Inspection of Windows Phone applications

Dmitry Evdokimov, ERPScan, d.evdokimov@erpscan.com

Andrey Chasovskikh, andrey.chasovskikh@gmail.com

BlackHat Abu Dhabi 2012

Abstract. Nowadays, the number of smartphone users is growing pretty fast. People use smartphones more intensively for personal and business needs. And so grows the number of mobile applications, from leisure applications like games to critical applications like mobile banking clients and business applications. Many of them operate sensitive user data, and that is why the topic of mobile applications security is very important.

Security auditors use special tools for application assessment. For desktop applications, these tools are countless. There are many tools for Android and iOS mobile platforms because they have been on the market for long enough. But there is a lack of such tools for the new Windows Phone platform.

The purpose of this paper is to describe approach to assessment of mobile applications, to overview existing tools for Windows Phone platform, and to present a new tool called Tangerine, which allows to automate analysis and to make it easier.

Keywords: mobile applications, application security, Windows Phone, .NET, instrumentation

1. Security assessment of mobile applications

Assessment of mobile applications usually includes examination of the following three parts:

- Backend server
- Communication channel
- Mobile application

This is typical client-server interaction. And while first two parts are common for all mobile platforms and do not involve any specific issues, the third part is more interesting from the mobile researcher's perspective.

Analysis of a mobile application includes the following steps:

1) Preparing environment
   On this step, the assessor prepares all the stuff which is required for analysis. This may include decrypting target application, configuring device emulator, or unlocking real device etc.

2) Static analysis
   Static analysis is used to assess applications without running them. This includes: compilation parameters analysis, metadata analysis, examining source code (original or decompiled).
3) Dynamic analysis
   Dynamic analysis is used to assess applications while running them.

Each mobile platform has its own security model; they differ in how applications run, how they can communicate, what capabilities devices expose for applications etc. Applications for different platforms are written using different languages and frameworks. Each platform has its own specific vulnerabilities, which can be non-applicable to other platforms. For example, Windows Phone has managed runtime, so memory access issues are very rare.
But there are also vulnerabilities which apply to any mobile platform. OWASP Top 10 mobile risks list [1] states the most important of these platform-independent vulnerabilities. A lot of applications are not standalone applications: they communicate with different services and other applications. They use well-known technologies (HTTP, XML etc.) with well-known vulnerabilities (XSS, XXE, SQLi etc.). A lot of applications have problems related to transferring, receiving and processing data. This usually happens when developers do not follow practices of secure development. Storing passwords in plaintext, storing data in inappropriate places, missing data encryption leads to data interception. This is very important for mobile applications as smartphones usually process and store a lot of private user data (like credentials, geolocation). Disclosure of such data is highly undesirable.

2. Windows Phone platform

Windows Phone platform provides managed environment (.NET) for 3rd party applications. This means that all calls from user code to critical code will pass through safe-critical code, which will check permissions etc. 3rd party application themselves run with minimum permission set, in so called Least Privilege Chamber.
Windows Phone introduces the sandboxing model as well. This means several things:

- Applications cannot communicate with each other
- Applications cannot run in the background
- Applications have their own isolated storages (IS) and cannot access the IS of other applications
- File system structure is hidden from applications

So, as you can see, Windows Phone platform is pretty secure and leaves little space for applications to bypass restrictions. Instead, auditors can focus on application-specific vulnerabilities, such as storing unencrypted used data or disclosing user data by sending it somewhere.

3. Problem definition

The third type of application vulnerabilities includes errors in an application's business logic. These are "logical" errors which are applicable to all mobile platforms. Their exploitation may lead to inappropriate application work. To find such error, auditors should understand the whole business process, business requirements, and how the application actually works. But it is very hard and time consuming to find such vulnerabilities using only static analysis, even if the original source code is accessible. It is much easier to examine applications with live data.

There are usually two main ways to dynamically analyze any client application (and mobile applications are usually just clients):

- Alter the client
- Intercept and tamper with communication data

3.1 Altering the client

If an auditor has access to the application binaries, following steps can performed:

1) Decompile code from binaries to get source code
2) Change source code to add some payload
3) Compile modified code back to binaries

This approach can work for the platforms that use managed environment (Android and Windows Phone) since they have intermediate code (byte-code and IL code). But anyway, it is time consuming and error-prone. Decompilers do not guarantee 100% correct decompilation, so auditors can get source code that does not compile. Compilation errors are costly to fix, especially if the auditor is not qualified enough to recover the source code. Also, the whole process, which has to be iterated many times, is fatiguing and slow.

In case of iOS platform, only disassembling is possible, and it is hard to alter an application at all.

3.2 Intercepting communication

Applications can "communicate" or give away data in two ways: communicating with some server or storing to and loading from device storage.

Internet traffic can be viewed with tools that allow intercepting HTTP traffic, like Fiddler. But there is also a chance that an application does not use HTTP as transport protocol or uses HTTP but does not use plaintext format. An application could use decryption or some serialization format which could be hard to understand and change.

As for application storages, we can't interfere in the process of loading or saving data. We can just observe the results: the data that application has saved. It does not give much information to think about, either.

3.3 Problems with Windows Phone applications

Windows Phone is still a new platform, and while iOS and Android have a lot of handy instruments to tackle the problem of application analysis, like SIRA [2] and DroidBox [3], Windows Phone still lacks this kind of tools.

Static analysis can be performed using a lot of various tools, from Windows Phone SDK tools, like Capability Identification tool, to disassemblers and decompilers.

Digging into Windows Phone specific tools brought Windows Phone Application Analyzer [4], which can perform static analysis by itself and with the help of 3rd part tools like CAT.NET or FxCop.

Although static analysis helps to read and understand code, it is better to see what happens when the application is running. And here comes one of the biggest problems of Windows Phone application analysis — lack of programmable debug interface for both emulator and phone devices. Visual Studio can only be used to debug applications with source code. So we need to somehow get information from running application to understand how it behaves and what data it operates. This can be achieved by dynamic analysis.

Dynamic analysis tools are only presented by XAPSpy [5] and a tool which is based on it and called XapSpyAnalysis [6]. XAPSpy is a dynamic analysis tool that helps to automate work with XAP files and to see application stack trace with method names and parameters. XAPSpy uses static byte-code instrumentation and emulator console to log this information while running an application in emulator.

XapSpyAnalysis extends XAPSpy by adding graphical representation of exported XAPSpy data and possibility to dump stack traces to file.

As you can see, there is not much to start with, and application analysis needs more advanced tools.

4. Solution: Tangerine tool

Tangerine is a new tool that was created to extend existing tools and give auditors more ways to test security. Tangerine is based on XAPSpy, and its main goal is to extend the capabilities of static and dynamic analysis. Below are the main features of Tangerine:

- Automates routine work with XAP files
    - XAP files are automatically unpacked, packed, and deployed
- Performs static analysis
    - Application capabilities analysis
    - Application code structure analysis
    - API usage detecting
- Performs dynamic analysis
    - Logging application stack trace (including parameters and return values)
    - Adding custom code to applications
- No source code required

As .NET assemblies contain MSIL code, it makes them relatively easy to read and modify. Tangerine uses popular Mono.Cecil [7] library to read assembly metadata information (like code structure) and to manipulate MSIL code (allows performing dynamic analysis).

4.1 Static analysis

Tangerine allows performing some kind of static analysis on the target application:

- Allows to view manifest information, including application capabilities
- Allows to view the application tree which includes assemblies, types and methods
- Highlights the methods which use set of “interesting” APIs (System.IO, System.Net and System.Security)
- View IL code

This allows making a quick application overview and finding interesting methods to test. Like methods that directly use System.IO or System.Net or methods around them definitely deals with communication and can process unprotected data. Also Tangerine allows to quickly view methods IL code.

4.2 Dynamic analysis

Dynamic analysis is the main feature of Tangerine. It allows installing hooks into methods. These hooks can perform various actions:

- Log method names
- Log parameters values
- Log return values
- Add custom code to the beginning of method
- Replace method
- Add custom code to the end of method
- Change parameter values with custom code

Logging application stack trace allows security testers to understand application control flow better and maybe to find some interesting functions or argument values and then look at them closer using other hooks.

Hooks with custom code have a lot of reasons to use. You can replace some license check with returning the required value. Or you can return unexpected value from a method or change its parameters values to see how it affects the rest of the code. Or you can perform additional actions on method enter. These hooks make it easy to modify applications and tamper with data, so security testers should figure out which method should be hooked and in which way.

4.3 Limitations

As any tool, Tangerine has its own limitations. Some of them were made intentionally, some of them can be overcome and some not yet:

- Currently, works with emulator only (though instrumented XAP file could be deployed on the device using another tools)
- Does not help to overcome obfuscated code
- Does not work with system assemblies (.NET Framework, SDK)
- Application needs to be decrypted before analysis
- Works with Windows Phone 7 applications and with Windows Phone 8 applications NOT from Windows Phone Store

The last restriction refers to the fact that Microsoft has changed the way how applications are downloaded from Windows Phone Store and how they are stored on the device. Long story short, there are no more MSIL assemblies: all applications on Windows Phone 8 are now downloaded in precompiled state (almost native) and are finally compiled during installation, so we have native images on the device [8].

4.4 Practical usage information

Encrypted XAP files issue

All XAP files for Windows Phone 7 applications downloaded from Windows Phone Store are encrypted and cannot be unpacked using ZIP, so there is no way to analyze them. What auditor can do instead is to unlock device, install an application and then grab the unpacked application from the device. All installed applications are placed to "\Applications\Install\<ProductID>\Install\" path in Windows Phone 7. Product ID stands for unique application identifier and can be looked up in the Store's application URL (like http://www.windowsphone.com/en-us/store/app/<AppName>/<ProductID>).
Tangerine supports both XAP file and directory modes.

Requirements

Tangerine requires .NET Framework 4.0 and Windows Phone 7.1 SDK for correct work.

References

[1] OWASP Mobile Security Project - Top Ten Mobile Risks — https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks
[2] SIRA — http://www.siratool.com/
[3] DroidBox — http://code.google.com/p/droidbox/
[4] Windows Phone Application Analyzer — http://www.securityninja.co.uk/application-security/windows-phone-app-analyser-v1-0-released-today-2/
[5] XAPSpy — http://www.sensepost.com/blog/6081.html
[6] XapSpyAnalysis — http://xapspyanalysis.codeplex.com/
[7] Mono.Cecil — http://www.mono-project.com/Cecil
[8] Deep Dive into the Kernel of .NET on Windows Phone 8 — http://channel9.msdn.com/Events/Build/2012/3-005