

# Think like a hacker! – Mobile application threats

Paweł Kuryłowicz

AGH University of Science and Technology, Krakow, Poland  
pawel.kurylowicz95@gmail.com

**Abstract:** Cybercrime targeting mobile applications is a reality, and while many consider it the biggest issue in the mobile security world, it is definitely not the only one. Many app developers are focused on profits and because of that, it is common that things are done quickly, to save time, which means, in turn, to save money. We have to try to find a balance between the rush to market and reflecting on the security of our applications. Usually at work, managers expect developers to produce code and that is completely understandable, but many of them are required to do that quickly and hence, not securely. Developers without specific knowledge of secure coding and development practices are unlikely to even know the scope of this problem. They do not realize what can happen to their applications, and hence the users of those apps, through various common vulnerabilities. Cybercrime scenarios are becoming more and more sophisticated, which is why I want to share my knowledge about some of the ways that hackers can mess with mobile applications. I hope that this point of view will help you understand how cyber criminals think, because if we realize that, then we can start defending ourselves by securing these points of attack. As an example, I want to show a real crime scenario with Tinder - an application which in fact allows us to track people live and that can lead to stalking, robbery or even help in murder.

**Keywords:** Application, Security, Vulnerability, Threats, Awareness, Penetration test, Methodology, Malware, Tinder, Facebook

## 1 Three types of hackers

There are three types of hackers. Those without any knowledge, those who know some helpful tools and those who know exactly what they are doing. In the first group are people who sometimes are not even interested in IT Security. They're just clicking random buttons or filling fields with various values to see what will happen next. Of course, usually nothing happens, but once in a blue moon they surprise themselves because something goes wrong and now they became the lucky owners of a new notebook without paying. Those in the second group are smarter but not that smart. They are commonly referred to as "Script Kiddies", because they prefer tools instead of thinking. The only thing they need are instructions that tell them how to launch the pre-packaged tools made by others: "Please enter an IP address and press the START button to hack this person". In the third group we find the

most interesting people, including some sophisticated hackers. They have deep knowledge and understanding of IT Security. They know exactly how to find vulnerabilities, how to prepare a targeted attack or what would be the greatest risk to an application. Today we will try to think like one of these hackers, to see how they act and where they are messing with mobile applications.

## **2 Motivation**

But first, we have to think what motivates us as a hacker? Why we are trying to break someone's application? Most people assume it is for money, but there are a bunch of people who are doing this just for satisfaction, getting information, obtaining some kind of influence or even just for fun. If hacking is your hobby, then it really doesn't matter how long it will take to find a vulnerability or how difficult it will be.

## **3 Information gathering**

If we know why we are doing this then let's start doing it. How will we start? We can start by finding an application of interest either to us or to a possible future client. We have to get to know the application and collect as much information as we can. How does it work? What functions does it have? And the most important thing, what are the risks and profitable scenarios for us? Different applications will have different potential threats. For example, in a banking application the primary concern would be theft of funds whereas with let's say, Facebook, the main concern would be theft of data.

## **4 Static analysis**

Now we have completed the first step and we know what the potential risks are, we can move to static analysis. As we're focusing on Android apps here, this starts with obtaining an APK package. The APKs of applications available from the Google Play store can be found at [www.apkpure.com](http://www.apkpure.com). Alternatively, they can be copied off a device where the app is installed, using the Android debugging interface, with the command `"adb pull /data/app/[NameOfApplication]"`. Once we have the APK package, we need to decode and unpack it. For this step, there is a helpful utility called "Apktool", which will do all this for us. Next, we will convert the .dex files into .jar files with the "dex2jar" tool, resulting in Java archives. Finally, thanks to "JD GUI", we will extract the Java source code. Reading through the app's source will help us understand the application's functionality, perhaps identify implementation errors, or find helpful comments left by the developers. Sometimes we can even find tokens, keys or passwords.

## **5 Dynamic analysis**

After static analysis, we move to dynamic analysis. In this step we are checking what is stored inside application logs, how the app communicates with any servers, and how it interacts with other active processes. By setting up a proxy we are able to see all network traffic between the mobile application and its servers. But it is not always that easy. Sometimes there are obstacles like root detection or certificate pinning which will not allow us to connect through a proxy. But as a sophisticated hacker we can also try to bypass these. A smart thing to do here would be to launch the "adb logcat" tool, to see all logs from the device. It can be very useful to detect the cause of a problem if an application won't cooperate with you.

## 5.1 Root detection

Let's move to root detection. If we have this problem, then it means that the application is looking for specific packages or files, or it's checking directory permissions. The easiest and laziest method to bypass this kind of protection is to download an application like "Magisk" and just simply hide the root at the click of a button. We can also take a closer look into the source code to find the function that checks if the device is rooted and delete or modify it. Here, a really helpful program can be "Apktool" which will decompile .dex files into .smali files. Right here we can make our changes. After compilation of our modified file, we have to remember to sign it. "d2j-apk-sign" will do this for us.

## 5.2 Certificate pinning

Another popular security mechanism is certificate pinning. It's a good and widely used protection. The application doesn't have to rely on the device's trust store, because the certificate is hard-coded into the app during development. Anyway, there are a few methods to bypass this mechanism too. We can use "Apktool" again, to find and modify the function responsible for pinning, just like we did before with the root detection. Another option here would be to change the protocol from HTTPS to HTTP. If that works we can avoid SSL and the whole problem of pinning.

## 6 Dynamic analysis

When we finally set the proxy, the real fun begins with dynamic analysis. At this stage we are checking what is stored inside the application logs and how the app communicates with its servers, or how it interacts with other active processes. We are looking at requests through different functions and we are trying to modify them in ways the developer may not have planned having to handle, producing unexpected results. This allows us to verify our scenarios from the information gathering stage.

## 7 Tinder as a real example

Let's do this with a real application.

Let's assume that, as sophisticated hackers, we sometimes feel lonely in our binary world. Fortunately, technology has found a cure even for that!

Over 50 million people use Tinder. Its Android mobile application is dedicated to helping singles find a partner or a friend. There are three basic functionalities "Like", "Dislike" and "Super Like". After matching we are able to chat with that person, which can turn out to be a future partner or even, a future wife! Through Tinder, by default, we are sharing information such as our school, job and age from our Facebook profiles, and the distance between us and our future wife.

What potential risks can be lurking here? Stealing the user's data from their Facebook profile, making purchases in the application without paying, and many more.

But today we will try to focus on tracking people through this application. Is it even possible? Normally it should be safe, because even if we have information about the distance, we don't know where our future wife is, exactly, without a specific direction. Let's try to find if there is a way to get around that. As almost always we will have to bypass a couple of security mechanisms such as those already mentioned above. This time it is certificate pinning, so we have to check the .smali files to find our obstacle. After searching for words such as "cert", "certificate", "pinning", "pinner", etc, we will finally find the "CertificatePinner" file, in which we can find the method responsible for that. To bypass this, we can simply delete the whole method. After that, when we have compiled and signed the application

with certificate pinning removed, we can start using a proxy. If we look closer at the communication between the server and the mobile application we will find an HTTPS request which sends our geographical coordinates:

**Request HTTPS:**

```
POST /user/ping HTTP/1.1
platform: android
User-Agent: Tinder Android Version #####
os-version: 23
Accept-Language: en
app-version: ####
Content-Type: application/json; charset=utf-8
Content-Length: 35
Host: api.gotinder.com
Connection: close
X-Auth-Token: #####

{"lat":50.0389679,"lon":19.9420084}
```

**Response HTTPS:**

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Fri, 28 Apr 2017 15:33:59 GMT
X-Request-ID: #####
Content-Length: 14
Connection: Close

{"status":200}
```

Following another request, below, the server creates a response with the information on how far away our future wife is:

**Request HTTPS:**

```
GET /user/5543#####f HTTP/1.1
platform: android
User-Agent: Tinder Android Version #####
os-version: 23
Accept-Language: en
app-version: ####
Host: api.gotinder.com
Connection: close
If-None-Match: "#####"
X-Auth-Token: #####
```

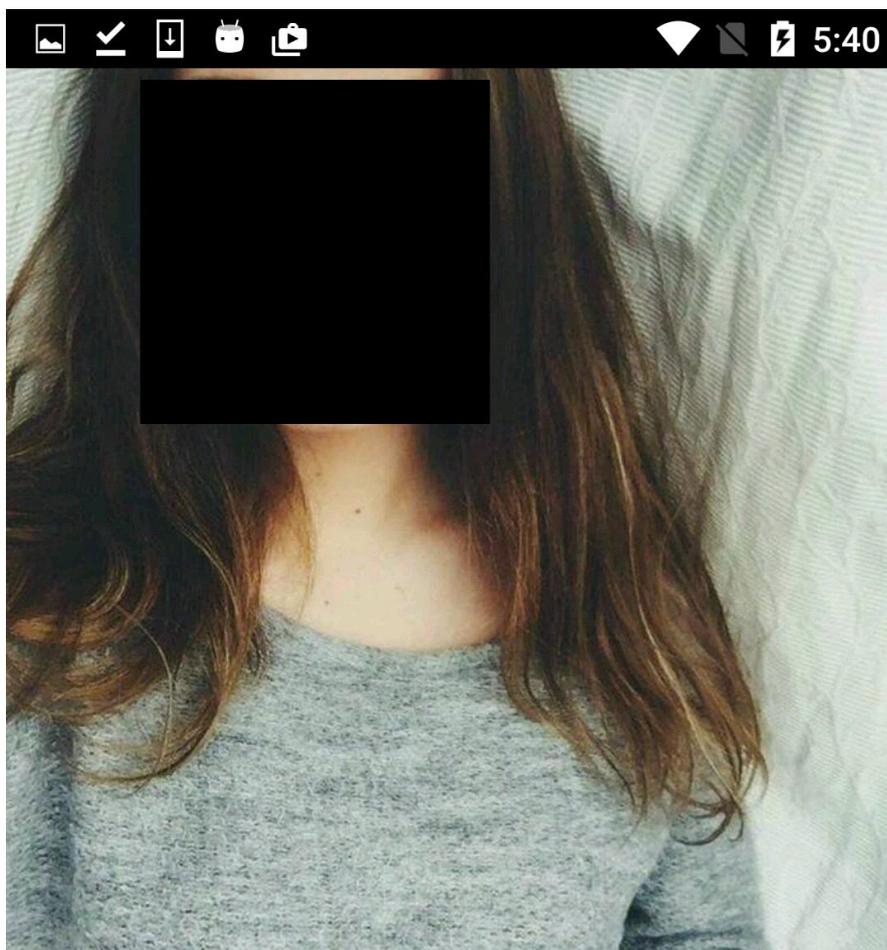
**Response HTTPS:**

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Fri, 28 Apr 2017 15:34:08 GMT
ETag: "#####"
Vary: Accept-Encoding
Vary: Accept-Encoding
X-Request-ID: #####
```

Content-Length: 12955  
Connection: Close

```
{"status":200,"results":  
(...  
,"distance_mi":7}}
```

What can we do with that? As a hacker, we can modify the first request and then we will know how far our future wife is from some other location. What will this give us? The distance between a new location and our future wife. That is still not enough to find out where she is. But if we change our location one more time, then we've hit our jackpot. Why? Because we can create three circles with our future wife's distance as a radius from those places and, applying some simple high school math, we know that these circles intersect exactly at one point, our future wife's location. To show you precisely how it works, we will use Paula, one of matched girls as an example. This is Paula:



, 18  
📍 7 miles away

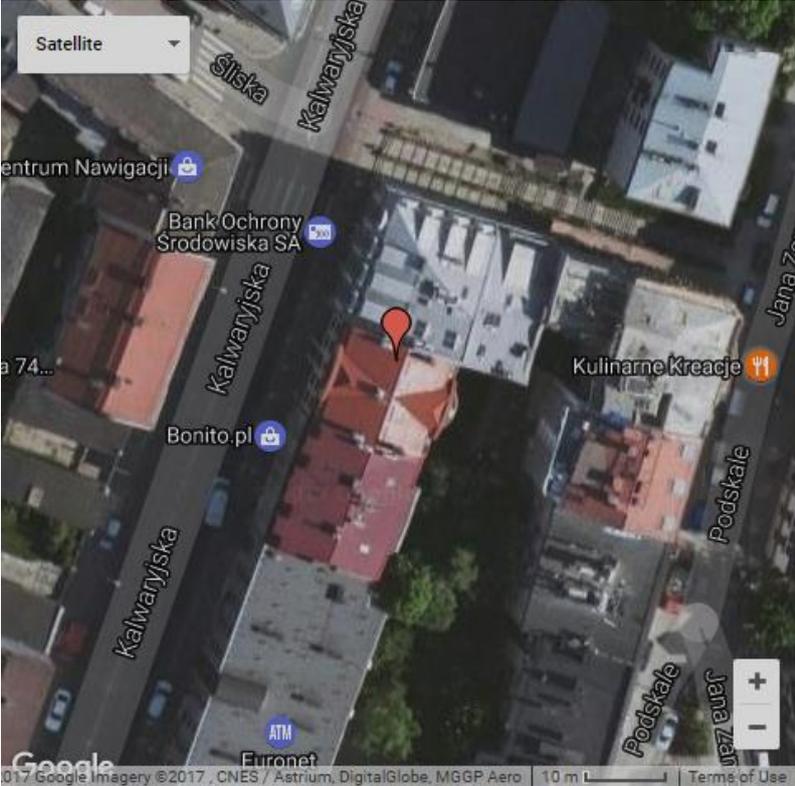


---

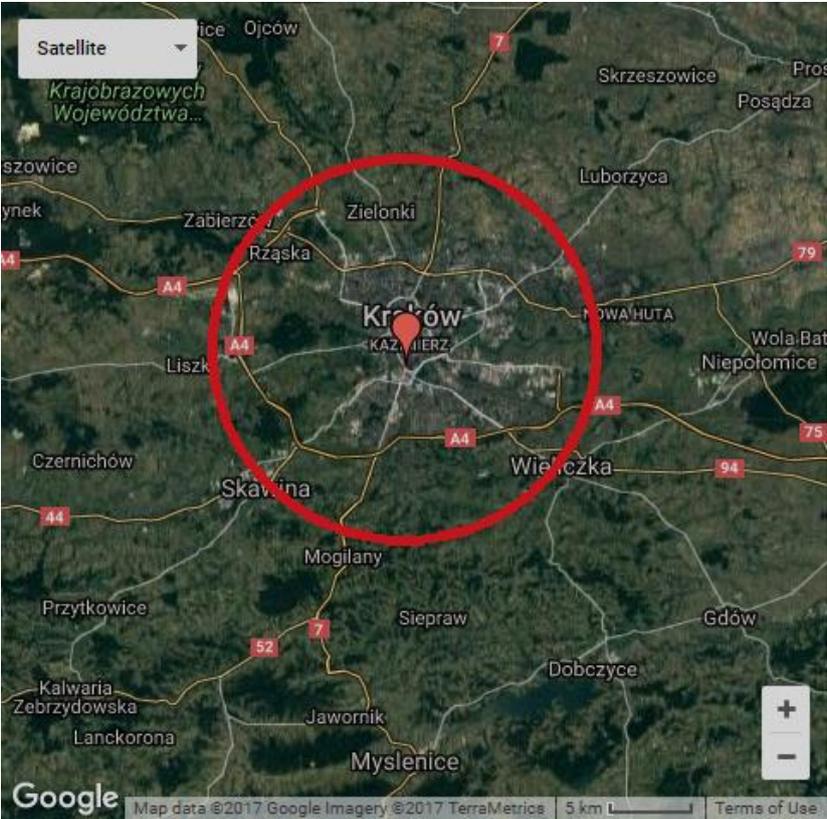
Szukam męża

---

Kalwaryjska 65, this is the place where I work.



Paula is 7 miles away from my office, but we don't know in which direction. So, she can be anywhere on this circle.

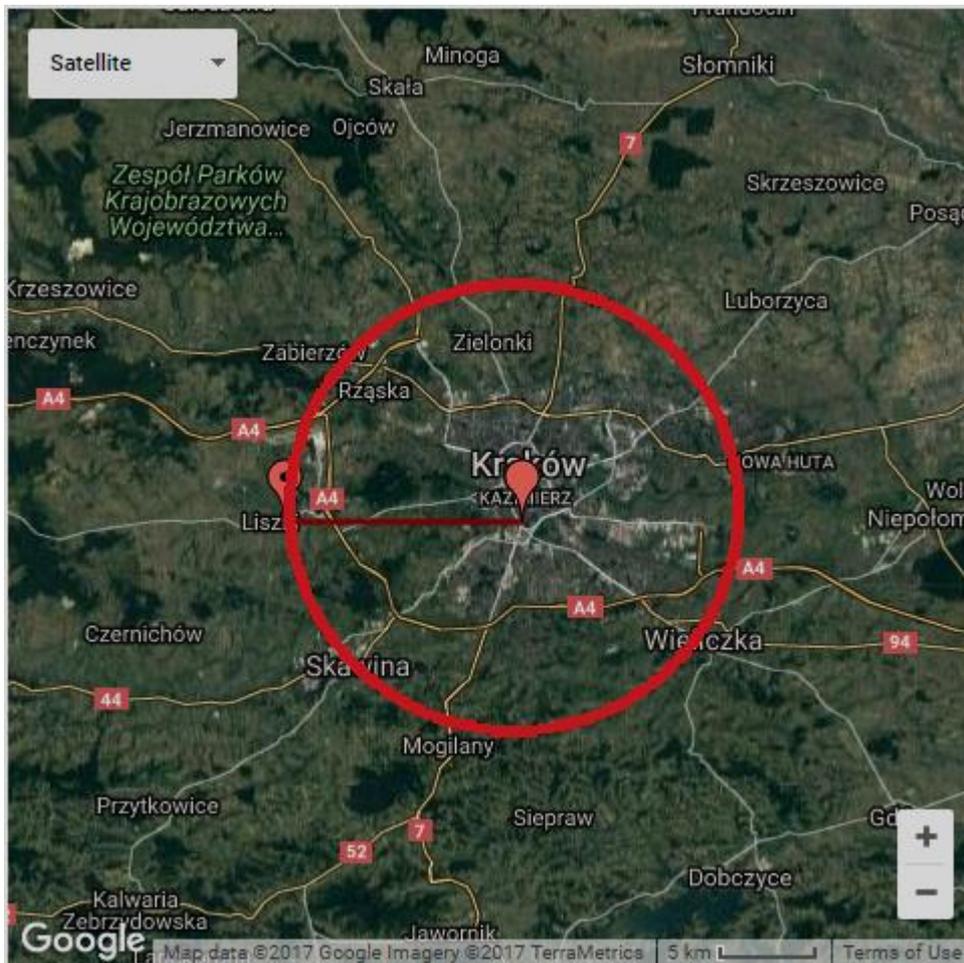


Let's try to change our location to the left of the circle. To do that, we have to send a modified request with our location about 7 miles to the west of the office. We need to change the location – “{“lat”:50.0389679,“lon”:19.9420084}” – in the initial request body and replace it with a new one.

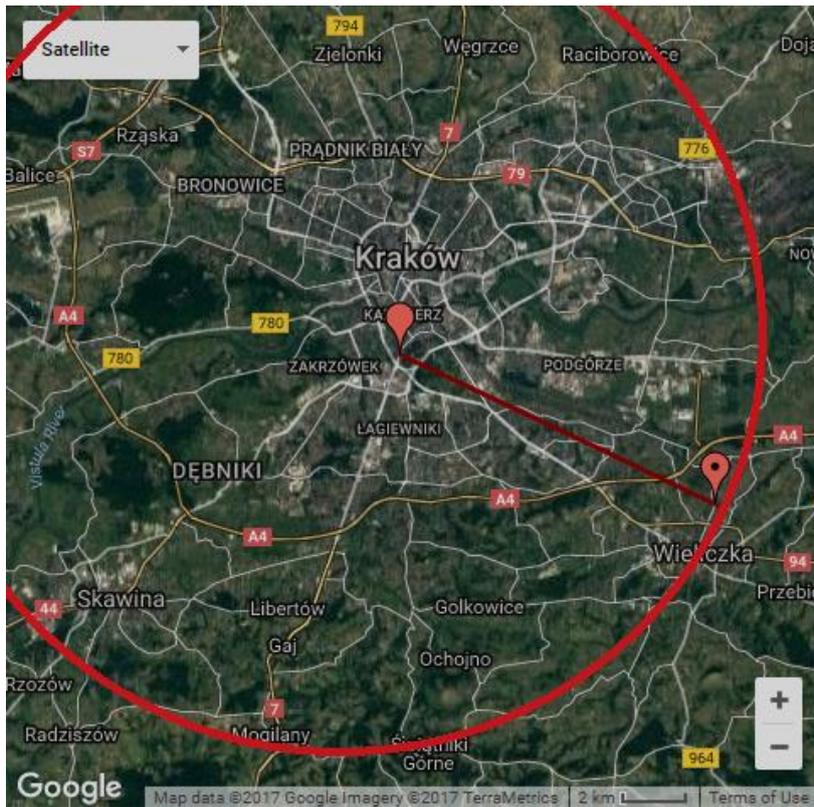
### Modified request HTTPS:

```
POST /user/ping HTTP/1.1
platform: android
User-Agent: Tinder Android Version #####
os-version: 23
Accept-Language: en
app-version: ####
Content-Type: application/json; charset=utf-8
Content-Length: 35
Host: api.gotinder.com
Connection: close
X-Auth-Token: #####

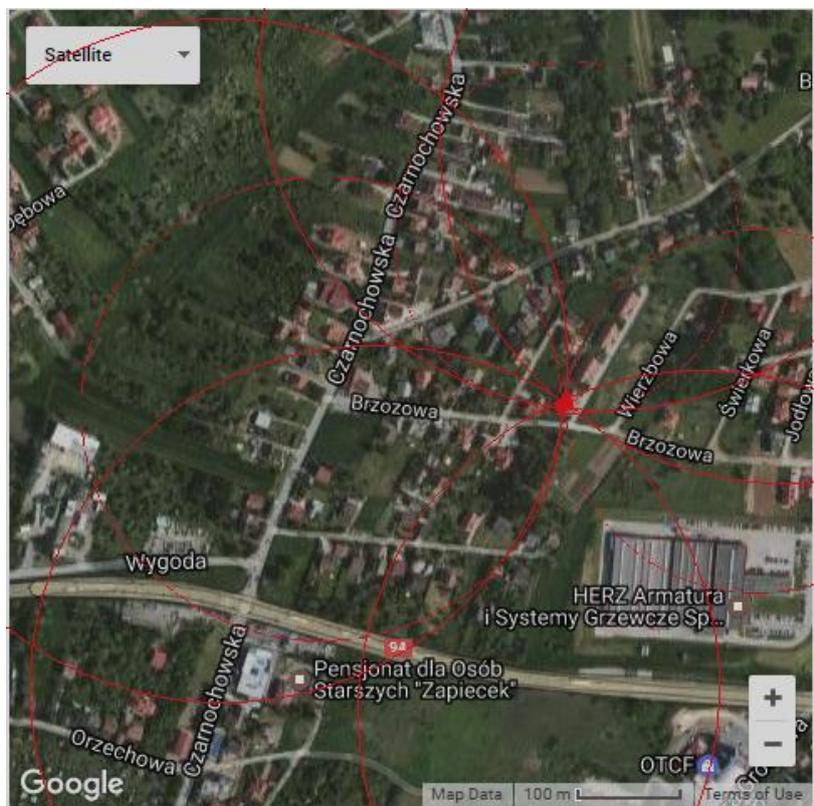
{"lat":50.0389679,"lon":19.7800000}
```



From this point, the distance between us and our future wife increased to almost twice the initial response, which simply means that this is the wrong direction. We won't give up; after a few more requests we are able to find which way we should go.



If we keep sending such requests from various locations, then finally we will find our future wife.



This location is used just as an example, it's not a real address.

## 8 Worse scenario of the same issue

It seems great as long as we are a hacker, who just fell in love. But we don't have to stop here. We can continuously send those modified requests. Or write a script which will allow us to track her live. We could become a stalker or we can also rob her house stress-free, because we will know exactly, how far away from home she is. And what if this hacker was a pedophile or a murderer?

## 9 Final thought

Tinder doesn't look like a tool for criminals. It's just a great application. But every unsecured application can become a weapon in the hands of malicious hackers. Fortunately, it's not as bad as it looks. Standing opposite the cybercriminals that are called "black hats" there are the "white hats", ethical hackers with great responsibility for keeping all of us safe. They are doing almost the same things but in good faith. They are not just finding vulnerabilities, they are also finding solutions for how to fix them. Testing an application once a year will help, but will not provide enough security. Not every company realizes this. That is why we have to think like hackers to beat the hackers.

### References:

1. Florian Stahl, Johannes Ströher.: Security Testing Guidelines for mobile Apps.  
[https://www.owasp.org/images/0/04/Security\\_Testing\\_Guidelines\\_for\\_mobile\\_Apps\\_-\\_Florian\\_Stahl%2BJohannes\\_Stroeher.pdf](https://www.owasp.org/images/0/04/Security_Testing_Guidelines_for_mobile_Apps_-_Florian_Stahl%2BJohannes_Stroeher.pdf)
2. OWASP Mobile Security Project. [https://www.owasp.org/index.php/OWASP\\_Mobile\\_Security\\_Project](https://www.owasp.org/index.php/OWASP_Mobile_Security_Project)
3. Łukasz Bobrek.: Testowanie bezpieczeństwa aplikacji dedykowanych na platformę Android.  
<https://www.slideshare.net/wojtwo/testowanie-bezpieczenstwa-aplikacji-dedykowanych-na-platform-android>
4. Sławomir Jasek.: Testowanie bezpieczeństwa aplikacji mobilnych.  
<https://www.slideshare.net/wojtwo/testowanie-bezpieczenstwa-aplikacji-mobilnych>