Saving You Money.

Tips for Better Mobile App Penetration Testing.

Just a little upfront planning for your mobile app penetration test can pay real dividends. Making sure that all the important parts of the app are reviewed for weaknesses.

A penetration test will assess how resilient your app is. How it is built and configured. How it communicates with its supporting networks and data stores. All should be included to guarantee a useful test.

Here are a few essential questions you should answer before starting:

1. Do you have a Native App?

These are written to be installed and to run on a mobile device. There are two main types based on the type of mobile device.

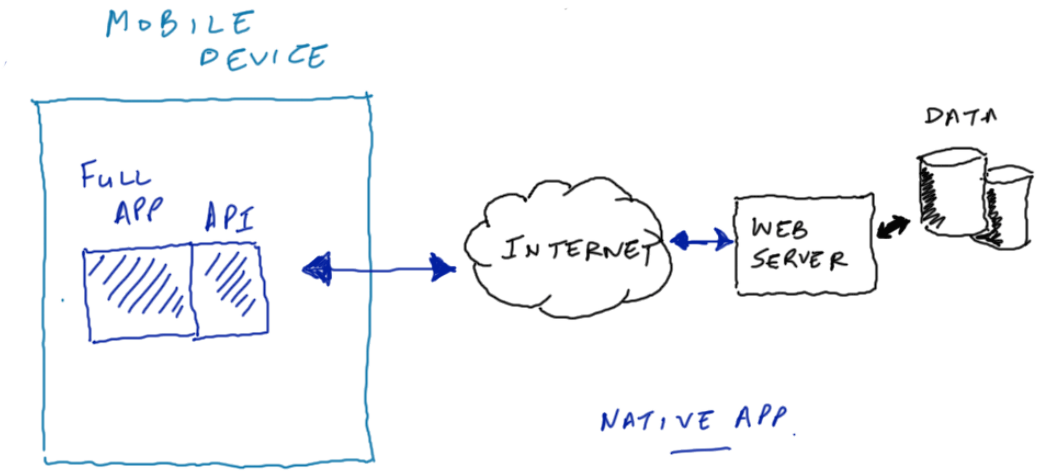
The first is iOS (Apple), where apps are normally written in Objective-C. The second is Android where Java is often the development language.

Usually, native apps need to be downloaded from an app store. If for private company use, they can often be found on a restricted company store.

Native apps will only run on the platform that they are developed for. This makes them more expensive to develop, as different versions need to be produced for each platform. However, they generally perform far better and give a much richer user experience than hybrid and web apps discussed below.

How the app is built is important for the penetration test. This is because testing will attempt to reverse-engineer a native app. This is a technique used to generate a blueprint of the app.

Why do this? To better understand how the app is working. How it is configured. What services it communicates with. How it can be exploited to behave in a manner that benefits an attacker. This supports the OWASP mobile app testing methodology.



2. Do you have a Hybrid App?

This is a native app that supports the running of a general web application.

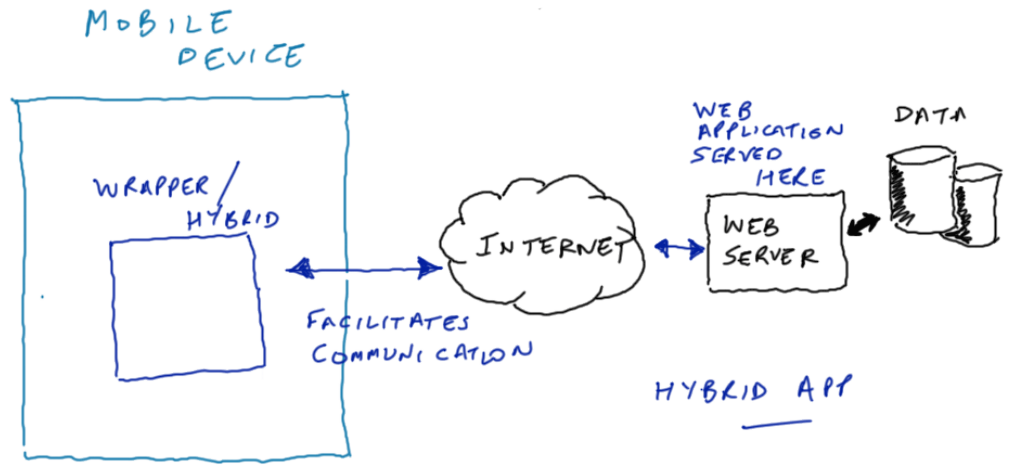
As above, it is downloaded and installed for a device. However, this app is designed to run a general web application that operates on all devices.

Why do this? Mainly to save development costs and money. The hybrid app is much simpler, so developing for different platforms is quicker and cheaper.

Once installed it facilitates and controls access to a web application that can be accessed by all platforms. Only one version of the web application needs to be developed and maintained. Hence the cost saving.

Hybrid apps interface with the device using 'plugins' to gain native behaviour through HTML & JavaScript. A hybrid app uses web technologies such as HTML and JavaScript but it is not a web application per se. It is rather like having a bunch of HTML files on the local hard disk that link to each other.

During the penetration test, the app is dissected to understand how it behaves. The same principles apply to native app testing above. The security assessment will include OWASP mobile app tests.



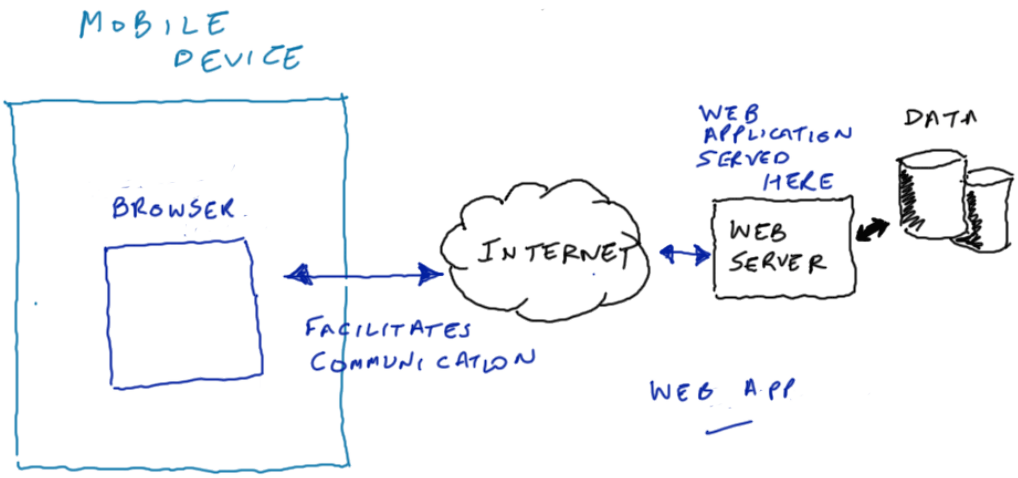
3. Do you have a Web App?

In terms of mobile app development, this is generally the least complicated.

A web application runs from the browser on the mobile device. Much as in the same way as on a PC.

Obviously, this is simpler and cheaper to develop. However, how the app operates depends on the resources available on the mobile device. It will not work offline or where there is limited Internet connectivity. There is no priority afforded to the app. It is simply another web resource accessed by the browser on the device.

Here, the penetration test will be a standard OWASP web application test. Bespoke mobile app components are likely not to exist as the target is accessed using a general Domain name such as http://www.testapptarget.com.



4. Does the App process or store data?

So far we have spoken about the front end of the app. Now, let's consider behind the scenes. The real prize for an attacker…

Data.

Some apps hold or access data held locally on the device. Users may give it permission to access their contacts for example.

Others communicate with remote networks to store, process and transmit confidential information. The interface to these resources is of interest and is often referred to as an Endpoint or Application Programming Interface (API).

Let us cut through all the jargon here. *This is simply an interface.* It is a call to a service that takes the data that is passed in. It processes the data in some way and returns a result.

Its function is to make sure that data that is passed in is presented in the right way so that it can be processed correctly.

Reverse engineering can often reveal these API calls for further manipulation during testing. Additionally, what can speed a test up is to tell the tester in advance how the API is called so that it can be tested directly. That is, testing by explicitly referencing it. *Not using the app.*

Why test the API directly?

It provides far greater flexibility when performing testing. It will also allow the tester to really assess how secure the API call is… *Is it validating input? Can it cope with odd or strange values? Is it using a default username or password?*

Sometimes these API calls lack resilience as they are written to only cope with controlled input that comes directly from app itself.

An attacker may be able to control input to the API. If this happens, all manner of damage can be inflicted if the API does not adequately check what is passed to it.

API or endpoint calls can be described using a standard such as WSDL (Web Service Definition Language) or Swagger. Alternatively, development specifications should contain a clear explanation about how to invoke the API and what arguments to provide.

If you are going to perform a penetration test on mobile apps, consider providing configuration details on both the front end (the app) and the back end (API and server calls).

This information will provide the tester with a more complete understanding of the architecture that will, in turn, lead to more comprehensive results.