

The nightmare behind the cross platform mobile apps dream

Marco Grassi

@marcograss

MGrassi@nowsecure.com

Sebastián Guerrero

@0xroot

SGuerrero@nowsecure.com

Agenda

- Background introduction
- What Cross-platform Frameworks are out there?
- Concerns
- The uniform attack surface
- Code formats and code retrieval
- Runtime manipulation and dynamic analysis
- Some examples of vulnerabilities in the frameworks
- Conclusions



NowSecure™

Who are we?



Marco Grassi
Mobile Security Researcher
NowSecure



Sebas Guerrero
Mobile Security Researcher
NowSecure

Warning!



Part of those slides were made while drinking alcohol
in international flights!

Background intro

Background

The mobile market is fragmented. Developers want their app on multiple platforms, at least iOS and Android.

This caused a growth in the number of tools and frameworks available for cross platform development (code reuse between multiple platforms) with different technologies.

Period	Android	iOS	Windows Phone	BlackBerry OS	Others
Q4 2014	76.6%	19.7%	2.8%	0.4%	0.5%
Q4 2013	78.2%	17.5%	3.0%	0.6%	0.8%
Q4 2012	70.4%	20.9%	2.6%	3.2%	2.9%
Q4 2011	52.8%	23.0%	1.5%	8.1%	14.6%

Source: IDC, 2014 Q4

Background

Those frameworks usually achieve code portability by leveraging HTML5 technology or interpreters with bindings on the native code platform, to expose the API of the underlying system in a way as platform independent as possible.

The result is that lot of code, if not all, can be reused.



Background

So far everything seems great, we can code once and run in every platform!

But how is the quality of the code of these frameworks?

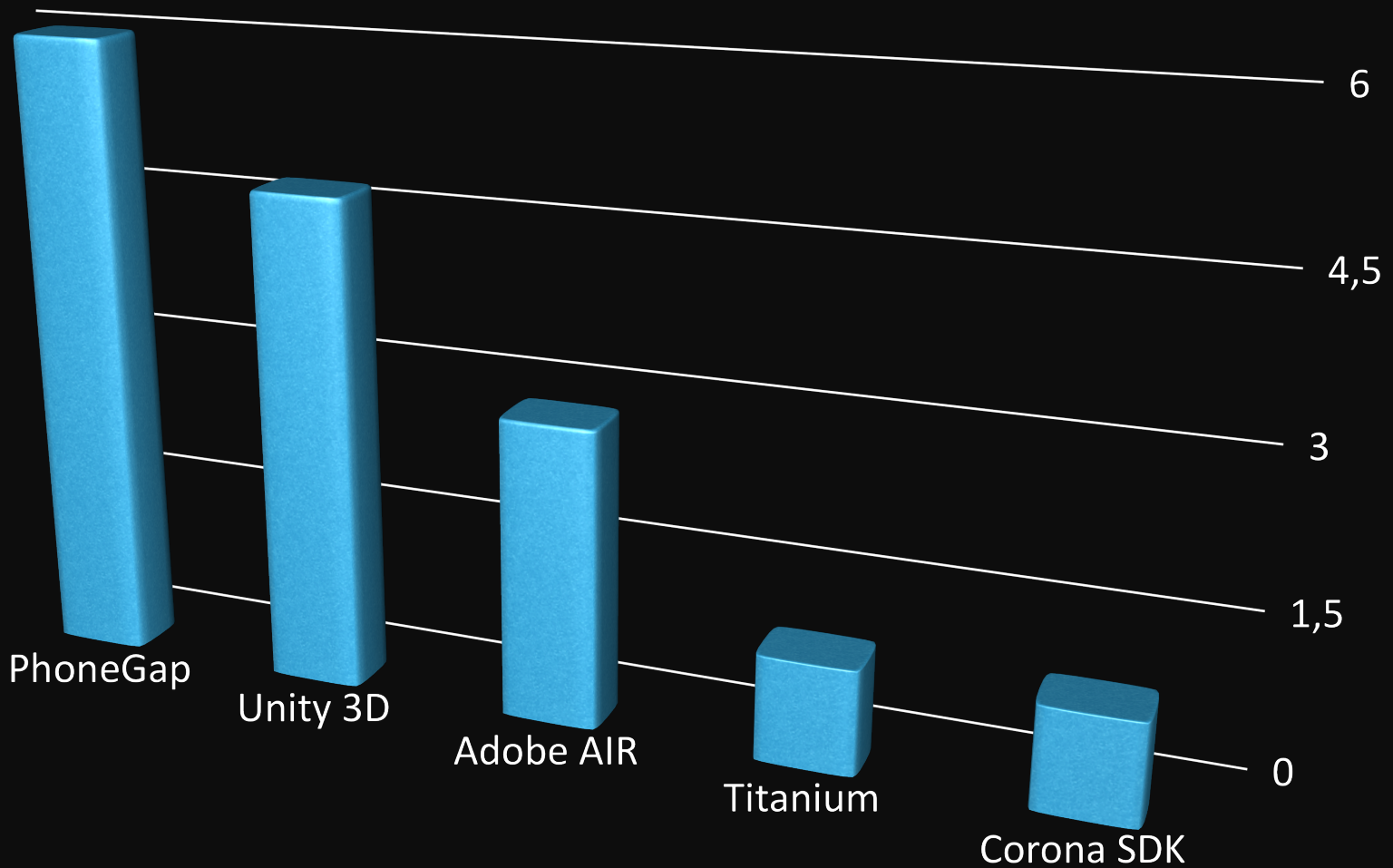
Do they introduce security vulnerabilities if we use them?

Are those vulnerabilities shared by all the applications using the same framework?



What Cross-platform Frameworks Are Out There?

Some Stats



■ % # of Android Apps (appbrain.com)



Cordova / PhoneGap

- Mobile development framework to write Mobile applications using web technologies such as HTML5, Javascript and CSS3.
- Cordova is the Opensource core, PhoneGap is owned by Adobe which leverages Cordova.
- Cordova leverages a “bridge” between javascript and native code, to invoke native methods from javascript code.
- OpenSource - <https://cordova.apache.org/>



Cordova / PhoneGap

- Some history of known vulnerabilities and CVEs
- Great job by David Kaplan and Roei Hay of IBM Security Systems
- CVE-2014-3500, CVE-2014-3501, CVE-2014-3502, all related to Android Intents somehow “mistrusted” leading to XAS, bypasses and leak of data.



- Cross platform game creation system, including the major mobile platforms.
- You can develop the game with Mono, so in .NET
- <http://unity3d.com/>



Adobe AIR

- Very popular cross platform runtime, not only on mobile but also on desktop. Used both in apps and games.
- You can develop your project with Adobe Flash or ActionScript.



Adobe AIR

- Lot of CVE, maybe mainly because they are shared with Adobe Flash Player for Desktop browsers
- In this context (cross platform application), those vulns are less relevant because you don't run remote/untrusted code like in the browser, you just execute code shipped within your application.
(if you don't do strange things)



Titanium Appcelerator

- Open source - <http://www.appcelerator.com/>
- You can develop your native mobile application in javascript
- The javascript runs on a interpreter and uses native UI and functionalities
- The IDE is Eclipse based



Corona SDK

- SDK to develop 2d games and apps
- You write Lua code using bindings to C++ and OpenGL



Kony Framework

- Platform to develop cross platform applications.
- Native Mobile Applications can be written in Lua or Javascript.

Concerns

Concerns

- Code reuse implies uniform attack surface between apps using the same framework.
- Vulnerabilities and attacks can be reused among different apps.
- The code is stored in high level languages or byte code. This makes reverse engineering process easier or non-existent as we will see soon.

Code Format: Some Examples On How To Retrieve The Code

With great portability...

... often comes great meta datas and easy reverse engineering.





Cordova / PhoneGap

- Almost all the code is written in javascript, the UI is developed in HTML5 and CSS3.
- The code by default it's just bundled in plain text inside of the mobile app, so retrieving it and reversing is trivial



Cordova / PhoneGap

- On iOS the Cordova code is bundled as a resources, or in a **www/** subfolder of the app bundle
- On Android it's usually bundled in the **assets/** folder of the apk



Cordova / PhoneGap

```
iPad:/var/mobile/Containers/Bundle/Application/  
43F3709A-3843-4F67-88A4-3E387D805DAA/Amazon.app root#  
find . -name "*.ios.js"  
./cordova.ios.js  
./mash.ios.js
```

```
→ HelloWorldNew.app ls  
Default-568h@2x~iphone.png Default@2x~iphone.png icon-40@2x.png icon-76.png  
Default-667h.png Default~iphone.png icon-50.png icon-76@2x.png  
Default-736h.png HelloWorldNew icon-50@2x.png icon-small.png  
Default-Landscape-736h.png Info.plist icon-60.png icon-small@2x.png  
Default-Landscape@2x~ipad.png MainViewController.nib icon-60@2x.png icon.png  
Default-Landscape~ipad.png PkgInfo icon-60@3x.png icon@2x.png  
Default-Portrait@2x~ipad.png config.xml icon-72.png WWW  
Default-Portrait~ipad.png icon-40.png icon-72@2x.png
```



Cordova / PhoneGap

→ assets tree

```
.  
├── OpenSans-Light.ttf  
├── OpenSans-Regular.ttf  
├── defappratepack.json  
├── deflanpack.json  
├── defpermissions.json  
├── www  
│   ├── akbank.lib.js  
│   ├── cashflow_dummy.js  
│   ├── cordova.android.js  
│   ├── cordova.js  
│   ├── index.html  
│   └── index_cashflow.html
```



Cordova / PhoneGap

```
dummyData = [  
  {"id":0,"name":"Diğer","sliced":false,"mStartAngle":144.0,"y":60.0,"mAngle":216.0,"selected":false,"order":0,"isListenClickEvent":false},  
  {"id":1,"name":"Gıda","sliced":false,"mStartAngle":72.0,"y":20.0,"mAngle":72.0,"selected":false,"order":0,"isListenClickEventOnSlice":false},  
  {"id":2,"name":"Turizm ve Eğlence","sliced":false,"mStartAngle":54.0,"y":5.0,"mAngle":18.0,"selected":false,"order":0,"isListenClickEventOnSlice":false},  
  {"id":3,"name":"Havayolları","sliced":false,"mStartAngle":0.0,"y":15.0,"mAngle":54.0,"selected":false,"order":0,"isListenClickEventOnSlice":false},  
];  
  
dummyData = [  
  {"name":"Diğer","isListenClickEventOnSlice":true,"mAngle":224.71200000000002,"mStartAngle":135.288,"y":62.42,"selected":false,"sliced":false,"id":0},  
  {"name":"Akaryakıt","isListenClickEventOnSlice":true,"mAngle":51.948,"mStartAngle":83.34,"y":14.43,"selected":false,"sliced":false,"id":1},  
  {"name":"Gıda","isListenClickEventOnSlice":true,"mAngle":83.34,"mStartAngle":0.0,"y":23.15,"selected":false,"sliced":false,"id":2},  
];  
  
dummyData = [  
  {"assetsAmount":"885,82 TL","name":"Vadesiz Altın Mevduat Hesaplarınız","y":93.6,"mAngle":336.96,"mStartAngle":24.479999999999999},  
  {"assetsAmount":"50.221.162,06 TL","name":"Vadesiz Döviz Mevduat Hesaplarınız","y":5.3,"mAngle":19.08,"mStartAngle":5.4000000000000004},  
  {"assetsAmount":"10.386.682,04 TL","name":"Vadesiz TL Mevduat Hesaplarınız","y":1.1,"mAngle":3.9600000000000004,"mStartAngle":1.08},  
  {"assetsAmount":"3.790,89 TL","name":"Yatırım Fonlarınız","y":0.1,"mAngle":0.36000000000000004,"mStartAngle":1.08,"id":3,"select":false},  
  {"assetsAmount":"2.023,01 TL","name":"Repolarınız","y":0.1,"mAngle":0.36000000000000004,"mStartAngle":0.7200000000000001,"id":4,"select":false},  
  {"assetsAmount":"25.958,13 TL","name":"Vadeli Döviz Mevduat Hesaplarınız","y":0.1,"mAngle":0.36000000000000004,"mStartAngle":0.36},  
  {"assetsAmount":"31.559,65 TL","name":"Vadeli TL Mevduat Hesaplarınız","y":0.1,"mAngle":0.36000000000000004,"mStartAngle":0.0,"id":5},  
];
```



Titanium Framework

- Real code is written in JavaScript.
- Load asset data at runtime through the AssetCryptImpl class.
- Assets range are defined in a HashMap within the initAssets class.
- Assets bytes are contained in a CharBuffer defined in the initAssetsBytes.



Titanium Framework

```

((Map)v0).put("alloy/sync/properties.js", new Range(2481952, 1104));
((Map)v0).put("alloy/sync/sql.js", new Range(2483056, 7312));
((Map)v0).put("alloy/underscore.js", new Range(2490368, 13488));
((Map)v0).put("alloy/widget.js", new Range(2503856, 800));
((Map)v0).put("alloy.js", new Range(2504656, 6560));
((Map)v0).put("av.nav.js", new Range(2511216, 3632));
((Map)v0).put("avforms.js", new Range(2514848, 1552));
((Map)v0).put("avmainloader.js", new Range(2516400, 5984));
((Map)v0).put("avnotifications.js", new Range(2522384, 14624));
((Map)v0).put("avparsepush.js", new Range(2537008, 3040));
((Map)v0).put("avrequest.js", new Range(2540048, 35056));
((Map)v0).put("avutils.js", new Range(2575104, 4016));
((Map)v0).put("navigate.js", new Range(2579120, 28320));
((Map)v0).put("pushactivity.js", new Range(2607440, 688));

```

```

private static CharBuffer initAssetsBytes() {
    CharBuffer v0 = CharBuffer.allocate(2610100);
    v0.append(";nGà\u00868ó\u008BÜilwQ\u008Eám \u0080\u001E~i$;\\l0ýù70/6éh\"Là' _EIV7a?\u00
    v0.append(" «k`á\u009E|F*#D.\u0001Ä\u0015;\u00834é'Ê/ai>S/bx\u0018s\u009E\u0002Äéz\uf0a
    v0.append("T\ffp4É\u0004r\u0016ò\u008AYbFrâ}m-D\tÄNS\u0080ÇNEv\u009D!fÄE\u0086\u0015D<
    v0.append("JfJ\u00880ÐE~i\u0006p;È\u001C\u0091\u001F\u0091\u008D\u0085+<0óIhfÈ*`bA\u00
    v0.append("ÊâB<\u0094M\u0088\u008B40Ø.8 øÛ,\u001DI E\u007FW\u009B8qf\`âZ\u0092â!\u001
    v0.append("4Û{±f@e\u0099=U> \u00952È\u001C8BÇ0ç«D\u0018\u0099\u000Er\u009C?\u0004H\u00
    v0.append("\u0014 \u0081.,AS)\u0004>\u008C7Ç\n \u001A\u0003\u0088eyNP\u0098ir;f\u001CÑ
    v0.append("\u0089;.^`p\u0013;6\\50\u001A\u0081cG\u0001Ä\u0093p\u0018ø<$+1] (=i\ri\Ä\rnþ
    v0.append("wRm6Ø\u00805ðLo0Ü4\u0085\" \u008BY0\u001E\u0082ÈÏ' \u0011\u009EG\"7\u001Fyp\
    v0.append("I.YÄáinú=iib470wi\u009BfU\u0000áóòD\u008C~f1YX*`/D+*Ä<~ý\u0085f4\u009E\u009
    v0.append("â~!fH\BM\u0015Kl=äxn[G=×óMéýÜ[Pniä\u001F[\u0017ÖyÈ\néó!L\u0003\u008EeÄeb\u00
    v0.append("^Fk\u0010Ø\u0017K\u009FwGfiZç:\u000E^J'\u000B'\u007F'z\u0012ÈX4\u001D\u0000
    v0.append("4\u0095ID9\u0001ÿ*}\u009DfU+\u0096\u008A\u0082J&ø`4UH*0\` \u0001\u0091' \u0
    v0.append("40\u0096k\u0098\u0090iE?0ó«n[*[Bd\u0006A\u0011t\t0âu8Y\u0084\u0000\u00960qø
    v0.append("\u00858 {i#\u0000ý~BÈé\u0094\u0085\u0017ÛpÄ\u0096áâÈ×U]mSGq\u008F[R\u0086óX0
    v0.append("ù\u00004\u009DfPn{[fqJ\u009EV+\u0000Pr äk4=\u009D^D#~\u0002\u0089P@e\u008B) !i

```



Titanium Framework

- One python script to rule them all:
 - Parse the code looking for the 'initAssets' method:
 - `.method private static initAssets()Ljava/util/Map;`
 - Apply some regular expression to spot the HashMap containing all the assets:
 - `'invoke-direct \{(v[0-9]+), (v[0-9]+), (v[0-9]+)\}, Lcom/*****/*****/AssetCryptImpl\;$Range;-><init>\(II)V'`
 - Repeat the same process for the Ljava/util/Map call:
 - `'invoke-interface \{(v[0-9]+), (v[0-9]+), (v[0-9]+)\}, Ljava/util/Map;.*'`
 - Once all the ranges have been retrieved, its time to extract the assets bytes:
 - `start_init_assets = ".method private static initAssetsBytes()Ljava/nio/CharBuffer;"`
 - `const_string = 'const-string v1, "'`



Titanium Framework

- But the assets are encrypted... NO PROBLEM, DO YOU EVEN REVERSE ENGINEERING, BRO?!?
 - The crypto is described in the JNI function
'Java_org_appcelerator_titanium_TiVerify_filterDataInRange' in 'libtverify.so'

```
byte[] filterDataInRange(byte[] bytes, int offset, int count) {  
    SecretKeySpec key = new SecretKeySpec(bytes, bytes.length - 0x10, 0x10, "AES");  
    Cipher cipher = Cipher.getInstance("AES");  
    cipher.init(Cipher.DECRYPT_MODE, key);  
    return cipher.doFinal(bytes, offset, count);  
}
```



Titanium Framework

- Yeah, OK, enough theory, show me some real sh*t.

```
[+] Extracting "xxx/ui/checkDeposit/DepositLandingPage.js"  
[+] Extracting "xxx/ui/transfers/TransferApprovedPage.js"  
[+] Extracting "shared/xxx/gbapi/checkDepositHistoryResponse.js"  
...  
./xxx/modules:  
total 40  
-rw-r--r-- 1 sebas staff 1560 Mar 9 03:09 BalanceHelper.js  
-rw-r--r-- 1 sebas staff 1618 Mar 9 03:09 StepUpServiceWrapper.js  
-rw-r--r-- 1 sebas staff 5675 Mar 9 03:09 UserContext.js  

```




Kony Framework

- If Lua is used, the code is stored inside a “konyappluabytecode.o.mp3” (wut)

```
→ assets file konyappluabytecode.o.mp3  
konyappluabytecode.o.mp3: Lua bytecode, version 5.1
```

- Can be decompiled with OSS decompilers

```
function popupAppIntegrityError()  
  print("Start Of Display a Popup For Application Integrity Has Been Compromised")  
  if "android" == os.platform().name then  
    gblCommonPopupInfo = {
```



- .NET code, trivial to reverse if not obfuscated, lot of good quality decompilers out there, like ILSPY.
- You can find the code in the **assets/** folder in Android for example.



```
ILSpy
File View Help
IL
Search
ISmartTextOutput
Language
Languages
MainWindow
OpenFromGacDialog
Base Types
GacEntry
gacEntries : ObservableCollect
filteredEntries : ObservableCol
cancelFetchThread : modreq:l
filterTextBox : TextBox
listView : ListView
nameColumn : SortableGridVw
okButton : Button
_contentLoaded : bool
CS$<>9_CachedAnonymous
SelectedFileNames : string[]
.ctor() : void
OnClosing(CancelEventArgs) :
FetchGacContents() : void
AddNewEntry(OpenFromGacD
FilterTextBox_TextChanged(ob
ListView_SelectionChanged(ob
OKButton_Click(object, Routed
InitializeComponent() : void
_CreateDelegate(Type, string)
System.Windows.Markup.ICon
<get_SelectedFileNames>b_l
SessionSettings
SmartTextOutputExtensions

.method private hidebysig
instance void FetchGacContents () cil managed
(
// Method begins at RVA 0x5670
// Code size 123 (0x7b)
.maxstack 5
.locals init (
[0] class [System.Core]System.Collections.Generic.HashSet`1<string>
[1] class [Mono.Cecil]Mono.Cecil.AssemblyNameReference
[2] string
[3] class ICSharpCode.ILSpy.OpenFromGacDialog/GacEntry
[4] class [mscorlib]System.Collections.Generic.IEnumerator`1<class [Mono.Cecil]Mono.Cecil.As
)
IL_0000: newobj instance void [System.Core]System.Collections.Generic.HashSet`1<string>::.ctor()
IL_0005: stloc.0
IL_0006: call class [mscorlib]System.Collections.Generic.IEnumerable`1<class [Mono.Cecil]Mono.Ce
IL_000b: callvirt instance class [mscorlib]System.Collections.Generic.IEnumerator`1<!> [mscorlib
IL_0010: stloc.s 4
.try {
IL_0012: br.s IL_0063
// loop start (head: IL_0063)
IL_0014: ldloc.s 4
IL_0016: callvirt instance !0 [mscorlib]System.Collections.Generic.IEnumerator`1<[Mono.C
IL_001b: stloc.1
IL_001c: ldarg.0
IL_001d: volatile.
IL_001f: ldfld modreq([mscorlib]System.Runtime.CompilerServices.IsVolatile) bool class I
IL_0024: brfalse.s IL_0028
IL_0026: leave.s IL_007a
IL_0028: ldloc.0
IL_0029: ldloc.1
IL_002a: callvirt instance string [Mono.Cecil]Mono.Cecil.AssemblyNameReference::get_Full
IL_002f: callvirt instance bool [System.Core]System.Collections.Generic.HashSet`1<string
IL_0034: brfalse.s IL_0063
```

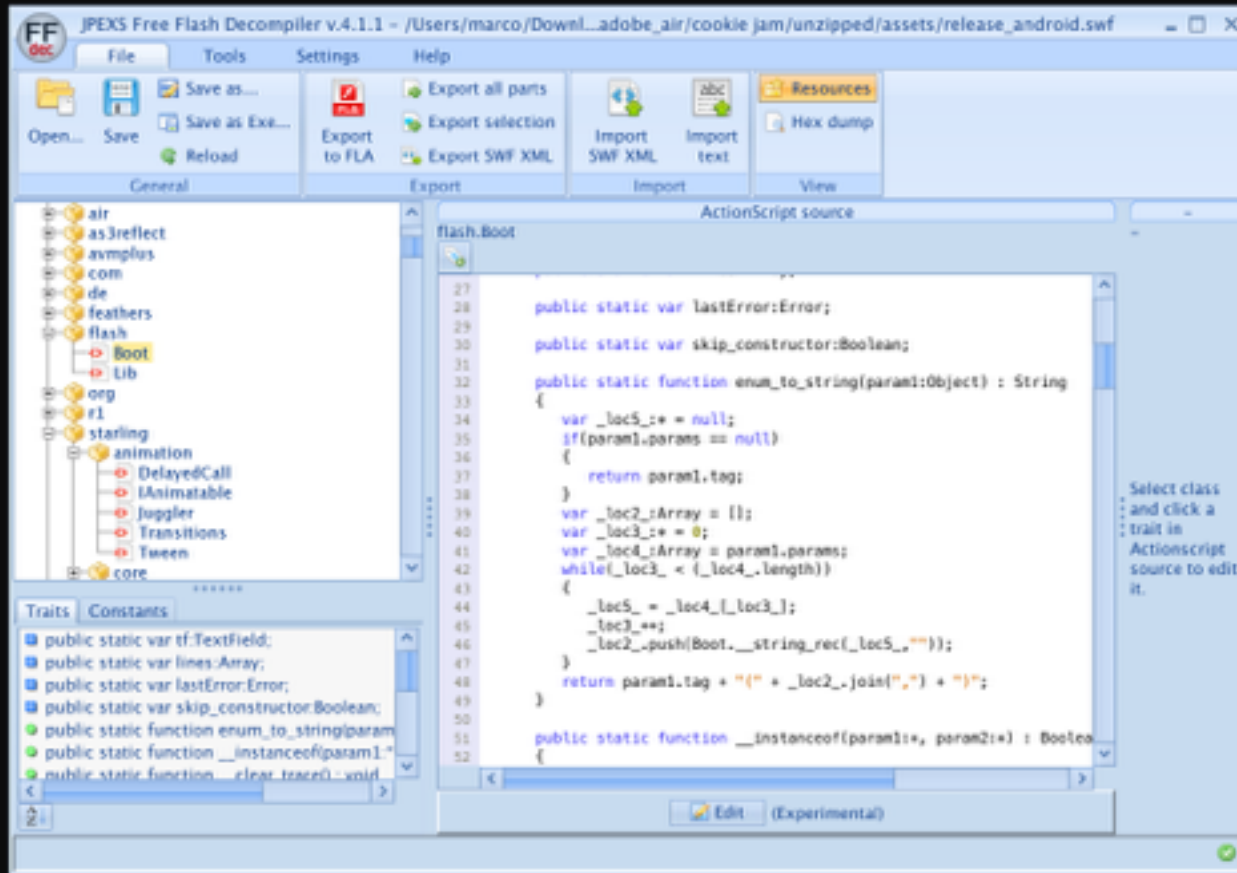


Adobe AIR

- You can find the .swf file in the **assets/** folder on Android
- Trivial to reverse if not obfuscated, some decompilers available



Adobe AIR



Enough with static
analysis attacks..
What about dynamic?
Runtime Manipulation

Runtime attacks, leveraging the shared codebase

- Like we said before, apps using those frameworks share most of the code.
- This fact comes handy also for runtime attacks. We can deploy a runtime attack and use it against all applications that leverage the same framework with little or **without any change!**

Can I have it for free Adobe?

In App items for free!

1. Reverse engineer the Adobe AIR code
2. Spot implementation of in app purchases on Android
3. Verify it's shared between multiple apps
4. Develop a runtime attack to make the purchases appear legitimate when they are not
5. ???
6. PROFIT!

Reversing and spotting the shared code

```
public static boolean verifyPurchase(String arg2, String arg3, String arg4) {  
    boolean v0;  
    if(arg3 == null) {  
        Log.e("IABUtil/Security", "data is null");  
        v0 = false;  
    }  
    else {  
        v0 = true;  
    }  
  
    return v0;  
}
```

- Very well known trivial code copy pasted from Google's examples that can be found in almost all the apps using In app purchases on Google Play

Develop a runtime attack

- Leverage a framework to patch verifyPurchase to return always true and other small mods (with Xposed framework for example)
- if the app doesn't check signatures server side for the purchase (which almost none do) we are done
- For more informations on runtime attacks on iOS or Android: **ZeroNights '14 - Steroids For Your App Security Assessment** - <https://speakerdeck.com/marcograss/steroids-for-your-app-security-assessment>

Some examples of vulnerabilities in the frameworks

A tale about chained vulns in Cordova

- CVE-2014-3500 XAS via Android Intent URLs.
- CVE-2014-3501 Whitelist Bypass for Non-HTTP URLs.
- CVE-2014-3502 Apps can leak data to other apps via URL Loading.

CVE-2014-3500

- Cordova-based applications use a WebView to renders the website requested, via the 'CordovaWebView' activity, through the 'loadUrl()' function.
- If the url provided is 'about:blank' or 'javascript' it will be loaded via 'loadUrlNow(url)' method. Otherwise the register v0 will triggered after a call to 'getProperty("url", null)'.
- In such case, the 'url' parameter is taken from 'getIntent().getStringExtra()', which can be provided externally.

CVE-2014-3500

```
public void loadUrl(String url) {  
    if((url.equals("about:blank")) || (url.startsWith("javascript:"))) {  
        this.loadUrlNow(url);  
    }  
    else {  
        String v0 = this.getProperty("url", null);  
        if(v0 != null && this.urls.size() <= 0) {  
            this.loadUrlIntoView(v0);  
            return;  
        }  
  
        this.loadUrlIntoView(url);  
    }  
}
```

```
public String getProperty(String name, String defaultValue) {  
    Bundle v0 = this.cordova.getActivity().getIntent().getExtras();  
    if(v0 != null) {  
        Object v1 = v0.get(name);  
        if(v1 != null) {  
            defaultValue = v1.toString();  
        }  
    }  
  
    return defaultValue;  
}
```

→ ~ adb shell am start -a android.intent.action.VIEW -c android.intent.category.DEFAULT -e url file:///sdcard/test/test.html -n packageName/.vulnerableActivity

CVE-2014-3500

- Pretty similar to the previous one, the 'errorurl' parameter can be passed via Intent extras in 'CordovaActivity'.
- The 'errorurl' will be rendered by the WebView when a network request fails.
- The parameter's content must either be in the whitelist or be part of URI scheme file. Otherwise it will not be triggered.

CVE-2014-3501

- The framework overrides Android's 'shouldInterceptRequest()' method, to ensure that the WebView only allows requests to URLs in the configured whitelist.
- The mechanism is only configured for HTTP/S or the file URI, being possible to bypass it through other protocols like WS/S

CVE-2014-3502

- Cordova overrides 'shouldOverrideUrlLoading()'. If the scheme is not handled by the function will be launched in the default viewer.
- If an attacker calls the WebView to load a new URL (location.href), 'shouldOverrideUrlLoading()' will be called. Independently of the whitelist validation.
- It's possible to force Cordova to launch a URL using the default viewer.

Adobe AIR's EncryptedLocalStorage API

“The EncryptedLocalStorage class (ELS) provides an encrypted local storage mechanism that you can use as a small cache for an application's private data. ELS data cannot be shared between applications. The intent of ELS is to allow an application to store easily recreated items such as login credentials and other private information.” - Adobe documentation

Adobe AIR WTF

On Android, the data stored by the EncryptedLocalStorage class are not encrypted.

Adobe basically rely on the fact that application private folder are not accessible by an attacker thanks to Android uid/gid separation between apps. As we will see this assumption is not valid.



So basically a developer expects to store data encrypted but....

Adobe AIR WTF

```
public boolean setItem(String arg4, String arg5, byte[] arg6) throws OutOfMemoryError {  
    String v0 = Base64.encodeToString(arg6, 0);  
    SharedPreferences$Editor v1 = AndroidActivityWrapper.GetAndroidActivityWrapper().getActivity()  
        .getApplicationContext().getSharedPreferences(arg4, 0).edit();  
    v1.putString(arg5, v0);  
    return v1.commit();  
}
```



SEEMS LEGIT

Instead the implementation of the “EncryptedStorage” is just a Base64 Encoding!

Probably **lot** of others platform specific quirks around the code base...

Small parenthesis... adb backups

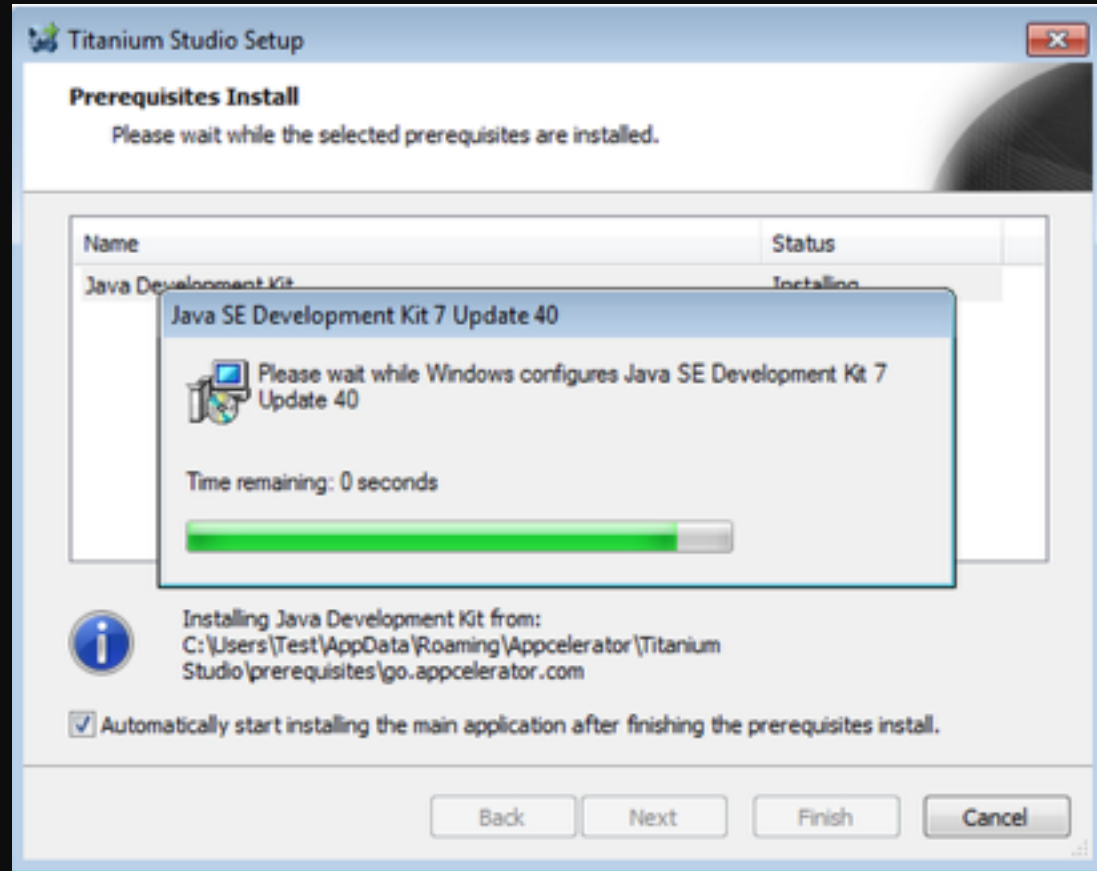
- Functionality introduced in Android 4.0.
- If you have physical access to the phone and you can activate usb debugging, you can backup to your computer the content of the private application folder that have the flag “**allowBackup**” set to true in their AndroidManifest.xml
- If the flag is omitted, the default value is **true**.

Adobe AIR WTF

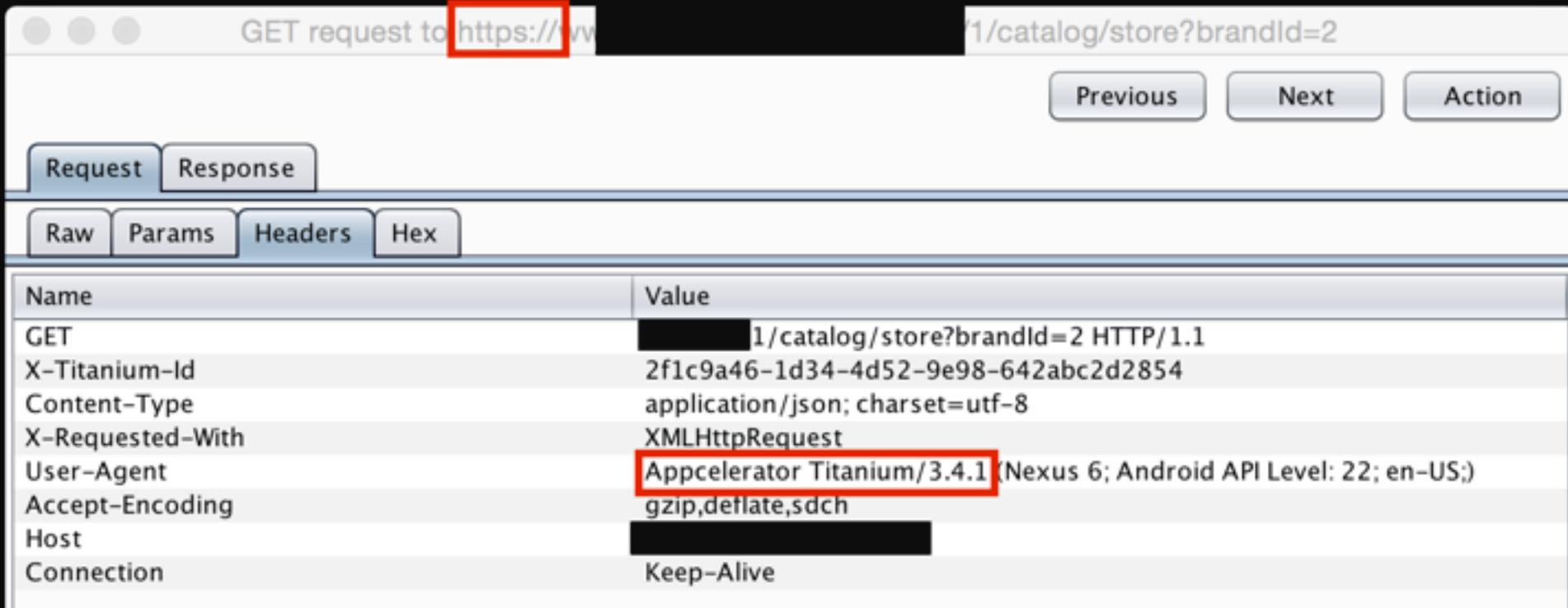
The Adobe AIR applications by default have the flag **allowBackup** unset, so the default value of **true** kicks in.

This allows an attacker without root to backup the content of the private folder of the Adobe application with `adb backup` and retrieve those **unencrypted** supposedly private informations.

What About the Development Tools?



Titanium Broken Default HTTPS



GET request to https:// [redacted] /1/catalog/store?brandId=2

Request Response

Raw Params Headers Hex

Name	Value
GET	[redacted] /1/catalog/store?brandId=2 HTTP/1.1
X-Titanium-Id	2f1c9a46-1d34-4d52-9e98-642abc2d2854
Content-Type	application/json; charset=utf-8
X-Requested-With	XMLHttpRequest
User-Agent	Appcelerator Titanium/3.4.1 (Nexus 6; Android API Level: 22; en-US;)
Accept-Encoding	gzip, deflate, sdch
Host	[redacted]
Connection	Keep-Alive

Who's fault? App Developer, or worst, framework developers?

Titanium Broken Default HTTPS

So the applications developed by this framework are often vulnerable by MiTM. **Why?**

By default the framework **doesn't validate** the certificate on Android!

Titanium Broken Default HTTPS

If the developer doesn't specify a pinning for his certificate, the framework simply leverage dummy classes called “NonValidatingSSLConnectionFactory / NonValidatingTrustManager” and so on.

So the result is that if you rely on the defaults (which pretty much everyone does), the SSL validation is completely skipped, allowing an attacker to MiTM traffic even without a trusted certificate!

Titanium Broken Default HTTPS

```
public NonValidatingSSLSocketFactory() {  
    try {  
        SSLContext context = SSLContext.getInstance("TLS");  
        TrustManager managers[] = new TrustManager[] { new NonValidatingTrustManager() };  
        context.init(null, managers, new SecureRandom());  
        sslFactory = context.getSocketFactory();  
    } catch (Exception e) {  
        Log.e(TAG, e.getMessage(), e);  
    }  
}
```

Titanium Broken Default HTTPS

```
public class NonValidatingTrustManager implements X509TrustManager {  
  
    private static final X509Certificate[] certs = new X509Certificate[0];  
  
    @Override  
    public void checkClientTrusted(X509Certificate[] chain, String authType)  
        throws CertificateException {  
    }  
  
    @Override  
    public void checkServerTrusted(X509Certificate[] chain, String authType)  
        throws CertificateException {  
    }  
  
    @Override  
    public X509Certificate[] getAcceptedIssuers() {  
        return certs;  
    }  
  
}
```



Titanium Broken Default HTTPS

You can use their APIs to pin your own certificate to workaroud this issue, but it will require to pin every single cert you use.

```
var securityManager = https.createX509CertificatePinningSecurityManager([
    {
        url: "https://www.yourorg.com",
        serverCertificate: "yourorg.der"
    }
]);
```

Conclusions

Conclusions

- Security is not taken too much seriously.
- Mechanisms used to protect the application's assets are not good enough. (so your IP is at risk!)
- Few hours and some minion bottles of vodka were necessary to find these issues. What could possibly go wrong if we dedicate more time?

Next Steps

To the infinity and beyond

- Automatic framework recognition and develop instrumentation to trace also interpreted execution.
- Merge all the code extractors in one unique utility.
- Find more vulnerabilities in the framework cores.
- Suggestions?

WE ARE HIRING!



NowSecure™

<https://www.nowsecure.com/careers/>

OR... SAY HI TO US AND WE WILL GRAB A DRINK AND TALK!

THANKS!