Introspy

Security Profiling for Blackbox iOS and Android

Marc Blanchou

Alban Diquet

Introduction – What is it about?

- Tool release: Introspy
  - Security profiler for iOS and Android applications
  - Useful to developers, pen-testers & security researchers

- Security profiling ?
  - Figuring out what an application is doing at runtime
  - Automatically Identifying potentially dangerous behaviors

Introduction – Who are we?

- Three persons worked on this project
  - Tom Daniels – *github/thirstscolr*
  - Marc Blanchou – *github/mblanchou*
  - Alban Diquet – *github/nabla_cod3*
- Security Consultants @ iSEC Partners

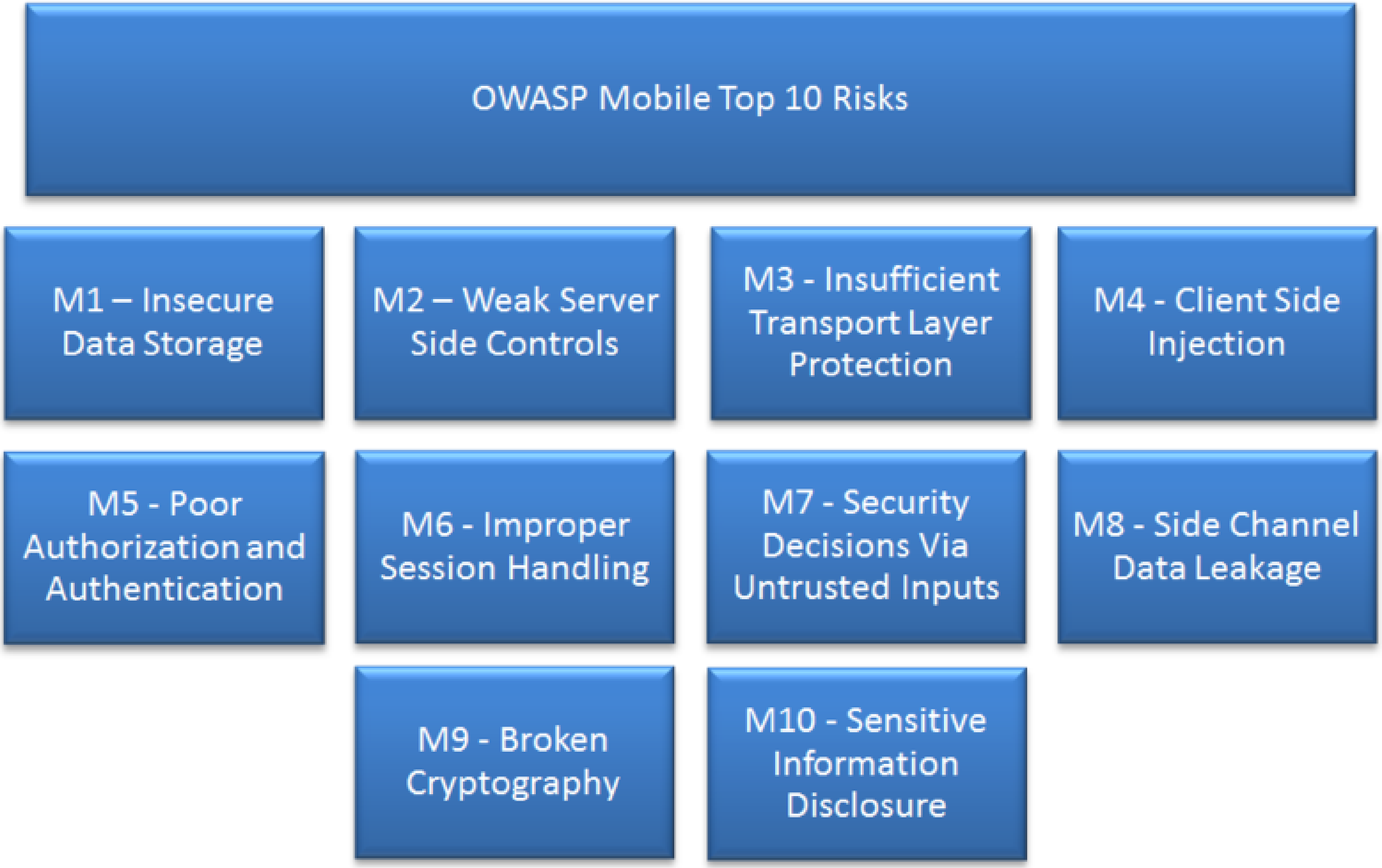
Agenda

- Mobile threats
- Blackbox iOS & Android
- Introspy
- Demo
- Conclusion

Mobile Attack Vectors

- Malicious application running on the device
  - Poorly policed markets
  - Exploits
  - Side-loading
- Active network attacker
  - Wifi or even GSM
- Stolen device

OWASP Mobile Top 10



OWASP Mobile Top 10 Risks

M1 – Insecure Data Storage

M2 – Weak Server Side Controls

M3 - Insufficient Transport Layer Protection

M4 - Client Side Injection

M5 - Poor Authorization and Authentication

M6 - Improper Session Handling

M7 - Security Decisions Via Untrusted Inputs

M8 - Side Channel Data Leakage

M9 - Broken Cryptography

M10 - Sensitive Information Disclosure

Blackbox Testing

- No access to the source code
- Usually time-constrained
- Tester has to:
  - Understand how the app works
  - Understand how it interacts with other components/apps
  - Identify security issues

Blackbox Testing: Methodology

Static analysis: Inspect the application’s binary

- Analyze the binary in a disassembler (IDA)
- iOS
  - Dump encrypted code section (Appstore DRM)
  - Use Mach-O tools: otool, class-dump
- Android
  - Convert Dalvik bytecode to Java bytecode
  - Decompile to Smali or Java
  - Can usually be re-compiled and re-signed with modifications (from Smali code)

Blackbox Testing: Methodology

Dynamic analysis: Run the application on a device

- Monitor inputs / outputs
  - Filesystem, user preferences, keychain
  - IPCs
    - iOS: Pasteboard, URI schemes
    - Android: Activities, Receivers, Content Providers, Services
  - Network: proxy the application's traffic
- Hook functions: MobileSubstrate, CydiaSubstrate
- Debug the application using GDB or JDB
- Bypass jailbreak/root detection

Blackbox Testing: Conclusion

- Lack of automated, security-focused tools on Mobile
  - Debuggers and hooking frameworks are generic
  - Better tools are available on the desktop
- It should be easier than this
  - Most security issues on Mobile are well-known
  - Pen-testing engagements are time-constrained

Introspy

- Security profiler for iOS and Android applications
- Goals
  - Easy to use
  - Help the tester understand what an application is doing at runtime
  - Automatically identify potentially dangerous behaviors

Introspy: How it Works

Introspy is actually comprised of three components:

- Two tracers
  - One for iOS, one for Android
  - Runs on the devices
  - Collects data about functions called by the applications
- An Analyzer
  - Runs on the tester’s computer
  - Partially runs on the device on Android
  - Analyzes data collected by the tracers
  - Creates an HTML report

Introspy: Android & iOS Tracers

- Has to be installed on a jailbroken/rooted device
- Hooks security-sensitive system APIs
  - Logs API calls made by applications
    - Class, method name, arguments and return value
  - Hooks implemented using Cydia/Mobile Substrate
- Stores logged data in a SQLite DB on the device
  - Optionally displays function calls to the console in real-time

Introspy: iOS Tracer

MobileSubstrate

- "de facto framework that allows 3rd party developers to provide runtime patches to system functions"
- Easy to use and very powerful
- Hooks C functions as well as Objective-C methods
- Requires a jailbroken device
- http://iphonedevwiki.net/index.php/MobileSubstrate

Introspy: iOS Tracer

```
/* Example: hooking rand() */
extern SQLiteStorage *traceStorage; // Introspy’s SQLite storage functions
static int (*original_rand)(); // Points to the “original” rand()

// Introspy code to replace rand()
static int replaced_rand() {

   int origResult = original_rand(); // Call the original rand() and store the result
   // Log this function call to the Introspy DB
   CallTracer *tracer = [[CallTracer alloc] initWithClass:@"C" andMethod:@"rand"];
   [tracer addReturnValueFromPlistObject: [NSNumber numberWithUnsignedInt:origResult]];
   [traceStorage saveTracedCall: tracer];
   [tracer release];

   return origResult;
}

MSHookFunction(rand, replaced_rand, (void **) &original_rand); // Hook rand()
```

Introspy: iOS Tracer

Security-Sensitive APIs on iOS ?

- **Crypto:** CCCryptor, CCHmac, CCDigest, rand(), etc.
- **IPCs:** UIPasteboard, URI Handlers
- **File System:** NSData, NSFileHandle, NSFileManager, NSInputStream, etc.
- **User Preferences:** NSUserDefaults
- **Keychain**: SecItemAdd(), SecItemDelete(), etc.
- And more…

Introspy: Android Tracer

Cydia Subtrate

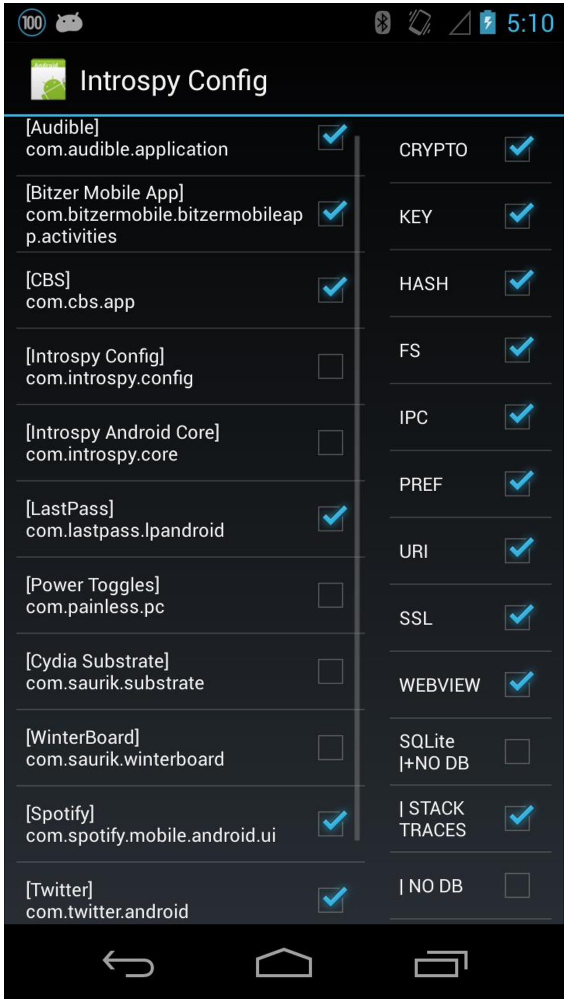
- Supported from Android 2.3 to 4.3
- Same person behind Mobile Substrate on iOS
- Inject code into the Zygote process
- Hook “all” traditional and system apps
- Can also hook native code with a native API (as opposed to Xposed)
- http://www.cydiasubstrate.com/

Introspy: Android Tracer/Analyzer

Security-Sensitive APIs on Android ?

- **Crypto**
  - javax.crypto.Cipher (init, update, dofinal etc.)
  - java.crypto.spec (KeySpec, PBEKeySpec)
  - Etc.
- **IPCs**
  - startService, startActivity, registerReceiver, sendBroadcast, grantUriPermission etc.
  - Programmatic permissions
- **Storage**
  - Files (java.io.File, java.io.FileOutputStream etc.)
  - SQLite APIs
  - Shared (hidden?) preferences, Logs, etc.
- **SSL**
  - Used everywhere? Cert validation?
- **Misc**
  - **Webview** APIs etc.

- Also provides relevant call traces if needed



Introspy Config

[Audible] com.audible.application	☑	CRYPTO	☑
[Bitzer Mobile App] com.bitzermobile.bitzermobileapp.activities	☑	KEY	☑
[CBS] com.cbs.app	☑	HASH	☑
[Introspy Config] com.introspy.config	☐	FS	☑
[Introspy Android Core] com.introspy.core	☐	IPC	☑
[LastPass] com.lastpass.lpandroid	☑	PREF	☑
[Power Toggles] com.painless.pc	☐	URI	☑
[Cydia Substrate] com.saurik.substrate	☐	SSL	☑
[WinterBoard] com.saurik.winterboard	☐	WEBVIEW	☑
[Spotify] com.spotify.mobile.android.ui	☑	SQLite |+NO DB	☐
[Twitter] com.twitter.android	☑	| STACK TRACES	☑
		| NO DB	☐

Introspy: Android Tracer/Analyzer

How to add hooks with Introspy?

- Add a new HookConfig object in a hook list:

```
new HookConfig(
  /* enable hook */      true,

  /* category */         "CRYPTO",
  /* sub-cat.  */        "KEY",

  /* class    */         "javax.crypto.spec.PBEKeySpec",
  /* method */           "PBEKeySpec",

  /* params */           new Class<?>[]{char[].class, byte[].class, Integer.TYPE},
                                   /*  password,    salt,        iteration number */

  /* call handler */     new Intro_CRYPTO_PBEKEY(),

  /* notes  */           "Derive a key from a given password");
```

Introspy: Android Tracer/Analyzer

- Then you just need to create the call handler class
- Extend “IntroHook” and implement an “execute” method

```
// hook for javax.crypto.spec.PBEKeySpec
// PBEKeySpec(password, salt, iterations)

class Intro_CRYPTO_PBEKEY extends IntroHook {
        @Override
        public void execute(Object... args) {
                _logBasicInfo();
                // retrieve parameter the interest us
                int iterationCount = (Integer)args[2];
                // log data:
                _logParameter("Iterations", iterationCount);
                // implement runtime security checks
                // example:
                if (iterationCount < 1000)
                    _logFlush_W(“Low iteration count to generate a key!");
                else
                    _logFlush_I();
}
```

Introspy: Analyzer

- Script running on the tester's computer
- Enumerates and retrieves tracer DBs available on the device
- Analyzes and processes tracer DBs
  - Turns a tracer DB into an HTML report
  - Can also list all files or URLs accessed by the application

Demo

Introspy: Limitations

- It doesn't trace what happens outside of the system APIs
  - Including libraries packaged with the app (such as OpenSSL)
  - We may add hooks to support popular libraries

- It requires a relatively good understanding of the iOS & Android frameworks/APIs
  - Not an autopwn tool

Try it !

- Available on github:
  - https://github.com/iSECPartners/introspy-iOS
  - https://github.com/iSECPartners/Introspy-Android
  - Feedback/suggestions appreciated

- Lots of other pen-testing tools on iSEC Partners' Github
  - Mobile, Web, Network, etc.

There's More…

- SSL cert pinning bypass on Android
https://github.com/iSECPartners/Android-SSL-TrustKiller

- SSL cert pinning bypass on iOS
https://github.com/iSECPartners/ios-ssl-kill-switch

- Cydia Substrate extension for Android to make any application debuggable
https://github.com/iSECPartners/Android-OpenDebug

- Cydia Substrate extension for Android to bypass signature checks:
https://github.com/iSECPartners/Android-KillPermAndSigChecks

Thank You

- Marc Blanchou
  - Principal Security Consultant at iSEC Partners
  - mblanchou@gmail.com

- Alban Diquet
  - Principal Security Consultant at iSEC Partners
  - http://nabla-cod3.github.io
  - alban@isecpartners.com

Questions ?

UK Offices

Manchester - Head Office
Cheltenham
Edinburgh
Leatherhead
London
Thame

European Offices

Amsterdam - Netherlands
Munich – Germany
Zurich - Switzerland

North American Offices

San Francisco
Atlanta
New York
Seattle

Australian Offices

Sydney