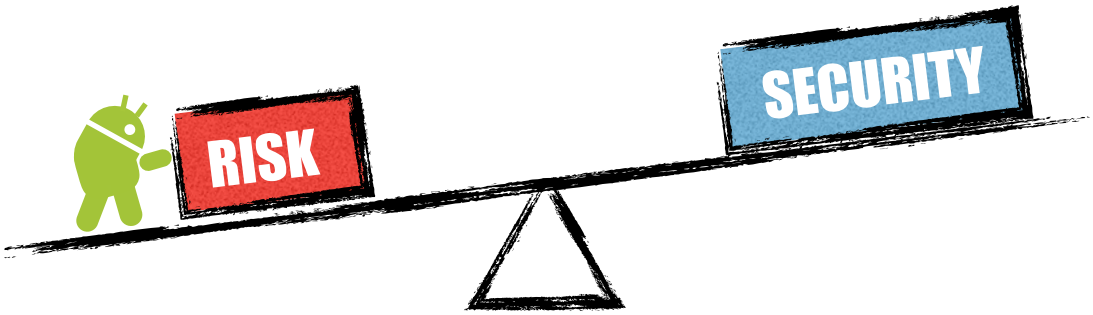
Economics of Cyber Security
Assignment block 3:

# Strategies To Cope With Risk Within Android Environment




Group 5:

| | |
|---|---|
| Raditya Arief | 4500318 |
| Eren Celik | 1550985 |
| Ana Guerra | 4520084 |
| Luigi Tuttobene | 4522729 |

TU Delft

## Introduction

In our first assignment, three metrics were used to analyze the security issues of Android applications. (1) Malware detection ratio which is the average of malware infections of every app in a particular market; the higher the malware detection ratio, the worse the security control (2) Malware presence ratio which is the rate of malware-infected apps in particular marketplace (3) Malware download ratio which closely measure the extent of harm spread from a particular app marketplace. Our results also show that both Chinese's application markets, Baidu and 360, are fairly compromised, on average there are more apps that contain malicious code rather than clean applications. In addition, malware download radio is higher than malware presence ratio therefore, malware is a bigger part of the eco-system than that the malware presence ratio implies.
This paper is built on our previous work. In the first section, the problem owner of security issues of Android application is identified. Second, we establish differences in the security performance of our metrics. Third, risk strategies of the problem owner are identified. Then, additional actor who are able to influence the security issues of Android applications are identified, and their risk strategies are analyzed. After that, Return on Security Investment is utilized to assess how financial decision influence security decision.

## The Problem Owner

Due to its widespread acceptance, Android mobile operation system has become more attractive for criminals to target Android devices. One of the growing threat of these devices is malware, which as we showed in our first assignment, is highly presented in android marketplaces such as Baidu and 360 Mobile Assistant. Usually, criminals infect android app with different families of mobile malware and use those compromised apps to spread malware on devices using android application markets. Therefore, Android users are dependent on application markets to ensure its security.
Android markets have special characterizations because, it is here, where application providers and applications users converge, and because they are the main mechanism used by attackers to spread malware. Moreover, our previous work showed how compromised these markets are. Therefore, marketplaces can be considered as the problem owner of customer's security issues due to malware. In our specific case, the two markets that are been analyzed are located in China where Google, and thus the official Android application market, and every service got blocked in mid-2014. Users in this country get android applications from "alternative" markets with not much options regarding the security of the content. In addition, it is not clear the level of liability of this market regarding security issues in android applications and compromised devices. In its disclaimer section, Baidu only addresses the application or violation of legitimate rights and interest of third parties, including but not limited to patent, trademarks, copyrights and neighboring rights, portrait rights, privacy, and reputation, however; no information is given regarding losses of end users due to compromised applications from this market. The same is true for 360 Mobile Assistant.
As we state above, in this paper, we will consider app markets as the problem owner, however, any actor which actions can improve the current situation have a degree of liability with the consequences of malware spread. For instance, users who decided to download applications (or APK files) from third-parties to avoid payments, and without considering the security threats of their

decisions need to get responsible for its actions. This is the case also for developers who cannot ensure the integrity of its applications

## Difference in Security Performance of the Metrics

One of the most interesting comparison, which shows the relevant differences in security performance, is by contrasting the result of our metrics with the information provided by the official Android application market Google Play. According to Google Android Security Report (2015), the percentage of app carrying malware on the official store was 0.1%. This result enormously differs from the result of our metrics in the two Chinese markets (43.6% for Baidu Software, 76.2% Baidu Games, 49% 360Software, 45.5% 360Games). Although there are some limitations in our results due to how VirusTotal get the malware information, the difference is still shocking.
Google is committed to ensuring that Android is a safe ecosystem for users and developers. This is done by providing a security app and services for Android, working on a constant strengthen of the core platform, and fostering an ecosystem rich with security innovation, and sharing data about the security of Android ecosystem (Google, 2015). All of these activities show that Google is investing to maintain its market as clean as possible to keep its position in as a technology leader. However, third-parties such as Baidu and 360 have different objectives. In addition, they operate in a country where the official app store is not there, and where users need their services and have no better option that guarantees its security.

## Risk Strategies

This report centers on the app-market and sees the app-market owners as the problem owner. According to the data both app-markets suffer from a high amount of malware. If this issue is deemed to be detrimental to the company goals it has to implement certain strategies to prevent and mitigate the problem of malware. Before any strategy can be formed the life cycle of an app must be understood.
A developer first needs to be registered with the app-market. After registration the app is uploaded to the market. The market owner checks in a very quickly if content is complying with their terms (such as the use of an accepted SDK and security checks). The app is finally listed and downloadable off the app-market. The developer at this stage is still able to update and pull their content.
The app-market owners (APO) play a significant role as the hub for apps. Nonetheless their direct risk reduction strategies are limited to control strategies and insurance, in which they control access and distribution of apps. These control strategies fall in to the categories of Prevention, Interdiction, Detection and Response.

### PREVENTION

The old saying goes as: "Preventing is better than curing". This also rings true for any cyber security issue. Preventing any calamity is preferred than fixing the damage it has caused. This idea can be translated to the app-markets too. The goal is to decrease the amount of malicious apps turned in by attackers masking as developers. The strategy must prevent attackers seeming like legitimate developers. A possible "carrot" solution can be a program as a "trusted developer" certification. This

is a program that rewards developers that let their firm to be audited and hold a clean track record. In being trusted they are then rewarded for it by better rankings etc. According to Sabathai et al ???? this method is very hard to implement. The huge numbers involved makes it hard to make this a preventative measure. Problem with certification is that attackers will also try to abuse the certification system. On the "stick" end of the solutions is a punishment strategy that dolls out fines or holds back deposits when there is foul play. Of course a punishment system is only effective if fines are more costly than profit made with malicious content and that fines must be enforceable. A stick is only a threat if you actually can be hit with it.

## INTERDICTION

Interdiction is actually a part of prevention, but the mechanism is so different that it warrants its own section. Interdiction in the context of app-market owners is the process in which an app is checked after it has been offered for publishing and then blocked from being published on the market when malware is detected/found. Code can be manually reviewed, which is time consuming and costly. A large skilled workforce is needed to check if certain code seems to be malicious. The alternative to manual checking is automated malware scanning and sandboxing. Sandboxing involves the app-market owner installs and runs the app in a special protected environment that closely monitors if the app shows malicious behavior. Malware scanners will scan for known malware in apps.

## DETECTION

Even if prevention and interdiction are done at a high intensity it is still possible for malware to slip through the cracks. It is now important that a detection mechanism is present that will detect and pull malicious apps from the market. In essence there are modes of detection, active and reactive. In the active mode, the appmarket owner analyzes app behavior sent in by monitoring plugins that runs on the platform or is added by the app-market owner. An AI can flag the results for further investigation. On the reactive end, market owners can use information provided by antivirus companies or user review/flagging as a data source to act upon.

## RESPONSE

Once a compromise is found the problem owner has to make a decision on how to respond. Depending on the business strategy, philosophy and malware type it can choose to do nothing, pull the app from the market and issue a warning to the installed user base or remotely uninstall it off of android devices.

## INSURANCE

An APO can also opt to shift its risk to an insurer.
Publishers, historically, benefited from Media Liability insurance covering defamation, invasion of privacy, and copyright infringement. Similarly this insurance has also moved to the digital age under the name of Electronic Media Liability insurance. Other cyber risk insurance policies can cover damages incurred by cyber attackers. The challenge here is that the authors did not find any concrete insurance that would cover the app-market construct. The question is how liability is distributed and how damages are valuated among actors that incur damages and decide to hold the APO responsible. It is easy to get the water muddled in such cases

Another issue is that a simple analysis of the market indicates heavy malware presence. Would and insurance company be willing to commit in an ecosystem where so much malware is present?

## COMPETITORS

Before a strategy can be formulated it is always good to look at other APOs'. Though Apple and Google are not direct competitors, they do own two most known app markets.
Checks done by Google for the Android Play Store are very limited. An automated malware scan is the only thing between a developer/attacker and the market. Typically an app will be available for download within 30 minutes of submission. Flagged, rising or suspicious apps are checked only afterwards. Current estimates are that 0.1% of the apps in the Android Play Store had malicious content at one time or another compared to the 8% of Baidu (Kelly G. 2014). These numbers clash with the Virustotal results, but this can be explained due how virus total works and what the author classifies as malware. In any case it is clear that Baidu in this case has 80 times more malware present.
Apple has an even more rigorous. Submitted apps are scanned for malware, afterwards the code is reviewed by hand. The review criteria can be found here: https://developer.apple.com/app-store/review/guidelines/
They are extensive and cover the subjects of Safety, Legal, Design, Business and Performance. It is only after this review an app is allowed to enter the market. Thanks to the stringent requirements and review process malicious content is not a permanent issue of debate when it comes to the iTunes app store. An added benefit of emphasizing security is that IOS was historically affected by less than 1% of all cyber threats in contrast to 97% of Android (Kelly G., 2014). This comparison may not be fair of course since user base of Android is much higher and the sheer volume of apps for the android platform

## UNCONVENTIONAL

The previously summed solutions are a part of standard text book risk management line of thinking. Unconventional solutions have the potential to change the face of the app market landscape. Looking at the problem two points are clear: Too many app submissions to extensively review by hand (too costly), no technology to intelligently review code. The latter is a technological issue and can't be resolved until intelligent IA is developed. The former point is just an issue of human resources and thus it is possible to address this problem. We can compare this problem to law enforcement. The resources to locate a person of interest is too costly without prioritization so law enforcement issues bounties to people who aide in this process. A similar tactic is already present for the Android platform, namely the Android Security Rewards. Any individual can participate in identifying vulnerabilities to the android system (running on a selected hardware). Depending on severity a reward is doled out. The same strategy can also be employed to identifying malicious content in apps. Though there are a few obstacles for its effectiveness. First of all if the app is not open source it becomes more difficult to analyze, secondly the sheer amount and severity of malicious content might dilute the bounty too much and thirdly an attacker might try to abuse such a system by identifying self-created malware.

## STRATEGY

It is important to understand that if at any stage security was circumvented that it offers no protection on later stages. It is therefore imperative to adopt a risk strategy that minimizes risk at every stage thus a mixed risk strategy is preferred. But to what extent this strategy should take form is dependent on a number of factors. The problem owner must first of all know what level of security it wants to adopt, then it should look at its financial means to see if it is possible to realize. Since both markets are free app markets, receive their profits from advertising and considering their unique position in the Chinese market they can get away with much more. They are much more served with the cheapest solution, Simple developer sign up, automated malware scan and short sanboxing period and only react to large amount of responses.

## Additional actor who influence the security issues

The app-market owners aren't the only actors in this game. Governments, consumers and app developers are all a part of this ecosystem.
In this case the Chinese government can be a very strong actor. The Chinese government has been known with its near totalitarian governance. It too can have a major impact on the malware density of the market by supporting regulations that prohibit the programming of malware in commercial apps and make it mandatory that each app-market adopts an extensive risk strategy.
Consumers control their own actions, their choices are instrumental to the malware creators. The consumer can be educated or educate himself. By becoming a much more aware consumer it is possible to prevent infection and know how to mitigate the damage caused by accidental infection. The caveat is that it is very hard to discern between legitimate apps and shady ones. The consumer is actually the weakest actor in this ecosystem because they are the least skilled with the least of means. On the other hand the consumer can become the strongest if it can be empowered by the APOs in novel ways such as bounty schemes.
App developers are also an actor within this ecosystem. Legitimate developers suffer from a compromised market. The only thing that can be done in an organized fashion are developer collectives. These collectives review the code of their peers and as a collective guarantee malware free apps. These collectives should also be recognizable by consumers. This way they reduce the load on a market owner for prevention and detection but also offer consumers a promise of clean apps. With the app-market owners' help, apps from these collectives can be ranked higher. All of this helps dev-collectives to gain a competitive advantage and slowly edge attackers out of the market, but his is at the risk of forming app cartels.

## Actors' risk strategies

As it has been said above, there are also many other actors in our problem scope in addition to the market owner. Not surprisingly, it is highly likely for each of these actors to implement different security strategies. This might be the case because these actors are also likely to belong to a different category, which means they are likely to have different problems and objectives. Next, the strategies used by the various actors and how they have developed over time are going to be described.

### Attackers

The first actor that obviously affect the app marketplace security problem are the attackers. From the previous submission, it can be concluded that the attackers have quite successfully infected the app marketplace by looking from the infected apps statistic. The previous report also showed that the most common attack strategy is based on malware. However, the lack of time dimension in the data means that it will be hard to establish how the attacker strategies have changed over time. Nevertheless, since it is already determined that the attacker methods and strategies are always a response to the defender's implemented controls, it can be predicted that their methods and strategies will only be more sophisticated in the future. Therefore, it is almost certain that the advance of strategies and method by the attackers will always result in the increase of risk.

### Regulatory Authorities

The second actor that highly influence the problem at stake is the regulatory authority. Since the app marketplace in the previous report based in China, then the relevant regulatory authorities are the Cyberspace Administration of China (CAC) and China's Ministry of Industry and Information Technology (MIIT). Although there is no clear conclusion about where the regulatory authority such as MIIT and CAC belong to in cyber security key player category, in this report it is determined that they are most closely resemble the security consumers.
There has been information about the activity of these two government entities. In May 2016, the MIIT released a list of 29 malicious apps after inspecting 43 app stores, four of which are from Baidu app store (Newsdoug, 2016). This might be a good sign that the government agency has been implementing risk reduction strategy to reduce harm suffered by customers. In June 2016, the CAC ordered various app marketplace to record users' activity for up to 60 days (Ramli, 2016). Although can be classified as risk reduction strategy, this action was arguably intended to support their "business strategy" instead of increasing real security level. Nonetheless, the risk reduction of regulatory authority through the release of regulations most likely have reduced the existing risk.

### App Developers

The third group of actors is the app developers. This actor group belongs to the security provider category. The app developers have been long known for their "ship today, fix tomorrow" philosophy, which has been a huge threat to security and privacy. The characteristics of information goods, such as externalities, security as sunk cost, and economics of scale have driven the developers to this method and somehow justify this insecure app development approach. By practicing this approach means that the app developers have performed risk acceptance strategy; they consider that the risk from lack of security is not economical to mitigate.
Unfortunately, even these days it seems that this approach is still practiced by many app developers. One report says that large companies spend around USD 34 million each year, but only 5.5 percent is allocated for security (Ramel, 2016). This fact highlights that this "ship today, fix tomorrow," or also called "rush to release" phenomena, still exist and applied by many app developers. However, this practice is not the only one to blame. It has been reported that some companies also suffer from lack of qualified developer (BI Intelligence;, 2016), which hinder their effort in realizing more secure mobile apps. Even though there has to be a risk mitigation strategy done by at least some app developers, the more common risk acceptance strategy arguably dominates this group of actors. Looking at these facts above, it is really difficult to say that the app developer risk strategy has gone towards a better way.

### Consumers

The fourth group of actors is the consumers. Consumers are those actors that download and use apps from the app marketplace. Since in the cyber security key player categories the security consumers refer to IT industry customers, the consumer category, in this case, may not be considered to be included in the cyber security key player category.
The most reasonable risk strategies that may be employed by these consumers could be either risk reduction or risk avoidance. There is no substantial evidence that these strategies have changed over time, although it is highly likely that the application of these strategies might have been increasingly more effective and efficient along with the increasing awareness of cyber security risk. An increase in risk mitigation strategy is, for example, there are fewer consumers who use the same password for several accounts. A different example of risk avoidance strategy might be in the form of consumer avoidance of downloading unverified applications. Even though these increasing awareness surely leads to a reduction of risks, but there is always a possibility that the attackers might also adapt to this change. Therefore, it is difficult to conclude whether the risk suffered by the consumers have decreased or not.

## Return on Security Investment (ROSI)

The strategies companies might pursuit to deal with risk are generally grouped into two families. There exist in fact both ex-ante mechanisms, aimed at the prediction and prevention of incidents, and ex-post facto strategies, to detect it and respond to it.

Despite in an ideal (NB ideal with risk, we mean that there is uncertainty about event, but without constraints on resources) situation the risk owner can (and should) distribute his resources along those four dimensions (prediction, prevention, detection and response (Rosenquist, M. (2009)), in reality this is not always possible.

We think this discrepancy might be induced by four causes.
First, the type of risk influences the measure: some risks are easier to forecast, other easier to prevent or easier to react to.
Secondly, there are some variables that the decision maker is not able to observe/control.
Moreover, in reality, businesses have constraints about their resources, concerning mainly the available budget and time.
Lastly, depending on the type of business, on the organizational culture, size, "age" and similar external factors, some strategies are more attractive than other.

In the real world, therefore, companies face the problem of assessing many possible strategies, operating with different objectives in different times. Fortunately, in order to deal with this puzzle, they can use some of the tools exploited in the investment evaluation, like the ROI (Return on Investment) or the NPV (Net Present Value). These two quantitative assessment methods support decision makers formalizing the material impact in terms of money. Our analysis will show how these methods might, however, lead to imprecise conclusions as they rely upon a money conversion that is not always possible.

Our assignment turns around three principal actors: the market owner, the app developer and the consumer.
The first problem in estimating the return on a risk mitigation strategy in our environment deals with the distribution of the outcomes on these actors.
In the case of malware, we argue that the consumer will have to pay the higher cost. Analysing Google PlayStore (Google, 2016), it has come out that most of the malicious code enables the collection of unauthorised data. While this can result in tremendous losses for companies whose employees have compromised devices (Snyder, 2016), it is extremely hard to estimate how much such a malware costs to Google or to the app developer. Their loss, in fact, will burden in terms of brand image and customer loyalty.

To overcome these difficulties we have decided to step back and analyse the root of the problem. In fact, malicious code takes advantage of Android's bugs and vulnerabilities to bypass security. It is thus reasonable to assume that the cost Google faces for a malware, is actually the cost of the bug that the malware exploits.
Hence, the problem of keeping malware out of the store can now be reformulated, in a slightly preventive fashion, in terms of developing bug-free applications.

Developers worldwide have produced a lot of guidelines and best practices for what has even been called "The Art of Debugging" (Matloff et al., 2008).

Whittaker et al (2012) estimates that 30/50% of developers time is dedicated to fix bugs. If you consider bugs hunting as an integral part of programming, then these numbers are acceptable. However, it might be argued that, though closely related, the two requires different skills, and might be performed by different individuals. From this perspective, 30/50% of the development team time would represent a significant drop in productivity.
The infographics of (Hughes et al., 2012) and (TestObject, 2014) provide a detailed overview of the economic environment of mobile applications. The experts claim that testing and debugging for a serious application range from $30000 to $150000, and represent 40% of the total expenses. It is a pretty large interval, but is at least a good starting point for our analysis.
As a matter of fact, it sounds logical assuming that the result of such an intensive test carried out by professionals will be a bug-free software, and may be considered as an insurance against future malware.

Nonetheless, it is spontaneous to realize the impossibility, even for a colossus like Google, to test every application on the PlayStore.
Despite source code reviews are however routine procedure, the second path to code testing is using automated tested. As the research towards AI progress, computers might represent a powerful resource for debugging. Baidu has recently invested in this technology to identify malware (Metz, 2016).

While Baidu seeks a solution exploring futuristic technologies, the Californian headquartered giant in June 2015 adopted a strategy already consolidated in the traditional fields: Android joined Google's Vulnerability Rewards Program (Google Online Security Blog, 2016).
Projects that reward researchers who spot and submit bugs are not a recent innovation: Microsoft, Mozilla and Google itselft already have taken advantage of them for a log time. The Android

Vulnerability Rewards Program, as the name fancifully suggest, offers monetary rewards according the following table:

| Severity | Bug Report* + Proof of concept + CTS + patch | Bug Report* + Proof of concept + (CTS or patch) | Bug Report* + Proof of concept |
|---|---|---|---|
| Critical | $8,000 | $6,000 | $4,000 |
| High | $4,000 | $3,000 | $2,000 |
| Moderate | $2,000 | $1,500 | $1,000 |
| Low | $1,000 | $500 | $333 |

The severity of the vulnerability is well defined within Android ecosystem, a complete description can be found in (Google, 2016)

This strategy offers different advantages:
1) payment of the vulnerability: no expenses for test that might reveal only small bugs;
2) it limits the deployment of resources: a lightweight team might be enough;
3) prompt to a continuous check and monitoring;
4) it involves consumers, bracing the loyalty and increasing awareness about security issues;

And so? Have we finally derived a good estimator of how much a malware costs to Google?
Well, our answer is that it might be a fair value, but some considerations must be accounted:
i) the definition of vulnerability as well as the level of severity, how Google reminds (Google, 2016), changes every year, making comparisons over time impossible.
ii) that number is the price big G "sells" vulnerability: as such, we can assume it is partially illustrative of the priority Google estimate the bug (and thus the actual cost that bug represent to the company). On the flip side of the coin, as a price, it must be subject to some classical economic constraint.

Briefly, a last interesting aspect of Android Vulnerability Rewards Program is that it prevent fresh vulnerabilities to be sold on black markets. As a matter of facts, the high-speed growth of the cyberspace resulted also in new frontiers of illegal entrepreneurs (Greenberg, 2016).
As it is clearly explained in (Paganini, 2013), there is an increasing market for the zero-day vulnerability exploit, critical bugs that, if sold to a malicious hacker, might become malware.
With this program, Google attempts to tackle this problem.
However, in order to get a real estimator of Google's "cost of malware", it would be important to integrate the value of Android's bug on these platforms. Nevertheless, very few data has been found.

One year later after the introduction of the program, in [https://security.googleblog.com/2016/06/one-year-of-android-security-rewards.html] Google proudly show some of the program results:
- over 250 vulnerabilities have been reported;
- $550000 have been paid to 82 researchers;
which leads to similar average to the cost of a high severity vulnerability found before.

In (Google, 2016) it is stated that the program contributed to spot 100 bugs more than the previous year.

We assume that Google's policy has not changed, that is, Android Rewards Program has not replaced other debugging procedure.

Finally, assuming that the number of new vulnerability is constant over years, we can now compute the return on security investment as follows:

Risk exposure: 250 vulnerabilities to an average of $2200 each is $550000;

Risk mitigation: 100 vulnerabilities disclosed on the average of 250/year is 40%;

Cost of the strategy $550000;

ROSI: ($550000x40%-$550000)/$550000=220000-550000/550000= -0.6

## References

BI Intelligence;. (2016, September 8). App developers are rushing to release — and it could be causing security problems. Retrieved Oct 15, 2016, from Business Insider: http://www.businessinsider.com/app-developers-are-rushing-to-release-and-it-could-be-causing-security-problems-2016-9?international=true&r=US&IR=T

Google. (2015). A (Ramli, 2016)ndroid Security: 2015 Year in Review, (April), 1–43. Retrieved from https://source.android.com/devices/tech/security/reports/Google_Android_Security_2014_Report_Final.pdf\nhttp://googleonlinesecurity.blogspot.com/2015/04/android-security-state-of-union-2014.html

Google 2016, Android Security, 2015 Year in Review http://static.googleusercontent.com/media/source.android.com/it//security/reports/Google_Android_Security_2015_Report_Final.pdf

Google Online Security Blog. (2016). Announcing Security Rewards for Android. [online] Available at: https://security.googleblog.com/2015/06/announcing-security-rewards-for-android.html [Accessed 17 Oct. 2016].

Greenberg, A. (2016). Here's a Spy Firm's Price List for Secret Hacker Techniques. [online] WIRED. Available at: https://www.wired.com/2015/11/heres-a-spy-firms-price-list-for-secret-hacker-techniques/ [Accessed 17 Oct. 2016].

Hughes, C., Souz, R. and Wagner, S. (2012). [Infographic] How Much Does it Cost to Make an App?. [online] Idea to Appster. Available at: http://www.bluelabellabs.com/ideatoappster/how-much-does-it-cost-to-make-an-app-an-infographic/ [Accessed 17 Oct. 2016].

Kelly, G. (2014, March 24). Report: 97% Of Mobile Malware Is On Android. This Is The Easy Way You Stay Safe. Retrieved October 14, 2016, from http://www.forbes.com/sites/gordonkelly/2014/03/24/report-97-of-mobile-malware-is-on-android-this-is-the-easy-way-you-stay-safe/#290038707d53

Matloff, N. S., & Salzman, P. J. (2008). The art of debugging with GDB, DDD, and Eclipse. No Starch Press.

Metz, C. (2016). Baidu, the 'Chinese Google,' Is Teaching AI to Spot Malware. [online] WIRED. Available at: https://www.wired.com/2015/11/baidu-the-chinese-google-is-teaching-ai-to-spot-malware/ [Accessed 17 Oct. 2016].

Newsdoug. (2016, May 30). INTERNET: Regulator Rebukes Baidu for Malicious Apps. Retrieved Oct 15, 2016, from Young's China Business Blog: http://www.youngchinabiz.com/en/internet-regulator-criticizes-baidu-mobile-assistant/

Pierluigi Paganini (2013). Zero-day vulnerability exploits, too precious commodities. [online] Security Affairs. Available at: http://securityaffairs.co/wordpress/20275/cyber-crime/nss-labs-zero-day-exploits.html [Accessed 17 Oct. 2016].

Ramel, D. (2016, August 23). Security Study: Developer 'Rush To Release' Increases App Risk. Retrieved Oct 15, 2016, from ADT Mag: https://adtmag.com/articles/2016/08/23/app-level-security.aspx

Ramli, D. (2016, June 28). Apple to Face More Scrutiny as China Cracks Down on Apps. Retrieved Oct 15, 2016, from Bloomberg Markets: http://www.bloomberg.com/news/articles/2016-06-28/china-orders-mobile-app-stores-providers-to-monitor-users

Rosenquist, M. (2009). Prioritizing information security risks with threat agent risk assessment. Intel Corporation White Paper.

Snyder, B. (2016). For Android, adware is the threat, not malware. [online] InfoWorld. Available at: http://www.infoworld.com/article/2910323/mobile-security/for-android-adware-is-the-threat-not-malware.html [Accessed 17 Oct. 2016].

TestObject. (2014). How Much Does Mobile App Testing Cost? – TestObject. [online] Available at: https://testobject.com/blog/2014/03/how-much-does-mobile-app-testing-cost.html [Accessed 17 Oct. 2016].

Whittaker, J. A., Arbon, J., & Carollo, J. (2012). How Google tests software. Addison-Wesley.
ISO 690