WINDOWS PHONE 8
APPLICATION SECURITY

HackInParis 2013

Dmitriy Evdokimov
Andrey Chasovskikh
About us

Dmitriy ‘D1g1’ Evdokimov
- Security researcher at ERPScan
  - Mobile security, RE, fuzzing, exploit dev etc.
  - Editor of Russian hacking magazine
  - DEFCON Russia (DCG #7812) co-organizer

Andrey Chasovskikh
- Software developer
- Windows Phone addict
Agenda

• Intro
• Security model
• First steps in Windows Phone 8
• Applications
• Application security
• Conclusion
INTRO
Intro

- 29 Oct 2012 – Windows Phone 8 released
- Based on Windows 8 core
  - ARM architecture
- Market share: 3,2% (Q1 2013, IDC)
- 145 000+ applications in Windows Phone Store
SECURITY MODEL
Chambers

- Trusted Computing Base (TCB)
  Kernel, kernel-mode drivers
- Least Privileged Chamber (LPC)
  All other software: services, pre-installed apps, application from WP store
## Capabilities

### WMApplManifest.xml

<table>
<thead>
<tr>
<th>Developers</th>
<th>OEM Developers</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Network</td>
<td>- Cell API</td>
<td>- Debug</td>
</tr>
<tr>
<td>- Camera</td>
<td>- Device management</td>
<td>- SMS API</td>
</tr>
<tr>
<td>- NFC</td>
<td></td>
<td>- Live ID</td>
</tr>
<tr>
<td>- SD card access</td>
<td></td>
<td>- SIM API</td>
</tr>
<tr>
<td>- Wallet</td>
<td></td>
<td>Etc.</td>
</tr>
<tr>
<td>- Speech recognition</td>
<td></td>
<td>Total 39</td>
</tr>
<tr>
<td>- Front camera</td>
<td></td>
<td>Total 350+</td>
</tr>
<tr>
<td>Etc.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total 27</strong></td>
<td><strong>Total 39</strong></td>
<td><strong>Total 350+</strong></td>
</tr>
</tbody>
</table>

HackInParis 2013
Sandboxing

- File system structure is hidden
- Local folder
  - Former isolated storage
- Limited app-to-app communication

Diagram:

- Chamber
  - App1
  - Local folder for App1
- Chamber
  - App2
  - Local folder for App2

URI, files
App-to-app communication

- File types 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
Local folder

Physical File Storage

Local Folder

Settings Storage

File Storage

Files

Directory

Database
Application protection

- All binaries are signed
- Application file is signed
  - Kind of checksum file is put into applications
- Certificate pinning for Store
- XAP file has DRM key
The Microsoft PlayReady Ecosystem
XAP file protection

• Before august 2012
  – ZIP archive
  – Sign

• After august 2012
  – New file format
  – PlayReady Header
  – AESCTR algorithm
FIRST STEPS IN WINDOWS PHONE 8
Windows 8 vs Windows Phone 8

- WP8 is migrating from the WinCE core to the WinNT core
- Win8/emulator (x86)
- WinRT/device (ARM)

http://intrepidusgroup.com/insight/2012/12/windows-phone-8-and-windows-8-similarity/
WP8 emulator

• Hyper-V images
  – %ProgramFiles(x86)\Microsoft SDKs\Windows Phone\v8.0\Emulation\Images\n
• Emulator vs. Device
  – x86
  – Fake binaries
    • FakeLed.sys, Fakevibra.sys, FakeModem.dll etc.
  – Different user-agent
  – Prohibited to install apps from the Store
WP8 device

• Windows Phone 8 has standardized bootloader
  – Full flash images are available

• ImgMount tool
  – FFU Image file as a virtual hard drive

C:\DATA\work\WindowsPhone8>ImgMount.exe RM825_1232.2110.1244.3002_RETAIL_eu_euro1_375_02_104614_prd_signed.ffu
WP8 ROM Image Tools v.1.0.204
htc ROM Image Editor (â) 2007-2012 AnDim & XDA-Developers
ImgMount Tool v.1.0.15

<htcRIE> Mounting the image file : 'RM825_1232.2110.1244.3002_RETAIL_eu_euro1_375_02_104614_prd_signed.ffu'
Loading .FFU image ... ok
<htcRIE> !WARNING! Successfully detached vhd file : 'C:\Users\d.evdkimov\AppData\Local\Temp\kml1501.vhd'
Creating virtual disk ... ok
Mounting MainOS partition as : '\RM825_1232.2110.1244.3002_RETAIL_eu_euro1_375_02_104614_prd_signed.mnt' ... ok
<htcRIE> Successfully mounted an image file.
Reversing WP8 internals

- No debug symbols
- Tip: restore information from Event Tracing for Windows (ETW)
- Use IDAPython

*InstallerWorker.exe*
Windows API calls

• Full Windows API is not available by default
• Originally posted on XDA for WindowsRT apps
  – Find kernerbase.dll address (“MZ”) -> Get “LoadLibraryA” and “GetProcAddress” functions -> call any function you want
• Works for Windows Phone 8
APPLICATIONS
.NET and CLR

Applications

Developer Platform (XAML, XNA, Device services)

.NET Framework (CoreCLR)

WP8 OS, Win8 based
Frameworks

WINDOWS PHONE API

.NET
Windows Phone Runtime
Direct3D, XAudio2, MF, WASAPI, Win32 & COM

MANAGED
MANAGED & NATIVE
NATIVE
Application kinds

- Microsoft
- OEM
  - XAP files are not encrypted (~ZIP)
  - C:\PROGRAMS\CommonFiles\Xaps\
- Windows Phone Store apps
  - C:\Data\Programs\{ProductID}\Install\
- Company applications
  - XAP files are not encrypted (~ZIP)
  - Company hubs
- Developer applications
  - Need developer unlock
Application file structure

- Application assemblies (in various formats)
- Resources
- AppManifest.xaml
- WMAppManifest.xml
APPLICATION SECURITY
“One of the goals of the Windows Phone app platform is to foster the creation of apps that are secure by design and secure by default.”

Security for Windows Phone
Application entry points

- User input
- SD card
- Sockets
- URI

- Web
  - Bluetooth
  - NFC
  - Speech2Text

**Green** – Windows Phone 7
**White** – Windows Phone 8
Vulnerabilities

Windows Phone 8
(C#/VB/C/C++)

iOS
(Objective-C)

Android
(Java)

Platform independent vulnerabilities

Platform specific vulnerabilities

Note: Main programming languages in brackets

HackInParis 2013
Work with SD card

- WP8 allows only read operations
- Only registered file types
- Files on SD cards are not encrypted

<table>
<thead>
<tr>
<th>OS</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>Work with SD card is absent</td>
</tr>
<tr>
<td>Android</td>
<td>READ/WRITE</td>
</tr>
</tbody>
</table>
Privacy

• Device Unique ID
  – Requires ID_CAP_IDENTITY_DEVICE
  – DeviceExtendedProperties.GetValue("DeviceUniqueId")

• Windows Live Anonymous ID
  – Requires ID_CAP_IDENTITY_USER
  – UserExtendedProperties.GetValue("ANID2")

• Both identifiers are per-publisher

<table>
<thead>
<tr>
<th>OS</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>UDID (apps that use UDIDs are no longer accepted, from May 1, 2013)</td>
</tr>
<tr>
<td>Android</td>
<td>telephonyManager.getDeviceId()</td>
</tr>
</tbody>
</table>
Privacy, part 2

• Device name, manufacturer, firmware versions
  – Requires ID_CAP_IDENTITY_DEVICE
  – DeviceStatus class

• Location tracking
  – ID_CAP_LOCATION
  – GeoCoordinateWatcher class

<table>
<thead>
<tr>
<th>OS</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>UDID (apps that use UDIDs are no longer accepted, from May 1, 2013)</td>
</tr>
<tr>
<td>Android</td>
<td>telephonyManager.getDeviceId()</td>
</tr>
</tbody>
</table>
Secure storage

- Device can be encrypted (not for all countries)
  - BitLocker 2.0/TPM
  - Available only in business settings
- Data Protection API (DPAPI)
- System.Security.Cryptography
- Algorithms: AES, HMACSHA1, HMACSHA256, Rfc2898DeriveBytes, RSA, SHA1, SHA256

<table>
<thead>
<tr>
<th>OS</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>Keychain, /System/Library/Frameworks/Security.framework</td>
</tr>
<tr>
<td>Android</td>
<td>android.security.KeyChain (from 4.0)</td>
</tr>
</tbody>
</table>
Data leak

- Keyboard cache is isolated per-application
- Cache for applications that access internet
  - Controlled by OS

<table>
<thead>
<tr>
<th>OS</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>plist, Custom created documents, Preferences, Logs, Cache data,</td>
</tr>
<tr>
<td></td>
<td>Keyboard cache, Pasteboard cache, Cookies</td>
</tr>
<tr>
<td>Android</td>
<td>shared_preference, logs, external storage, MODE_WORLD_READABLE</td>
</tr>
<tr>
<td></td>
<td>or MODE_WORLD_WRITETABLE</td>
</tr>
</tbody>
</table>
Work with URI

- Handling function: MapUri()
- Filter user input
- Exclude critical arguments from URI
  - Ex.: prgrm://command?request=data&role=admin

<table>
<thead>
<tr>
<th>OS</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>openURL(), handleOpenURL()</td>
</tr>
<tr>
<td>Android</td>
<td>android.net.Uri class</td>
</tr>
</tbody>
</table>
Cross-site scripting (XSS)

- WebBrowser control (based on IE10)
- JavaScript is disabled by default
- To see if enabled:
  - WebBrowser.IsScriptEnabled = true
  - `<WebBrowser IsScriptEnabled = “True” />`

<table>
<thead>
<tr>
<th>OS</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>UIWebView Class + stringByEvaluatingJavaScriptFromString() shouldStartLoadWithRequest()</td>
</tr>
<tr>
<td>Android</td>
<td>WebView.getSettings().setJavaScriptEnabled(); WebView.getSettings().setPluginsEnabled();</td>
</tr>
</tbody>
</table>

HackInParis 2013
Directory traversal

• Local folder API accepts paths with traversal
  – IsolatedStorageFile class (WP7)
  – StorageFolder class

• Win32 storage API

<table>
<thead>
<tr>
<th>OS</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>contentsAtPath, fileHandleForReadingAtPath, _fopen etc.</td>
</tr>
<tr>
<td>Android</td>
<td>ContentProvider + incorrect or missing rights, files functions</td>
</tr>
</tbody>
</table>
XML External Entity (XXE)

• System.Xml namespace
  – Entity resolving is prohibited by default
• Entities can be resolved by using custom XmlResolver for XmlDocument

<table>
<thead>
<tr>
<th>OS</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>libXML2 + _xmlParseMemory, NSXMLParser + setShouldResolveExternalEntities:YES</td>
</tr>
<tr>
<td>Android</td>
<td>setFeature(external-general-entities, True)</td>
</tr>
</tbody>
</table>
SQL injection

- **Bad:**

```csharp
string name = "...;
SqlCommand cmd = new SqlCommand("SELECT * FROM People WHERE Name = '" + name + "'");
```

- **Good:**

```csharp
string name = "...;
SqlParameter paramName = new SqlParameter("@Name", name);
SqlCommand cmd = new SqlCommand("SELECT * FROM People WHERE Name = @Name");
cmd.Parameters.Add(paramName);
```

<table>
<thead>
<tr>
<th>OS</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>sqlite3_exec()</td>
</tr>
<tr>
<td>Android</td>
<td>query(), rawQuery()</td>
</tr>
</tbody>
</table>
Memory corruption bugs

- Developers can use native code
- Format string, BoF, use-after-free etc.
  - C/C++ functions
- Compilation flags: /sdl, /GS, /DYNAMICBASE, /NXCOMPAT

<table>
<thead>
<tr>
<th>OS</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>-fPIE, -fstack-protector-all, -fobjc-arc</td>
</tr>
<tr>
<td>Android</td>
<td>Only in native libs, -fstack-protector, -Wformat-security, NX, ASLR, PIE</td>
</tr>
</tbody>
</table>
CONCLUSION
Conclusion

- Windows Phone 8 is pretty secure
- Greater attack surface
- Security-related API
  - More flexible than in iOS
  - More simple than in Android
Dmitry ‘D1g1’ Evdokimov
d.evdokimov@erpscan.com
@evdokimovdvs
Andrey Chasovskikh
http://andreycha.info
@andreycha