Inspection of Windows Phone applications

Dmitriy Evdokimov
Andrey Chasovskikh
About us

Dmitriy ‘D1g1’ Evdokimov
- Security researcher at ERPScan
- Editor of Russian hacking magazine
- DEFCON Russia (DCG #7812) organizer

Andrey Chasovskikh
- Software developer
- Windows Phone addict
Agenda

- Windows Phone intro
- Security model
- All about applications
- Not all applications are secure
- Tools overview
- Deep dive: finding vulnerabilities
- Conclusion
WINDOWS PHONE INTRO
History of Windows Phone

- The successor to the Windows Mobile OS
- 15 Mar 2010 – Windows Phone 7 series announced
- 21 Oct 2010 – Windows Phone 7 released
- 29 Oct 2012 – Windows Phone 8 released

WP7

WP7 NoDo

21 Oct 2011

WP7.5 Mango

27 Sep 2011

WP7.5 Tango

29 Oct 2012

WP8

version time
Market share

- **Android**: 72.4%
- **iOS**: 13.9%
- **Blackberry**: 5.3%
- **Windows Phone**: 2.4%
- **Bada**: 3.0%

Source: Gartner, November 2012
- 125 000+ applications
- Casual apps, social networks, mobile banking, enterprise applications etc.
SECURITY MODEL
Chamber concept, WP7

- **Trusted Computing Base (TCB)**
  - Kernel, kernel-mode drivers

- **Elevated Rights Chamber (ERC)**
  - Services, user-mode drivers

- **Standard Rights Chamber (SRC)**
  - Pre-installed applications

- **Least Privileged Chamber (LPC)**
  - Applications from WP store
Chamber concept, WP8

- **Trusted Computing Base (TCB)**
  Kernel, kernel-mode drivers

- **Least Privileged Chamber (LPC)**
  All other software: services, pre-installed apps, application from WP store
Capabilities

WMAppManifest.xml

Windows Phone 7
- Camera
- Contacts
- Location services
- Owner/phone identity
- Network services
Etc.

Windows Phone 8
- All WP7 capabilities
- NFC
- SD card access
- Wallet
- Speech recognition
- Front camera
Etc.

Undocumented
- Native code
- SMS API
- Access to user properties
- SIM API
Etc.
Sandboxing concept

- No app communication in WP7
- Limited app-to-app in WP8
- File system structure is hidden
- Isolated storages
App-to-App, WP8

- **File associations**
  - LaunchFileAsync()
  - Reserved: xap, msi, bat, cmd, py, jar etc

- **URI associations**
  - LaunchUriAsync()
  - Reserved: http, tel, wallet, LDAP, rlogin, telnet etc
  - Proximity communication using NFC
Isolated Storage

Physical File Storage

Isolated Storage

Isolated Settings Storage

Isolated File Storage

Files

Directory

Database
Signing

- Store applications are signed in WP7
- All binaries get signed since WP8
- Application file get signed
  - Kind of checksum file is put into applications
- Applications XAP files have undocumented format (since Aug 2012)
ALL ABOUT APPLICATIONS
.NET and CLR, WP7

Applications

Developer Platform (XAML, XNA, Device services)

.NET Compact Framework (BCL + Silverlight flavor)

WP7 OS, WinCE based
Framework
.NET and CLR, WP8

Applications

Developer Platform (XAML, XNA, Device services)

.NET Framework (CoreCLR)

WP8 OS, Win8 based
Framework

- Windows Phone API
  - .NET
    - Managed
  - Windows Phone Runtime
    - Managed & Native
  - Direct3D, XAudio2, MF, WASAPI, Win32 & COM
    - Native
Application file structure

- Application assemblies
- Resources
- AppManifest.xaml
- WMAppManifest.xml
- WMInteropManifest.xml*

* — optional for WP7, absent in WP8
Submission and certification

App Creation

.xap

App Submission

XAP File Validation

Adding Metadata

Certification Testing

Signing

Publication in Marketplace

Source code
Applications on a device

WP7:
\Applications\Install\<ProductID>\Install\n  - Content from XAP
  - WMAAppPRHeader.xml (package signature)
\Data\<ProductID>\Data\IsolatedStorage

Same idea in WP8, i.e. install path:
C:\Data\Programs\<ProductID>\Install\
NOT ALL APPLICATIONS ARE SECURE
Security assessment

Device/Emulator

App

Data channel

App

Server
Mobile applications security assessment

Prepare environment
- Get app (unpack/decrypt)
- Configuration device/emulator

Static analysis
- Properties of program compilation
- Metadata analysis
- Code analysis

Dynamic analysis
- How application works with file system/network
- Runtime code analysis
OWASP Top 10 Mobile Risks

1. Insecure Data Storage
2. Weak Server Side Controls
3. Insufficient Transport Layer Protection
4. Client Side Injection
5. Poor Authorization and Authentication
6. Improper Session Handling
7. Security Decisions Via Untrusted Inputs
8. Side Channel Data Leakage
9. Broken Cryptography
10. Sensitive Information Disclosure
WP vs. Android vs. iOS vulnerabilities

WP7 (C#/VB)
WP8 (C#/VB/C/C++)

iOS (Objective-C)

Android (Java)

Platform independent vulnerabilities
Platform specific vulnerabilities

Note: Main programming languages in brackets
TOOLS OVERVIEW
Arsenal

- Device
  - Full unlock
- Emulator
- Windows Phone Device Manager

- Network proxy: Burp Suite, Charles etc.
- .NET tools: .Net Reflector, ILSpy etc.
- IDA Pro

- RAIN, Boyan Balkanski
- Windows Phone App Analyzer, David Rook
- XAPSpy, Behrang Fouladi
  - XapSpyAnalysis, David Rook
Main issue

Static analysis is insufficient.
Lack of dynamic analysis tools:

• IDE allows debugging with source code only
• No programmable debugging interface
  • Managed code

Solution: static byte code instrumentation.
Tangerine
Automates routine with XAP files

- Unpacking
- Removing application signature
- Resigning assemblies
- Packing
- Deploying
Static analysis

- Application info
- Application capabilities
- Code analysis
  - Code structure analysis
  - API usage analysis
  - View IL code
Dynamic analysis

- Log application stack trace
  - Method names
  - Method parameters
  - Return values
- Run custom code
  - On method enter
  - Replace method
  - On method exit
- Change parameters values
DEEP DIVE: FINDING VULNERABILITIES
DEMO
How it works

(1) Changing CIL code
(2) Emulator console (writing/reading)
CIL Instrumentation

IL_0000: nop
IL_0001: ldarg.1
IL_0002: ldarg.2
IL_0003: add
IL_0004: stloc.0
IL_0005: br.s  IL_0007
IL_0007: ldloc.0
IL_0008: ret

IL_0000: nop
IL_0001: ldarg.1
IL_0002: ldarg.2
IL_0003: add
IL_0004: stloc.0
IL_0005: ldloc.0
IL_0006: call  void [mscorlib]System.Console::WriteLine(int32)
IL_000b: nop
IL_000c: ldloc.0
IL_000d: stloc.1
IL_000e: br.s  IL_0010
IL_0010: ldloc.1
IL_0011: ret
Limitations

- Emulator only
- Does not help to overcome obfuscated code
- Does not work with system assemblies
- Applications from store need to be decrypted
- Windows Phone 7 only
Cloud Compilation, WP8

C# Source Code → C# Compiler → CIL Assembly → MDIL Compiler → MDIL Assembly

Cloud

Download

Device

Native DLL → Native Image Generator → MDIL Assembly

Run
MDIL in work

```
int foo (int a) { return a+j }
```

R0 = this
R1 = a
R0 + 0x10 = j, where j is a field from base class

```
LDR R0, [R0 + 0x10]
ADD R0, R0, R1
BX LR

LDR R0, [R0 + "fieldToken<>"]
ADD R0, R0, R1
BX LR
```
MDILDump

```
METHOD_00000000:
000000: b4 bb 01 01 b2 b8 00 b9 9e 00 00 01 4e 03 00 02
000010: 00 20 4e 04 00 bb
MDIL_0000: B4 BB    PUSH_REGS    EBX,ESI,EBP,R12,R13,R15,
MDIL_0002: 01 01    LIT_MACHINE_INST_1  01
MDIL_0004: B2    EBP_FRAME
MDIL_0005: B8 00    FRAME_SIZE  00
MDIL_0007: B9     END_PROLOG
MDIL_0008: 9e 00 00 01    LOAD_STRING    EAX, 70000001
MDIL_000C: 4e 03 00    CALL_REF  0a000003
MDIL_000F: 02 00 20    LIT_MACHINE_INST_2  00 20
MDIL_0012: 4e 04 00    CALL_REF  0a000004
MDIL_0015: BB     EPILOG_RET
Method Size: 23 (0x17) bytes, Routine: 22 (0x16) bytes, Exceptions: 0
```

http://github.com/WalkingCat/mdildump/
Future work

- Support Windows Phone 8 applications
  - MDIL instrumentation
  - Windows Phone RT
- Add new features
  - Code graphical representation
  - Data flow analysis
- Fix bugs ;}
CONCLUSION
Conclusion

- Greater attack surface in WP8
- App-to-App
- Applications that use native code
- New technologies
- Logical bugs never die
Thanks

- Evgeny Bechkalo
- DSecRG team
Q&A

Dmitry Evdokimov
d.evokimov@erpscan.com
@evdokimovds
Andrey Chasovskikh
http://andreycha.info
@andreycha

Tangerine: http://github.com/andreycha/tangerine