/sofas  armchairs/armchairs/nolmyra  easy ↘
      chair  birch  veneer  grey  art 10233532/
doc6  https://headgum.com/the  easy  chair
```

Fig. 5. Illustration of the Web page collection step.

$Cluster_1$

```
doc1  http://easychair.org/
doc2  https://en.wikipedia.org/wiki/EasyChair
doc3  http://voronkov.com/easychair.cgi
```

$Cluster_2$

```
doc4  http://www.magisdesign.com/fr/elenco_prodotti/easy  family/easy  chair/
doc5  http://www.ikea.com/gb/en/products/sofas  armchairs/armchairs/nolmyra  easy ↘
      chair  birch  veneer  grey  art 10233532/
```

$Cluster_3$

```
doc6  https://headgum.com/the  easy  chair
```

Fig. 6. Illustration of the clustering step: we present three clusters built from the corpus presented in Fig. 5 bottom.

penalized if it appears in most of the documents, as it would not be relevant to characterize a specific document.

K setting. K-means requires the number of clusters (k) as an input parameter. [27] proposes a way to calculate k in the context of corpus clustering. It computes k as follows:

$$k = \lceil \frac{m \times n}{t} \rceil \qquad (2)$$

Where t is the number of non-zero coefficients in the TF–IDF matrix (the stack of all vec_doc_j).

K-means initialization. K-means also requires the initial position of the centre of the clusters as an input parameter. A wrong initialization could lead to a wrong result. We use k-means++ [28] to handle this issue. K-means++ is able to choose a satisfying value (not necessarily the optimal one) for the initial centre.

4.7. Irrelevant cluster elimination

The goal of this step is to discard clusters whose topic does not refer to a trusted entity (e.g. the third cluster in our illustrative example). To this end, we implement the following idea. Each cluster's topic is determined and used to query an accessible database of trusted entities, allowing a score to be assigned to the topic, indicating to what extend it is related to a trusted entity. Then, only clusters whose score exceeds a threshold (determined experimentally) are kept. The challenge here is threefold: cluster's topic

determination, finding an accessible database of trusted entities, and determination of an accurate scoring function (noted S).

Topic determination. Given a cluster C, we consider its topic being the list of words which summarizes the main idea developed by all its documents. We call these words *important words* (noted C_{IW}). C_{IW} is computed using the Vector Space Modeling of the cluster. This time, however, the TF standard (Term Frequency) is used instead of TF–IDF as previously. The TF standard is suitable here because we are looking for important words.

Accessible database of trusted entities. The subtle way we adopt is to rely on the two biggest semantic databases that exist: Wikipedia-dbPedia [29] and the World Intellectual Property Organization (WIPO) [30]. We implement a set of tools for accessing these databases. In this paper we focus on Wikipedia-dbPedia for illustration. Let us say a few words about wikipedia-dbPedia, necessary for understanding our solution. Wikipedia is a participatory-controlled semantic database composed of web pages, called *wiki concepts*. DbPedia is a structured version of wikipedia in which each wiki concept is characterized using several criteria. Among these criteria, the category allows to know if a wiki concept refers to a trusted entity or not. For instance, the category values "Company", "Organization" or "Business" refer to a trusted entity. A wiki concept is also associated to an abstract which quickly describes it.

Cluster scoring. Given a cluster C, its important words C_{IW} are used to compute its score, as follows:

$$S(C) = \max_{w_i \in C_{IW}} (S_1(w_i)) \tag{3}$$

where $S_1(w_i)$ is the score of w_i, computed in this way:

$$S_1(w_i) = \max_{wc_k \in WC_i} (S_2(wc_k)) \tag{4}$$

where WC_i is the set of wiki concepts related to w_i. WC_i is obtained by enforcing a Google search only on wikipedia pages ("w_i site:en.wikipedia.org"). The first 10 resulting items are used for extracting WC_i's elements.

Concerning $S_2(wc_k)$, it depends on both wc_k itself and the studied cluster C because two constraints should be respected: (i) If wc_k refers to a trusted entity (whatever it is), $S_2(wc_k)$ has to be high; (ii) If wc_k is unrelated to C, $S_2(wc_k)$ has to be low. This second constraint allows to minimize the false positive rate. For instance, if the topic of a cluster deals with "Orange" (the fruit), the wiki concept "Orange SA" (which is an enterprise) must have a low score. In summary, we compute $S_2(wc_k)$ as follows:

$$S_2(wc_k) = \delta(wc_k) + \sigma(wc_k) \tag{5}$$

with $\delta(wc_k)$ be the trusted entity closeness coefficient and $\sigma(wc_k)$ the cluster closeness coefficient. To compute $\delta(wc_k)$, we build a textbook (called the Rescued Category List, RCL) consisting of dbPedia category words which characterize a trusted entity ("Company", "Organization", "Business", etc.) so that: $\delta(wc_k) = 1$ if the category of wc_k appears in the textbook; $\delta(wc_k) = 0$ otherwise.

Note that in our textbook, only categories referring to companies were considered. We focused on companies since they are the main targets of attackers. However the textbook can be easily enriched with the lexical fields of many other types of organization such as universities and governments agencies.

About $\sigma(wc_k)$, we use the cosine similarity [31], which allows to estimate the distance between two texts. In our case, these texts are: the abstract of the wiki concept (noted abs_wc_k), compared with each document in the cluster. Therefore,

$$\sigma(wc_k) = \frac{\sum_{i=1}^{m_C} cosine(abs\_wc_k, doc_i)}{m_C}.$$

The application of this step to the illustrative example is as follows. Fig. 7 presents the important words of the illustrative clusters. We can see that only the first two clusters are related to trusted entities: EasyChair (the conference management system) and Ikea (furnishing trader).

4.8. Irrelevant document elimination

At this stage, all remaining clusters are merged in order to form a unique cluster. The purpose of this step is to focus only on the documents which directly belong to the trusted entity behind *appName*. To this end, we discard all documents whose domain name is not phonetically close to at least one word of *appName*. For instance, the document *doc2* in the illustrative example does not belong to EasyChair, thus it should be discarded. We choose the domain name because most of the time the company name is used as the basis for building its domain name. We combine two methods to achieve this step: a metaphone algorithm [32] (it assigns to a given string a key indicating its pronunciation) and a string distance algorithm [33] (it compares two strings). Our idea is to first compute the metaphone key of the domain name (noted MK_{dm}) and each *appName*'s word w_i (noted MK_{w_i}). Thereby, we compute the string distance between MK_{dm} and each MK_{w_i} (noted sd_i). Further, it is normalized so that $sd_i = 0$ means the two keys match perfectly. Therefore, if $\exists$ i so that sd_i is smaller than a configured threshold (we experimentally found that 0.3 is a good value, see the evaluation section), the domain name's document is kept. Fig. 8 presents the retained documents in the case of the illustrative example.

If the previous step provides no trusted entity, the discarded documents are given a second chance. Indeed, it is possible that an enterprise sells several products whose names are phonetically different from the name of the enterprise (thus its domain name). For instance, the URL of the official website of the famous video game *"Diablo 3"* is "http://eu.battle.net/d3". These cases are rare but exists. They include a category of enterprises that we call *catalogue enterprises* (they offer a catalogue of products). The purpose of the second chance step is to recover them. To this end, we exploit again dbPedia's wiki concept categories (see Section 4.7) as follows. We build a textbook (called the Rescued Category List, RCL) consisting of dbPedia categories ("DRM_for_Windows", "DRM_for_OS_X", etc.) which characterize a catalogue entity. Therefore, all domain names whose wiki concept category is part of the RCL are kept. The evaluation results show that this strategy is fairly effective.

4.9. Trusted entity's name and contact extraction

At this stage, the remaining documents (if exist) belong to the trusted entity which is behind *appName*. The objective of this step is to determine the contacts of this entity. To this end, we exploit a technique similar to the one described by Google [34]. In fact, websites are usually very clear about who created the content. There are many reasons for this: copyrighted material protection, businesses want users to know who they are, etc. Therefore, most websites have a contact page ("contact us", "about us" or just "about"), copyright information, or include HTML metadata which provide contact information. In nearly 100% of the cases, the company email address is successfully obtained (see the evaluation section). Then, a validation email is sent to the trusted entity and a counter-down is armed. If no response email is received, we suppose the app is illegitimate. The response email should imperatively respect a given format so that it can be parsed by our framework. Notice that if the app is legitimate, the trusted entity will be waiting for the email and will provide a response. For big companies which are subject to several faking, thus will receive a lot of emails, one may think that they will be lost in a

$Cluster_1$'s important words

```
easychair , conference , subscribe , sign , management
```

$Cluster_2$'s important words

```
ikea , nolmyra , chair , birch , veneer
```

$Cluster_3$'s important words

```
chair , easy , media , cardiff , supply
```

Fig. 7. The important words of the clusters presented in Fig. 6.

```
doc1  http :// easychair . org /
```

Fig. 8. The remaining documents after eliminating those which are not related to EasyChair.

myriad of emails. This issue can be easily handled with an email filter. We provide IMAD with such a component which can be easily integrated with popular mail readers.

To circumvent such a validation scheme, an attacker would have to create a fake web site and to exploit search engine optimization to augment the visibility of the web site, so that it becomes more visible than to web site of the company it attacks. Such an attack would be so visible that it would be easily observed and detected by the attacked company (its image is being stolen) which could take counter-measures. Generally, attackers prefer not to behave this way and to remain hidden.

4.10. Image search (based on the logo)

IMAD also leverages the logo of the app (noted *appLogo*) for determining its trusted entity. To this end, we rely on two observations. (1) A logo is strongly linked with a unique app or trusted entity. It is precisely its main purpose. For instance, 📘 uniquely refers to Facebook, the famous social network. (2) The logo of an app is strongly linked with the trusted entity which possesses it. For instance, 📘 is the logo of both the Facebook corporation and its social network app. We again use Google Custom Search Engine, more precisely its reverse image search system, for achieving this task. By performing a Google search with *appLogo*, we are almost sure to find the trusted entity among the first items. An item here can be of two types: images which are similar to the logo and websites which include the logo. We only consider the second type. Now, the main question is: how to select from the resulting websites the real owner of *appLogo*? To answer this question we implement a six-step algorithm.

Step 1: Construct the corpus, in the same way as presented in Section 4.5.

Step 2: Determine the topic of each document doc_i of the corpus, as presented in Section 4.6 (Topic determination).

Step 3: Eliminate all documents whose topic is not referring to a trusted entity, as presented in Section 4.7 (Cluster scoring).

Step 4: Let us note E_i the owner of doc_i. Search "the E_i logo" on the web. According to the Google ranking system, it is very likely that the logo of E_i is among the first x returned images. We empirically determined that $x = 30$ is fairly sufficient.

Step 5: Compare *appLogo* with each obtained image, using [35]. The latter takes into account several image modifications (rotation, scaling, etc.). The algorithm assigns a rank r_i to every matched image *appLogo*.

Step 6: Choose E_i, so that r_i is the smallest rank. Therefore, E_i is the trusted entity which owns *appLogo*. Its document doc_i is used for extracting its contact, as presented in Section 4.9.

5. Evaluations

This section presents the evaluation results. We evaluated IMAD from the following perspectives:

- the accuracy: is IMAD able to accurately detect the legitimate entity behind an app's GI?
- the scalability: how much resources IMAD consumes?
- the cost: how much money is needed to exploit IMAD?

5.1. Experimental environment

The experiments have been realized on two commodity machines, each composed of 4 CPUs (Intel Core i5-3337U, 1.80 GHz), 4 GB memory, an Intel Corporation 3rd Gen Core processor Graphics Controller, and a Gigabit Ethernet card RTL.8111/8168/8411 PCI Express. One machine hosts IMAD while the other machine runs a mail server and our mail filter, emulating what should happen in an enterprise. We built a data set playing the role of apps which are in the publication process. This data set is composed of 5000 apps, organized as follows:

- D_1: includes safe apps gathered from Androzoo [7] and Contagio [8], two popular data sets. These apps are collected from several sources, including Google Play.
- D_2: includes illegitimate apps from Androzoo and Contagio. The latter analyse all apps they collected using different AntiVirus products. Every app in D_2 has its safe version in D_1.
- D_3: includes apps we developed for the purpose of this paper (e.g. Orange Bank) in order to cover all company types (see below).

These apps have been selected so that all company types are represented, according to the following criteria:

- the size: big enterprises (BE), and small and medium-sized enterprises (SME).
- the location: developed countries (DC), and emerging countries (EC).
- the activity: catalogue enterprises which are generally gaming companies (GC), and non catalogue enterprises (NGC).
- the name: polysemous (PN) and non polysemous (NPN).

For instance, "Bank of America" is a big enterprise (BE) located in a developed country (DC); it is not a gaming company (NGC) and its name contains polysemous words (NPN). It is said to be of type BE_DC_NGC_PN. Types are equally represented in the data set.

5.2. Accuracy

The accuracy of our system depends on the accuracy of each step it is composed of, namely: (S_1) GI construction, (S_2) trusted

entity search, and (S_3) contact extraction. (S_2), as well as (S_3), has three versions in respect with the three elements which compose the GI (*appName v1*, *appName v2*, and *appLogo*). Therefore, (S_{ij}) corresponds to the *j*th version of (S_i). The evaluation protocol we put in place is as follows. We manually inspected each app in order to report the expected output of each step. Thus, the actual outputs obtained during the execution of IMAD are compared with the expected values. The accuracy of (S_i) is only accounted if (S_{i-1}) was accurate. We also evaluated the accuracy of the entire system. The latter is accurate when the output of at least one path (among $S_1 \rightarrow S_{21} \rightarrow S_{31}$, $S_1 \rightarrow S_{22} \rightarrow S_{32}$, and $S_1 \rightarrow S_{23} \rightarrow S_{33}$) corresponds to what we manually found.[6]

The evaluation includes two experiment types, differing from each other by the considered data set:

- The first experiment type allows to evaluate how IMAD behaves facing safe apps (GI elements have been built without having a malicious idea in mind). We relied on D_1 and D_3.
- The second experiment type allows to evaluate how IMAD behaves facing illegitimate apps. We relied on D_1.

The first experiment type. Fig. 9 presents the evaluation results, interpreted as follows. (1) The 100% accuracy rate observed in S_1 and S_3 validates the methodologies we use for extracting GI elements from the apk and contacts from web pages. (2) S_2 is the most critical step. (3) Both S_{21} (based on *appName v1*) and S_{23} (based on *appLogo*) succeed to identify big companies. This is explained by the fact that such companies are well indexed by Google, both their name and logo are popular. (4) The accuracy of S_{23} extends to other company types. This is explained by the fact that the logo is generally built in such a way as to return to a unique company. (5) S_{21} sometimes fails to identify companies which belong to the SME category because they are not so famous as big companies. However, we observed that EC-SME are well identified in comparison with DC-SME. Indeed, it is more difficult to identify a small company in the crowded market of a developed country. This is not the case for an EC-SME which has fewer competitors, thus a better Google index ranking. (6) We also observed that the accuracy of S_{21} is low on apps which belong to countries using a non-Latin alphabet (e.g. Arabic, Mandarin). This situation does not concern big companies of these countries because they most of the time provide an English version of their web site. To validate this observation, we integrate to IMAD a Mandarin string distance algorithm [36]. Then we repeated the previous experiments. We observed the improvement of S_{21} which minimal accuracy rate jumps from about 75% to 93%. The integration of other alphabets to IMAD would nullify the remaining error rate. This is subject of future work. (7) The accuracy of S_{22} (based on *appName v2*) is low because the text extracted from the logo is sometimes completely different from the company name. (8) Considering the fact that at least one IMAD's path is always accurate (the *appLogo* path), IMAD is accurate too.

The second experiment type. We also evaluated IMAD with illegitimate apps. Fig. 10 presents the evaluation results. We only focus on the second step. The following observations can be made from these results. (1) S_{21} leads to the lowest accuracy rate, near zero. This is explained by the fact that attackers generally use bizarre names [37], knowing that the right name is overlaid onto the logo. (2) This is why S_{22} and S_{23} provide better accuracy rates (more than 80%). (3) Therefore, the average accuracy rate of the entire system is about 80%, which is quite high. This is not so high as with safe apps because some attackers use sophisticated

---

[6] We send a validation email to the 3 identified trusted entities and consider the app submission is validated if one of these 3 contacts responds. Therefore, we consider that the approach is accurate if one of the 3 used methods is successful.

mechanisms for building the GI. For instance, it can be built at runtime by downloading a remote bitmap, making offline solutions inefficient. To handle such cases, we improved IMAD as follows.

Optimization: screen-shots analysis. We improved the GI construction step by extracting salient objects (those on which the human visual system pays more attention) from the first screen of the app. To do so, IMAD runs the app in an emulator and captures the first screen-shot, which has been proven to be enough for identifying an app [38,39]. Having the screen-shot, we use the algorithm proposed by [35] to extract salient objects. Since not all salient objects are important (e.g. object which represents a text field), we discard all objects representing components which are commonly used in forms (text field, list, checkbox, etc.). The remaining objects are used for performing web searches. The evaluation results of this optimization reveal a negligible improvement, less than 3%. This is due to the low accuracy of the salient object extraction algorithm.[7] Notice that salient object extraction is a recent and hot topic in the multimedia domain. The evaluation results presented in the next sections rely on IMAD without this optimization.

5.2.1. Comparison with existing solutions

We compared IMAD with two reference illegitimate app detection systems which cover all the existing approaches (Section 3.3): source code analysis, image analysis, and app name analysis. The former is provided by Androguard [9] while the two others are provided by FsQuadra [10]. Recall that the basic idea behind existing solutions consists in comparing the submitted app with those which already exist within the MMP. To compare IMAD with these solutions, we adopted the following protocol. We used half of D_1 as the initial content (yet published apps, noted A_{pubed}^{safe}) of the MMP. The set of apps playing the role of submitted apps consists of the other half of D_1 (noted A_{pub}^{safe}) on the one hand and all apps in D_2 (noted A_{pub}^{illeg}) on the other hand. Androguard and FsQuadra are accurate each time they are able to detect that A_{pub}^{safe} does not mimic an existing app while A_{pub}^{illeg} does. Concerning IMAD, it is accurate when it is able to detect the trusted entity behind the submitted app. Fig. 11 presents the evaluation results. We can see that all systems work perfectly on A_{pub}^{safe} apps. Concerning A_{pub}^{illeg} apps, Androguard and FSquaDRA provide poor results. This is explained by the fact that not all A_{pub}^{illeg} mimic apps which are yet in the MMP. IMAD does not suffer from this limitation since its search space is the web. The reader should refer to the previous section in order to have more explanations about the reported accuracy rate (81%).

For legitimate apps, we have 100% accuracy, which means that we are always able to identify the correct entity to contact in order to validate the submission. If this would not be the case, the submitter could contact the MMP for its submission to be handled manually (this would happen only for apps that have no existence at all on the net, which is very rare). For illegitimate apps, the 20% error rate means that (1) for 80% of these apps, we contacted the (attacked) trusted entity which is therefore informed about the attack and will not validate the submission, and (2) for 20% of the apps, another (wrong) entity is contacted and will not answer or

---

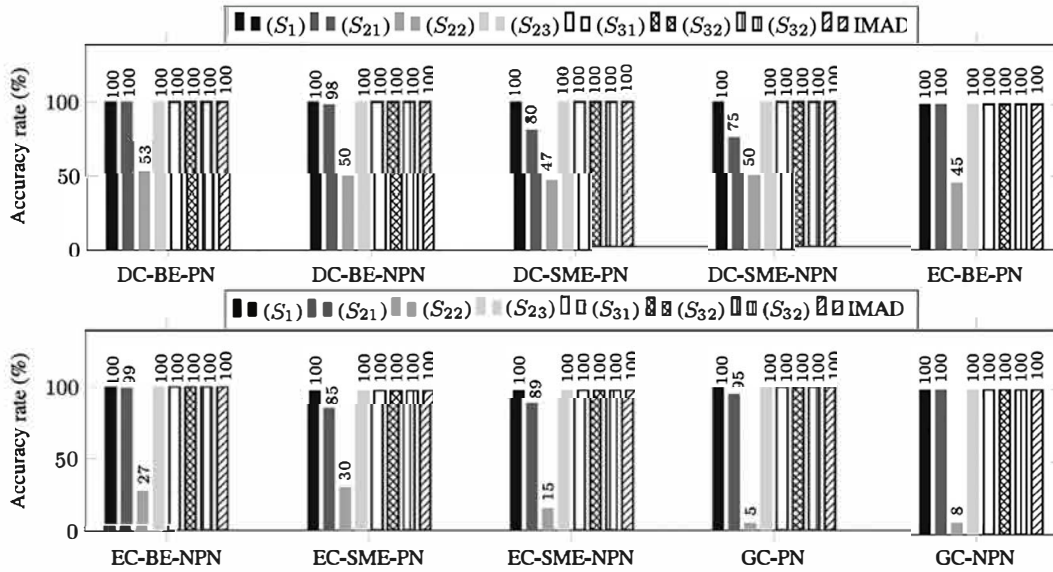[7] The algorithm we used is the most recent one at the time of writing this paper.

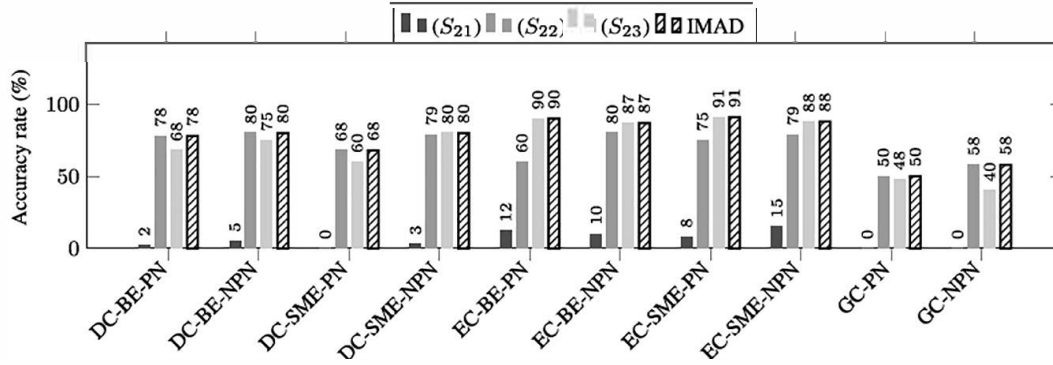**Fig. 9.** First experiment type: evaluation of each IMAD's step with safe apps.

Fig. 10. Second experiment type: evaluation with illegitimate apps from D_2.
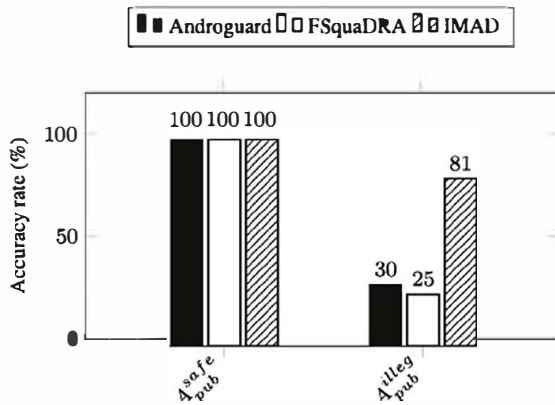

Fig. 11. Comparison of IMAD with Androguard and FSquaDRA, two reference illegitimate app detection systems.

will respond a rejection. The only threat is if an attacker is able to make noise on the web so that it will be more referenced than the trusted entity he attacks. This is a much difficult task for the attacker, as companies which make business with apps always have a communication strategy on the web.

5.2.2. IMAD configuration parameters

Table 2 summarizes the list of the most important parameters (noted L_i) that IMAD uses. The last column presents the best value we experimentally found, as follows. For each parameter we performed several experiments while increasing its value. The best value is the one from which the accuracy rate of the system does not improve. These experiments were first performed on a small data set and then validated on the entire data set. The small data set is disjoint from the entire data set which was used for validation. Fig. 12 presents the evaluation results, which justify the values reported in Table 2. For instance, using more than 20 documents (L_1) in the text search phase does not ameliorate the accuracy rate of the system (see the leftmost curve in Fig. 12).

5.2.3. Evaluation of IMAD with other search engines

We also evaluated our system with other search engines namely:

- Bing [40] (from Microsoft) for *appName* searches.
- Tineye [41] for *appLogo* searches.

***appName* searches with Bing.** The custom and programmable version of Bing are fairly new. However, the results obtained with this system are very poor (the accuracy rate is almost nil). Several reasons can explain these results. (1) Bing's ranking does not
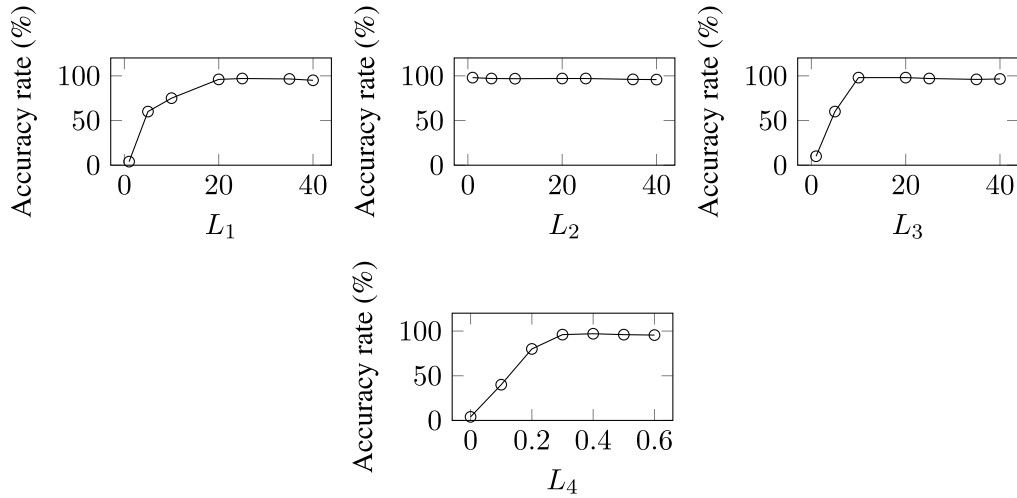
Fig. 12. Configuration parameters: estimation of the best value.

provide relevant documents at the top. (2) Its API does not allow the specification of the search scope. For instance, it is not possible to enforce a dbPedia search for determining trusted entities. Therefore, although the web search returns some results, it is not possible to classify those which are related to a trusted entity. (3) Small and medium-sized enterprises are almost never found. For these reasons, Bing is not mature enough to be used by IMAD.

***appLogo* searches with Tineye.** We have also tested Tineye [41] as the reverse image search system. We have made the following observations. (1) The Tineye search latency is too high (up to 10 s) in comparison with Google (up to 2 s). (2) Tineye does not implement a ranking system. Therefore, the entity popularity is not taken into account. In order words, the search of a popular entity's logo (such as Facebook) will not necessarily return Facebook's website at the top of the list. However, this system represents an acceptable alternative for Google.

5.3. Complexity

This section presents an analysis of the complexity of our approach. It is divided into two parts: local processing and global latency.

5.3.1. Local processing

We consider here the computations on data obtained from remote sources (google and DBpedia). We present separately the complexities of the text search and image search algorithms.

The text search algorithm mainly depends on the number m of documents obtained during the first google engine search. It includes the processing of the collected web pages to build a corpus (Section 4.5) whose complexity is $O(m)$, clustering (Section 4.6) based on Kmeans with a complexity tending towards $O(m^{(k+1)})$, with k being the number of obtained topics which is generally not more than 5. This step is followed by the elimination of unnecessary clusters (Section 4.7) with a complexity $O(mnwTN) + 0(mnNwAM)$ with n being the number of words in the dictionary, w the average number of wiki concepts relating to an important word, T the number of words in the RCL, N the number of important words, A the number of words in the abstract of a wiki concept and M the average number of words per document obtained during the first search. The step of eliminating useless documents (Section 4.8) has a complexity of $O(wT)$. Finally that of the extraction of the contact (Section 4.9) is done in $O(m)$. All the parameters except m being upper bound, the overall complexity of text search is $O(m^{(k+1)})$.

Finally, let us note that experimentally we have determined that a value of $m > 20$, does not improve the accuracy of the results.

The image search algorithm consists in the construction of the corpus with a complexity $O(m)$ and the clustering and the determination of the topic whose complexity is $O(m^{(k+1)}) + O(mnwTN) + 0(mnNwAM)$. The step of searching for the logo in the selected documents has a complexity of $O(m)$ and that of comparison of the logos $O(xmI)$ with x the number of logos retained for each document and I the complexity of the logo comparison algorithm. The sorting of the selected logos is done in $O(m \log(m))$. All this for a complexity tending towards $O(m^{(k+1)})$ for the same reasons as the complexity of text search.

The text search and image search evaluations on several applications have shown that scalability can be taken into account as depicted in Section 5.4.

5.3.2. Global latency

The overall latency is dominated by the time taken by each search on Google search engine and on DBpedia. All monetary costs of these searches are evaluated in Section 5.5.

5.4. Scalability

We evaluated the scalability of our system as follows. First, we evaluated the amount of resources consumed by each step. To this end, we considered three representative apps, namely: WhatsApp, Skype, and MPayOK (mobile money). Then we evaluated IMAD facing parallel checks, up to 10 apps at the same time (Google Play receives about one app every minute [42], thus checking 10 apps at the same time is enough). Figs. 13 and 14 present the evaluation results of the two experiment types respectively, interpreted as follows:

- (S_1) consumes a non negligible amount of CPU (this is not the case for other steps), see Fig. 13 leftmost. Its memory consumption level depends on the size of the checked apk (10MB–75MB in our experiments).
- When performing several checks at the same time, the processor is the most critical resource since it saturates before the main memory, see Fig. 14 left.
- (S_2) is the step which takes the most time, see the last picture in Fig. 13. This is because several web searches are performed during this step.
- the average time needed to check an apk is about 38 s.

These results show that the exploitation of IMAD does not require a particular hardware specification, even a commodity desktop can do the job.
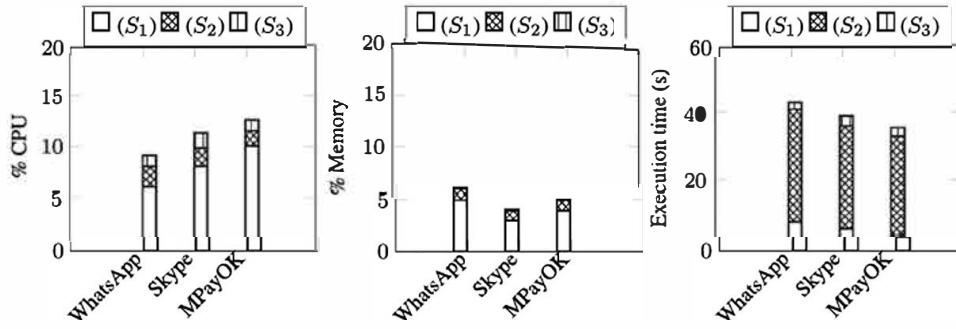
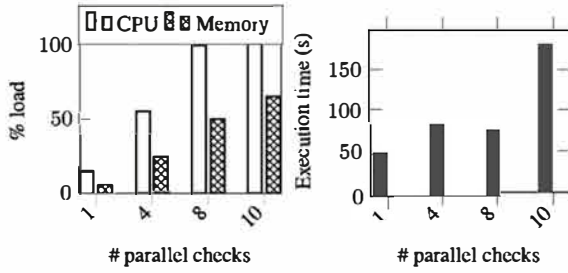Fig. 13. Evaluation of the amount of resources consumed by each IMAD's step.

Fig. 14. Evaluation of IMAD facing parallel app checking: (left) resource consumption and (right) execution time.

5.5. Cost evaluation

This section evaluates both the cost of deploying IMAD as an independent service (noted IMADaaS) on a commercial cloud and using Google search engine (which is not free).

Public cloud utilization cost. We evaluated the cost of hosting IMADaaS on a public cloud, Microsoft Azure [43] in our experiments. Our experimental machine (presented in Section 5.1) is comparable to the Azure *Standard_A3* virtual machine (VM) type. According to [42,44], Google Play as well as Apple store receives about 2,000 apps every day (about one per minute). Therefore, a single *Standard_A3* instance would be able to check within one minute the apks coming from up to five MMPs similar to Google Play and App store. This results to a bill of about $1571.28 per year.

Web search engine utilization cost. The number of searches (noted ns) needed by IMADaaS for checking an apk is given by the following formula:

$$ns = n + k \times m + 2 \times d + n \times d \qquad (6)$$

where n is the number of items in the corpus, k is the number of clusters, m is the maximum number of important words, d is the number of documents retained for the second chance step. In the case of our experiments, we found the following values: n=20, k=3, m=3, and d=4, resulting to $ns = 117$. We evaluated the cost of using Google custom search engine. The cost of a search in the latter is $0.0075, leading to $0.8775 the cost of checking an apk. Therefore, the cost for checking all daily apk from Google Play store or Apple store is about $1,755.

6. Conclusion

This paper presents IMAD, a security system capable to identify attackers who deploy malicious mobile applications in the name of well-known public companies which have not deployed their mobile app yet. The IMAD strength lies in the following. The attacker may distort the application label (name, logo) so that it

will cheat the detection system (the search engine). However, the label will be distorted in such a way that it is not able to fool the user anymore. We evaluated IMAD with up to 5000 enterprises, covering all enterprise categories (geographical location, activity, etc.). The evaluation results showed that IMAD can protect both big, medium and small-sized companies with an accuracy rate greater that 80%. We also compared IMAD with two systems (Androguard and FsQuadra). The evaluation results showed that IMAD does better as these systems for classical attacks while they were not able to detect the new attack discussed in this paper. The evaluation results also showed that our system is able to check all apps deployed within a day on Google Play or Apple App Store at a minimal cost (about $1,755).

This work opens many perspectives and it is worth mentioning some of them. First, the success of this approach depends on the ability to extract the correct email address of the person supposed to validate the application submission. This ability is difficult to experimentally evaluate and if it frequently fails, it would lead to a significant rate of app rejection. Alternatives to this scheme can be investigated. For instance, the submitter could provide an email address and IMAD would verify that this email address complies with the company it has identified, e.g. complies with the DNS domain of that company. One may see a contradiction with footnote 5 in Section 3.1 which argues that we do not trust the submitter. However, if he provides us with an email address and the domain name of this address is that of the trusted entity found by IMAD, the search for the contact can be bypassed and the system can directly send him the validation email. For example, if the given submitter address is submitter0@orange.com, and IMAD finds orange as the trusted entity of the application with the orange.com domain name, we are certain that the developer of the application is an employee of Orange and we will directly send him the validation email. Note that this scheme is secure as long as it is impossible to spoof a domain name. Another perspective is the integration of additional alphabets (we only experimented with Latin and Mandarin in this paper) which would improve the accuracy of IMAD. We can also consider using the app's description in our web search for the trusted entity to improve its accuracy. Finally, we are considering is the integration of a cache system which would allow to reduce the costs (in terms of time and money) of web searches.

References

[1] http://www.express.co.uk/news/uk/805003/WannaCry-virus-cyber-attack-NHS-security-computer-hack-bitcoin-Check-Point-
[2] Technologies. https://www.ben-evans.com/benedictevans/2015/6/13/the-lack-of-app-store-metrics.
[3] https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/.
[4] http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-fake-apps.pdf.
[5] Rooting Pokémons in Google Play Store, by Roman Unuchek.

[6] Orange Bank announcement. https://www.lesechos.fr/finance-marches/banque-assurances/0211888727149-orange-bank-lancement-prevu-mi-mai-2073226.php.

[7] K. Allix, T.F. Bissyande, J. Klein, Y. Le Traon, AndroZoo: collecting millions of android apps for the research community, Mining Softw. Repositories (2016).

[8] http://contagiominidump.blogspot.fr/.

[9] https://github.com/androguard/androguard.

[10] Yury Zhauniarovich, Olga Gadyatskaya, Bruno Crispo, Francesco La Spina, Ermanno Moser, FSquaDRA: Fast detection of repackaged applications, in: IFIP Annual Conference on Data and Applications Security and Privacy, 2014.

[11] Antonio Bianchi, Jacopo Corbetta, Luca Invernizzi, Yanick Fratantonio, Christopher Kruegel, Giovanni Vigna, What the app is that? Deception and countermeasures in the android user interface, in: IEEE Symposium on Security and Privacy, 2015.

[12] Yury Zhauniarovich, Olga Gadyatskaya, Bruno Crispo, Francesco La Spina, Ermanno Moser, FSquaDRA: Fast detection of repackaged applications, in: IFIP Annual Conference on Data and Applications Security and Privacy, 2014.

[13] Quanlong. Guan, Heqing Huang, Weiqi Luo, Sencun Zhu, Semantics-base repackaging detection for mobile apps, in: International Symposium on Engineering Secure Software and Systems, 2016.

[14] Fangfang Zhang, Heqing Huang, Sencun Zhu, Dinghao Wu, Peng Liu, View-Droid: Towards obfuscation-resilient mobile application repackaging detection, in: ACM Conference on Security and Privacy in Wireless & Mobile Networks, 2014.

[15] Kai Chen, Peng Wang, Yeonjoon Lee, Xiaofeng Wang, Nan Zhang, Finding unknown malice in 10 Seconds: Mass vetting for new threats at the google-play scale, in: USENIX Security Symposium, 2015.

[16] Giovanni Bottazzi, Emiliano Casalicchio, Davide Cingolani, Fabio Marturana, Marco Piu, MP-Shield: A framework for phishing detection in mobile devices, in: IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, 2015.

[17] Charlie Soh, Hee Beng Kuan Tan, Yauhen Leanidavich Arnatovich, Lipo Wang, Detecting clones in android applications through analyzing user interfaces, in: IEEE International Conference on Program Comprehension, 2015.

[18] Jon Oberheide, Charlie Miller, Dissecting the android bouncer, in: Summer-Con, 2012.

[19] Longfei Wu, Xiaojiang Du, Jie Wu, MobiFish: A lightweight anti-phishing scheme for mobile phones, in: International Conference on Computer Communication and Networks, 2014.

[20] Claudio Marforio, Ramya Jayaram Masti, Claudio Soriente, Kari Kostiainen, Srdjan Capkun, Personalized security indicators to detect application phishing attacks in mobile platforms, ArXiv e-prints, 2015.

[21] Zhi Xu, Sencun Zhu, Abusing notification services on smartphones for phishing and spamming, in: Usenix Workshop on Offensive Technologies, 2012.

[22] http://www.easychair.org/.

[23] http://connortumbleson.com/2015/10/12/apktool-v2-0-2-released/.

[24] R. Smith, An overview of the tesseract OCR engine, in: Ninth International Conference on Document Analysis and Recognition, 2007.

[25] https://cse.google.com/cse/all.

[26] Anil K. Jain, Data clustering: 50 years beyond K-means, Pattern Recognit. Lett. 31 (8) (2010).

[27] Rakesh Chandra Balabantaray, Chandrali Sarma, Monica Jha, Document clustering using K-means and K-medoids, CoRR abs/1502.07938, 2015.

[28] David Arthur, Sergei Vassilvitskii, k-means++: the advantages of careful seeding, in: Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2007.

[29] dbPedia.wiki.dbpedia.org.

[30] World Intellectual Property Organization. www.wipo.int.

[31] Sandeep Tata, Jignesh M. Patel, Estimating the selectivity of tf-idf based cosine similarity predicates, ACM SIGMOD Record 36 (2) (2007).

[32] http://aspell.net/metaphone/.

[33] Li Yujian, Liu Bo, A normalized levenshtein distance metric, IEEE Trans. Pattern Anal. Mach. Intell. 29 (6) (2007).

[34] http://link.fobshanghai.com/download/googlesearchqualityevaluatorguidelines.pdf.

[35] Ming-Ming Cheng, Niloy J. Mitra, Xiaolei Huang, Philip H.S. Torr, Shi-Min Hu, Global contrast based salient region detection, IEEE Trans. Pattern Anal. Mach. Intell. 37 (3) (2015).

[36] Inderjeet Mani, Alex Yeh, Sherri Condon, Learning to match names across languages, in: Theory and Applications of Natural Language Processing, Springer, 2008.

[37] Current Android Malware. https://forensics.spreitzenbarth.de/android-malware/.

[38] Longfei Wu, Xiaojiang Du, Jie Wu, Effective defense schemes for phishing attacks on mobile computing platforms, IEEE Trans. Veh. Technol. 65 (8) (2016).

[39] Luka Malisa, Kari Kostiainen, Srdjan Capkun, Detecting mobile application spoofing attacks by leveraging user visual similarity perception, in: Seventh ACM on Conference on Data and Application Security and Privacy, 2017.

[40] https://www.bing.com/.

[41] https://www.tineye.com/.

[42] https://www.quora.com/How-many-new-apps-are-added-to-Google-Play-everyday.

[43] Microsoft Azure. http://azure.microsoft.com.

[44] http://www.pocketgamer.biz/metrics/app-store/submissions/.

Lavoisier Wapet received a Master degree from Polytechnic National Advanced School of Yaoundé University in 2016. He then started a Ph.D. cursus in 2017 at Polytechnic National Institute of Toulouse, France. His main research interests are in Virtualization, Cloud Computing, Operating System and Security.

Alain Tchana received his Ph.D. in computer science in 2011, at the IRIT laboratory, Polytechnic National Institute of Toulouse, France. Since September 2013 he is a Associate Professor at Polytechnic National Institute of Toulouse, France. He is a member of SEPIA research group at IRIT laboratory, Toulouse. His main research interests are in Virtualization, Cloud Computing, and Operating System.

Giang Son Tran received his Ph.D. in computer science in 2014, at the IRIT laboratory, Polytechnic National Institute of Toulouse, France. Since August 2014, he is a lecturer at University of Science and Technology of Hanoi and a researcher at ICTLab. His research interests include operating system, mobile platforms, machine learning and high performance computing.

Daniel Hagimont is a Professor at Polytechnic National Institute of Toulouse, France and a member of the IRIT laboratory, where he leads a group working on operating systems, distributed systems and middleware. He received a Ph.D. from Polytechnic National Institute of Grenoble, France in 1993. After a postdoctorate at the University of British Columbia, Vancouver, Canada in 1994, he joined INRIA Grenoble in 1995. He took his position of Professor in Toulouse in 2005.