Luigi Tuttobene 4522729

Assessment of the Security Level of Two Third Party Android Application Markets

The spread of Android's user base have attracted attackers' attention to the platform. As a result, over the last decade, the phenomenon of mobile malware have rise. In this paper I am going to analyse the problem of mobile malware from the perspective of the economy of cyber security, investigating actors involved and the relations among them. Secondly, I will analyse the distribution of malicious code in two third party application markets as I believe that this kind of stores present a higher infection rate than the official PlayStore. The results, in line with the hypothesis, show a infection rate around 10%, an order of magnitude higher than Google's store.

Introduction

The huge spread Android has undergone over the last decade and its openness nature have made it a target for mobile malware.
From 2010, when Geimini, the first malware for Android, has been detected, the number of malware categories have grown to an surprising high rate. Only in December 2011, for example, Kaspersky Lab detected more than 1000 new trojan targeting Android, more than all smartphone viruses found between 2003 and 2010 (Arabo & Pranggono, 2013).
Android *ecosystem* (word picked up in the literature to highlight the symbiotic relationships among actors involved (REF)), is build around the interaction of four main actors, namely market operator, user, app developer and attacker.
As in many problems regarding cyber security, the main issue in assessing the security of application stores deals with the environment in which these actors interact, which is dynamic and characterised by a high level of uncertainty.
The contribution of this research is mainly to define metrics to measure the distribution of malware in two unofficial app distributors.

Research Objective

The aim of this paper is to assess the security level of a couple of third-party application marketplace, namely Baidu App Store and Qihoo 360 Mobile Assistant, respectively the first and second largest mobile application distributors in China (https://www.techinasia.com/10-android-app-stores-china-2014-edition). I hypothesise that these third party application stores have a higher rate of infection.
In order to achieve objective and representative results, the analysis is articulated into three sections. To begin with, an overview on Android ecosystem will discuss, from an economical perspective, the interdependencies and the dynamics that make this assessment difficult. In a second moment, I will introduce the tools used to define the level of security of the marketplace. In particular, I will first define metrics to effectively measure security, to then discuss their reliability. Lastly, the results and their interpretation are presented.

Literature Review

Given the rapid spread of mobile malware and its security implications, it is not surprising the existence of a significant amount of work in the literature from different point of views.

To start with, I am going to delineate the main characteristics of what has beed defined *Economy of Malware*. Bauer & van Eeten, (2008), offer a solid and comprehensive analysis of the economics behind cyber security. The hart of their book, at the same time dated and premonitory, is an analysis of the market externalities and of the incentives, cost and benefits that affect the decision makers.
They identify the origin of the challenge to define an optimal level of security in the discrepancy between the cost and benefits perceived to an individual and aggregated level. As a matter of fact, information systems are characterised with positive externalities, as investments in security by an agent might benefit others in the ecosystem (with the extreme of free rider behaviours), and negative externalities as well, as for instance inadequate efforts done by an actor affect others in a negative way.
Moreover, there are feedback loops, such as reputation effects, that bring some of the costs imposed on others back to the party who caused them.
In addition to the problem of externalities, many products and service suffer positive network effect: their value increase as the user base enlarges. Companies operating in similar position, it is argued, will react with product and price differentiations, in order to profit from first mover advantage, versioning and similar strategies. As a result of network effect, when dealing with a tradeoff between time-to-market and security, suppliers will postpone security testing to speed to market.
Bauer & van Eeten, (2008) argue therefore that information security is a quasi-public good, and conclude that there is an asymmetry in the relation between cyber crime and security. While on the one hand, a higher level of security will bring about higher cost of cyber crime, reducing its benefits, on the other hand, higher levels of cybercrime will raise the costs of security, but their impact on the security level is enigmatic. Since these mechanisms work in the same direction, increases in security will reduce the level of cybercrime.
Lastly, the authors identify three scenarios that might characterise the value net connecting actors:

i) No externalities;
ii) Externalities that are borne by agents in the value net that can manage them;

iii) Externalities that are borne by agents who cannot manage them or by society at large.

The relevance of these situations will be discussed immediately after the introduction of a model to describe Android ecosystem.

Arabo & Pranggono, in (2013), investigate emerging trends and challenges in mobile security, offering a general framework which can be a good base to derive a model for Android.

As illustrated in *Figure 1*, four are the most important actors:

1. *user* considered in both senses of individual or for profit and not for profit organisations of different sizes.
2. *a market operator* who manage the marketplace, the legal persona answering for the contents, i.e. Google for the official Android Play Store. He is the mediator between the consumer and the software writer.
3. *app developers*, usually organised in small and medium enterprises. They include both anti-viruses and entertainment/productivity application suppliers.
4. *attackers*. As cyber security and cyber crime dynamically co-evolve, an important role is played by attackers.

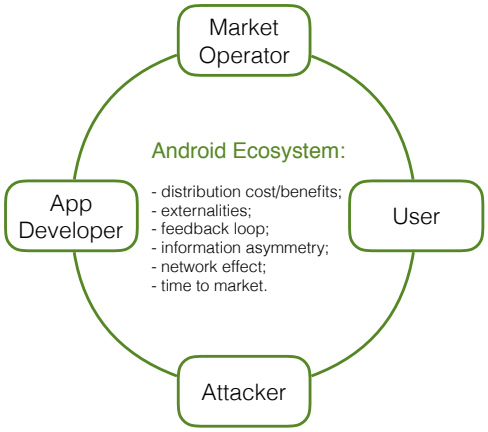


Figure 1. Representation of Android ecosystem.

A brief overview of the main feature of the market follows. In a previous assignment, the relationship between these actors have beed discussed, highlighting the manifestations of the externalities problem.

Arabo & Pranggono, (2013), referring to mobile platforms, argues that the distribution of cost and benefits is non linear, leading to difficulties in establishing causality. Moreover, in (Bauer & van Eeten, 2008) it is shown that similar interdependencies have oftentimes resulted in a blame game among actors accusing each other for insufficient efforts.

The security within the system is affected by information asymmetry. This is particularly manifest in the case of AntiVirus software. App developers, in fact, have more information about their product compared to the users.

The coevolution of cyber crime and security mentioned in (Bauer & van Eeten, 2008) find an instance in zero day exploit market, where novelly disclosed vulnerabilities are sold to attackers rather than reported to the responsible actor.

The aim of the foregoing overview is to show that in similar conditions externalities generated in a sector of the value net are internalised by other actors, resulting in outcomes of security decision which deviate from the social optimum. Consequently, the value net of Android ecosystem might be interpreted in light of the second scenario theorised in (Bauer & van Eeten, 2008).

Surely, there are other actors orbiting around Android galaxy, like governments, Internet Service Providers (ISPs), non governmental organisations (NGOs) and consumers associations. It is worthy to spend a couple of words about governments that in the present case have played a role. The decision to block Google in China allowed local companies to take its place as market operator, resulting in many third party application markets. Nevertheless, third party markets addresses what can be considered Android's core feature: the openness of its code and the high customisability. Several "unofficial Play Store" have grown over the past decade, and many of them present a larger portion of malicious code if compared to the official (Kikuchi *et al.*, 2016).

Security features common to Android involve process and file system isolations; application or code signing, ROM, firmware and factory restore; and kill switches. Nevertheless, the most importan issue with Android OS is it relies on the end-user to decide whether an app is safe or not. The user cannot rely on the OS to protect themselves from malware. I have already mentioned how market operator strategies to gain marketshare as big as possible by offering application as many as possible might lead to trade-off between speed to market and security.

Similar speculations are done by Porter Felt *et al.*, (2011), who have surveyed Android malware and analysed the incentive behind part of the malicious code. They identify different typology of malware: aimed to sell user information or steal his credentials; premium-rate calls and SMS; SMS/email spam; ransom; advertising click fraud. Among these categories, botnets are considered the biggest challenge, as they are used to spread spam or carry out DDoS attacks.

In the case of mobile application distributors, the control mechanism consists in subsequent verification steps from the moment the developer provides the software until the commercialisation of the app. In addition, aggregated and anonymised data sent from user devices are used to monitor the general state of the market. Android Security Year 2015 Review (Google, 2015) shows the diagram and explain into details the code review and feedback loop system Google has established.

There is a growing literature that has begun posing care to third party markets, distributors that repackage applications from Google Play and sell them.

Kikuchi *et al.*, (2016) explain that during this operation malicious lines of code can be insert to turn the application into a malware, attaching also interesting data about the most exploited vulnerabilities and the most common malware in Android ecosystem, and arguing therefore that the number of app repackaged might be a candidate indicator.

To do that, they rely on the infection rate, a metric largely discussed in (Truong *et al.*, 2014).
This rate, definable as the percentage of infected apps over the population of a market, is pretty general, but represent the first attend to assess the level of security of a market. To define this construct, it is necessary to provide a criterion through which assess the level of infection of a software. In the literature, popular choices are a set of AV and malware detectors such as Avast, Avira, Baidu-International, Kaspersky, Qihoo-360, VitusTotal and similar. Truong *et al.*, (2014) analyses also this aspect in his paper, concluding about the result, shown in *Figure 2*, that: *"A significant fraction of each dataset contains malware samples included only in that set, leading to the question whether there is any common agreement about what constitutes malware"*.

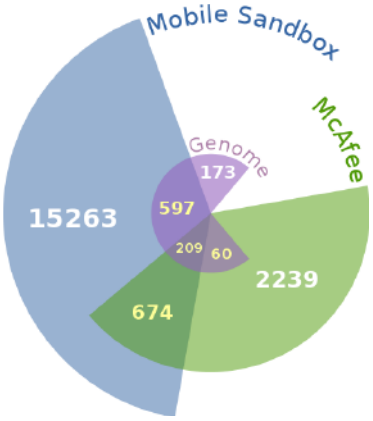


Figure 2.
Sizes of the three malware datasets and the extent of overlaps among them.
Source: Truong et al., 2014

(Porter Felt *et al.*, 2011) investigate also about the most applied techniques to detect malware, discussing about the efficiency of permission-based classification. Android malware commonly requests the ability to directly send SMS messages, which is uncommon among non-malicious applications. However, they were unable to identify any other permission-based patterns for malware classification.

Once agreed on a satisfying criterion of infection (that is likely to include a threshold to filter out the noise), the result will be something similar to what in (Kikuchi and al., 2016) is named *Positive Detection Rate*:

$$PDR(S,a) = |\{av \in S | av(a) = malicious\}| / |S|$$

where $av\ (a) = malicious$ means that an AV product av has detected application a as malicious, and S is the set of AV and detectors.
At this point, it is possible to finally formalise the infection rate as *Malware Presence Ratio*

$$MPR(m, S, M) = |\{a \in M | PDR(S, a) > m\}| / |M|$$

where M is the set of collected apps for a market and m is the chosen threshold value for the positive detection ratio.
Another interesting indicator discussed in (Kikuchi and al., 2016) is the *Malware Download Ratio*, defined as

$$MDR(m,S,M) = \Sigma_{a \in M'} DL(a) / \Sigma_{a \in M} DL(a)$$

where $DL(a)$ indicates the total number of downloads of the app a. M' represent the set of all apps in the app set M which are determined as malware by the threshold m.

In light of this, the infection rate, *MDR* and similar metrics are half way between vulnerabilities and incidents, as they can be based on the vulnerabilities that might be exploit or the number of application which are reported as malware (incidents).

So far, the problem of Android malware has been framed in an economical context, actors have been identified and their relations investigated. Then, metrics to measure the diffusion of malware in the market have been defined. Now, the methodology for the data-collection is presented, followed by a discussion about the findings.

Methodology

The main tool used for the metrics and statistics are the VirusTotal libraries and the data set provided. The dataset consists of a list of most popular android apps on the Baidu and Qihoo 360 android app marketplace. The dataset is subdivided into 4 parts by separating both markets and subdivide it between two groups: "Games" and "Software". Games consist of any software meant for entertainment. The software group consists of any other software (mainly utilities and productivity software). Every app is also assigned a category that it corresponds with. The dataset consists of a selection of around 300 most popular apps within each category, for a total over 50,000 android apps.

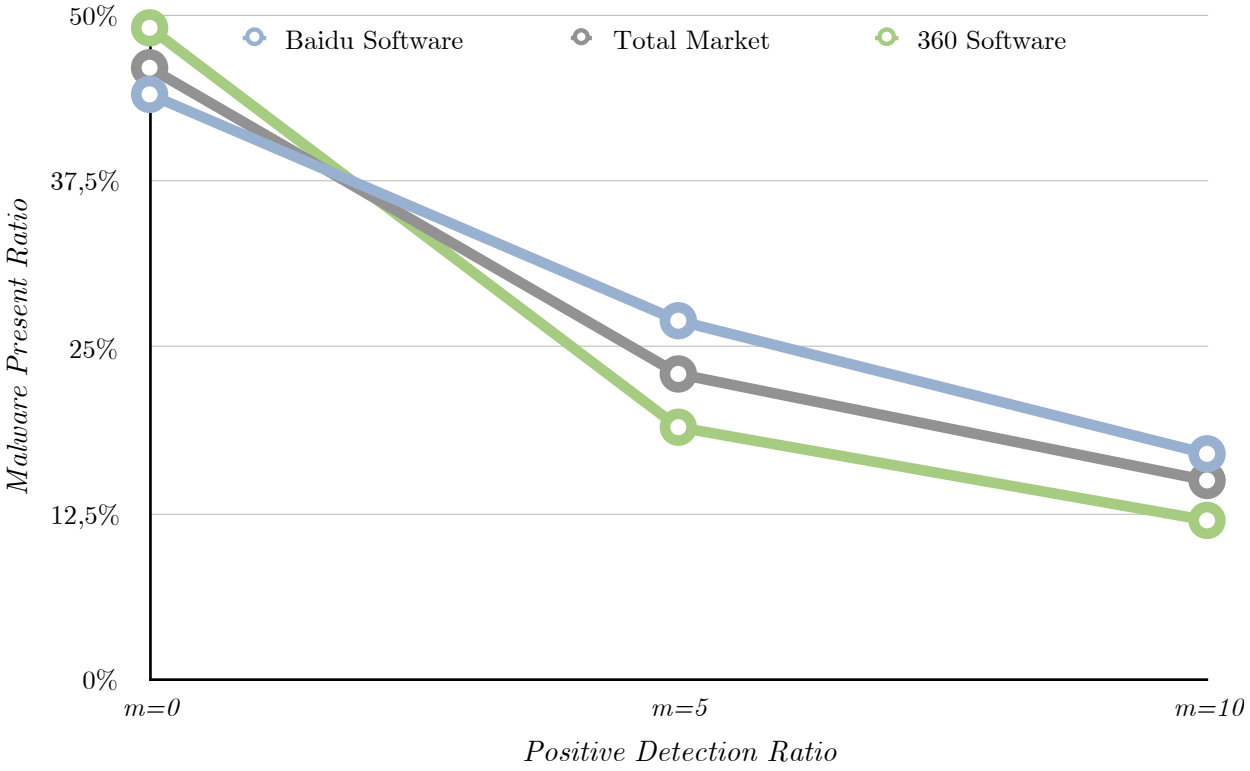
Without the use of some kind of malware detection the dataset is insufficient for any malware evaluation. For this end VirusTotal is used (virustotal.com). It is an online service for malware detection. It can check links/websites and uploaded files for any malware. VirusTotal is powerful because it pools resources from more than 50 anti-virus services to detect any malicious code. What one firm is not able to detect some others might. Therefor the results from VirusTotal is not only used to check for the presence of malware but can also be a useful check on how well it is detected among all the sources. This way, software can not only be checked for malware but also checked on how "elusive" the malware is.

The only way to scan such a large amount of applications through a site is by automation. VirusTotal provides a web API to make scripted processing possible. For bulk processing a private API key is necessary. Unfortunately due to circumstances it was not possible to receive a

private API key in a timely fashion. Without a private key query rates were limited to 4 scans per minute.
This would mean more than eight days for a single machine to go through the entire list. Another limitation to the public API key is that there is a file size limit of 32MB, which is an issue since a large part of the apps are >32MB. Because of these obstacles it was decided to only take a representative sample using a confidence level of 95% and a margin of error of 5%.
Taking standard deviation 0.5 (normal data) a representative sample size N = 385 is calculated. The sampling strategy is weighted sampling of each category. To get results applicable for the chosen metrics, care has been taken so that less downloaded apps are just as well represented as popular apps. This is done by sectioning each category into clusters of download counts. Within these clusters random samples have been taken with the help of excel.
The selected samples are then finally scanned for any malware and processed.

Unfortunately, the only sets we have managed to save the complete scan of (most popular detection included) are the software segment of Baidu and Qihoo 360. The following analysis is done on the dataset.
Different values for the threshold m in the Malware Presence Ratio, namely *m=1, m=5, m=10*. For each value, MPR and MDR are computed. Furthermore, the distribution of malicious code among various category is observed too.

Figure 3. Malware Present Ratio for three different Positive Detection Ratio



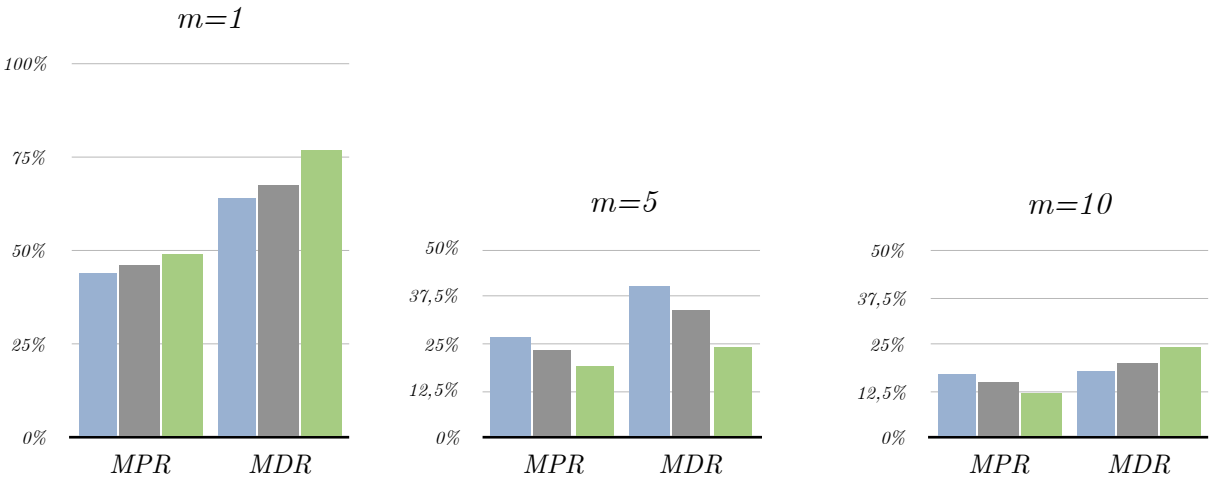


Figure 4. Malware Presence Ratio and Malware Download Ratio for the three considerate cases.

Results

Figure 3 provides an overview of the result. There, for each value of m, the MPR is calculated for the software segment of Baidu and Qihoo 360. The line in the middle represent the aggregate market.
The shape of the curve indicates that the use of actually increasing values of m filter out false positive. The spontaneous question is therefore what is a reliable choose for defining an application malicious?

As Kikuchi *et al.*, (2016) point out, the trade-off results or in a high number of false positive for low values of m, but on the other hand the infected rate is almost zero for higher PDR.
F-SECURE report (2013) found a MPR around 0.1%, writhing Google's PlayStore. This number lead to a PDR threshold around 30% (Kikuchi *et al.*, 2016).
During my analysis, I have noticed that the average number of check VirusTotal does on each application is barely fifty. For $m=10$ on 50, the PDR would be 20%, which is close to the foregoing mentioned 30%.

Figure 5a. Distribution of Malware by Category for m=1

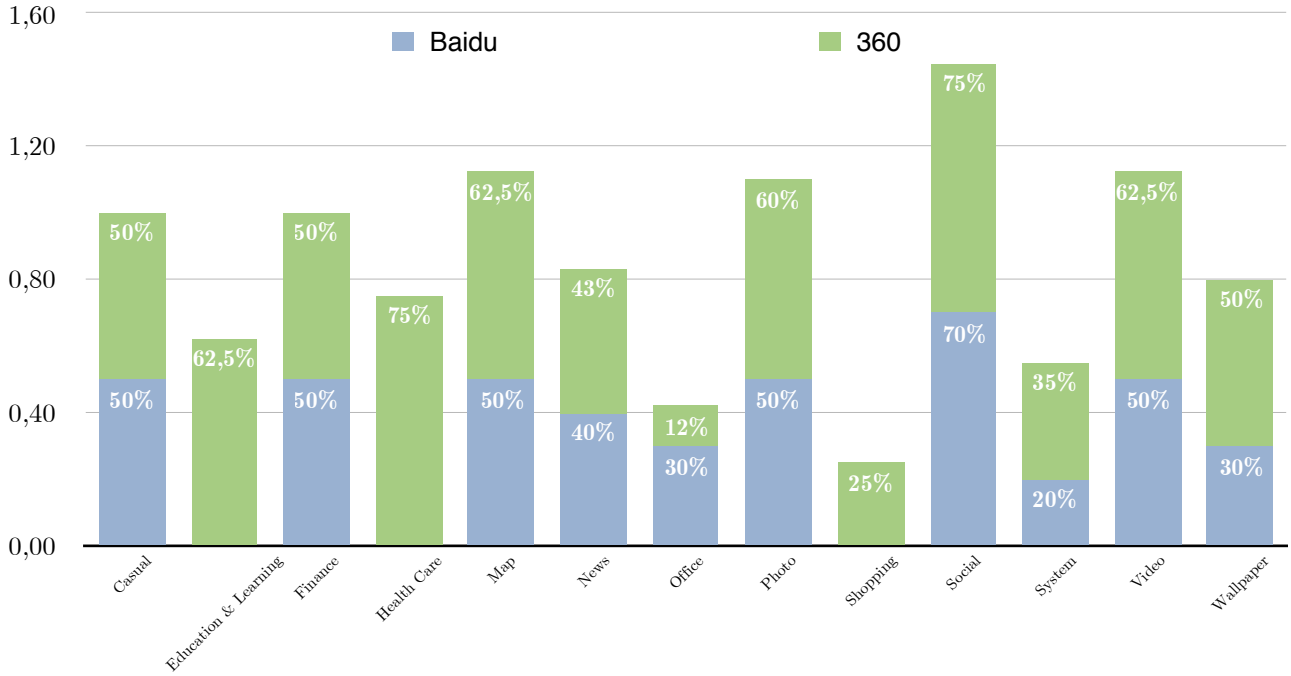


Figure 5b. Distribution of Malware by Category for m=5

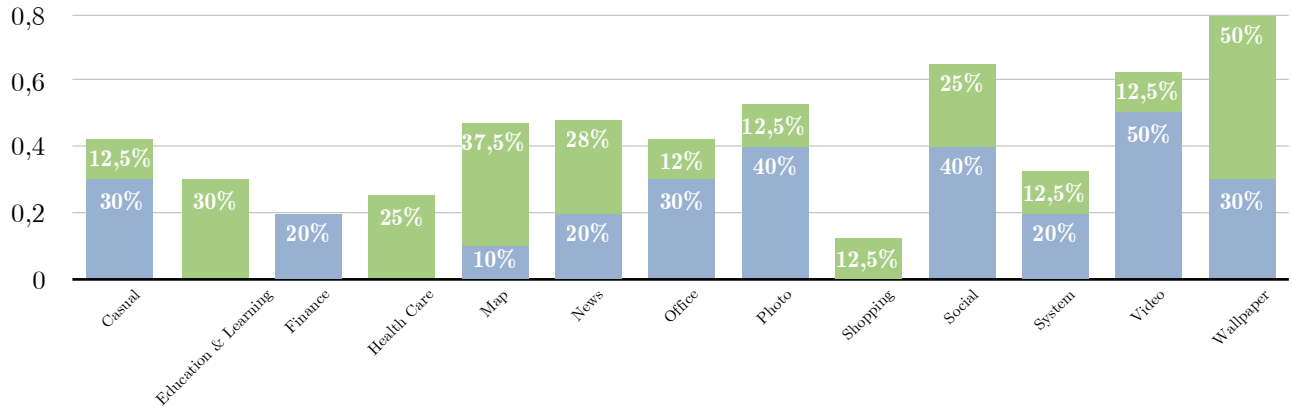
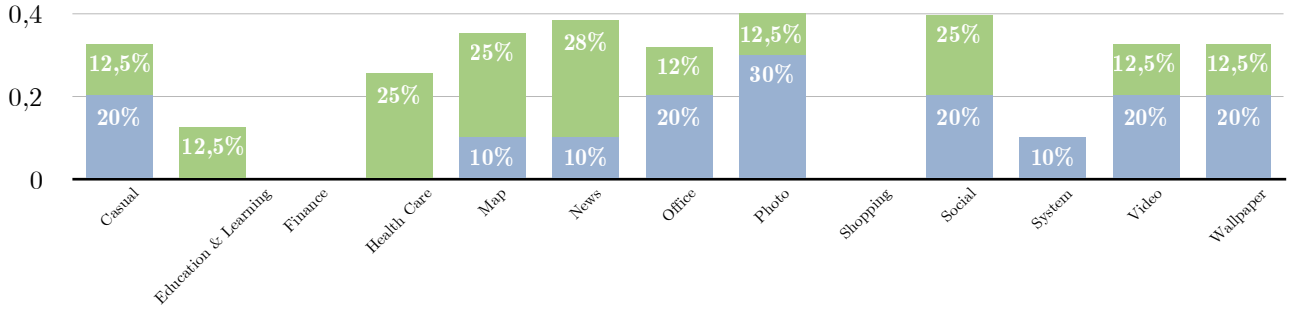


Figure 5c. Distribution of Malware by Category for m=10



I claim that the result for *m=10* might be representative to assess a reliable MPR.

For *m=1*, MPR and MDR are roughly at 50%, and the distribution of malicious code by category is shown if *Figure 5a*. The most popular detection for this case are various type of trojan (the most common is *Trojan.AndroidOS.Generic.A*), malware relative to what is named *Android.Jiagu*, and *Android/G2P*, spread uniformly on both market.

For *m=5*, MPR and MDR are still high, above 25%. In this case, the most popular detection are still various trojan and *Android/G2P*, together with *Android.Riskware*. The distribution on the various category is represented in *Figure 5b*.

Lastly, for *m=10*, MPR and MDR go below 25%, which is still a relatively high percentage of infection. The third graph of *Figure 4* shows that Baidu is a slightly more infected market, but Qihoo 360 present a higher Malware Download Ratio. *Figure 5c* illustrates that many of the categories in both market present a significant percentage of malware (in six categories out of ten, Baidu has around 20% of malicious code, for 360 the categories with a high percentage of malware are less, 4/13, but those infected present a larger part of malware, 25/28%). The most popular detection are the same of the case *m=5*.

Limitations

The main limitation of the above analysis is related to VirusTotal, the tool used to detect malware. As a matter of fact, despite is a useful and free application, its database might be not updated (an application might flag as malicious on the base of past researches). This, and other issues related to VirusTotal weaken the internal validity of the present work.
Moreover, external validity is treated not only by the fact that other market operator might pursue different strategies (i.e. Apple with iOS developed a different strategy, not based on the openers of the system), but also by the sampling techniques. In fact, I have analysed only the software segment of Baidu and Qihoo 360, and the results might be not largely generalisable.
Lastly, a more detailed analysis of the malicious behaviour of the observed application is advisable.

Conclusions

All in all, in this show report I have analysed the presence of malicious code in two third party application distributors.
I have firstly introduced the issue in the literature review, showing the relevance of the problem and defining constructs to effect measurements. Secondly, the analysis have been presented. This analysis shows, accordingly to the hypothesis, that third party app stores have a higher portion of malicious code. The most detected threats were mentioned as well.

Acknowledgements

I would like to thank the member of the group I have been working in the last two months for the good research work we have did together that lies at the base of this research.

References

Bauer, J. & van Eeten, M. (2009). Cybersecurity: Stakeholder incentives, externalities, and policy options. *Telecommunications Policy*, 33(10-11), 706-719.

Arabo and B. Pranggono, Mobile Malware and Smart Device Security: Trends, Challenges and Solutions 2013 *19th International Conference on Control Systems and Computer Science*, Bucharest, 2013, pp. 526-531. doi: 10.1109/CSCS.2013.27

F-SECURE. Threat report h2 2013. https://www.f-secure. com/documents/996508/1030743/Threat_Report_H2_2013.pdf

GOOGLE. Android security 2015 year in review. http://source.android.com/security/reports/Google_Android_Security_2015_Report_Final.pdf.

Kikuchi, Y., Mori, H., Nakano, H., Yoshioka, K., Matsumoto, T., & van Eeten, M. Evaluating Malware Mitigation by Android Market Operators. *9th Workshop on Cyber Security Experimentation and Test (CSET 16)*. USENIX Association.

Porter Felt, A., Finifter, M., Chin, E., Hanna, S., & Wagner, D. (2011, October). A survey of mobile malware in the wild. *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices* (pp. 3-14). ACM.

Truong, H. T. T., Lagerspetz, E., Nurmi, P., Oliner, A. J., Tarkoma, S., Asokan, N., & Bhattacharya, S. (2014, April). The company you keep: Mobile malware infection rates and inexpensive risk indicators. *Proceedings of the 23rd international conference on World wide web* (pp. 39-50). ACM.