Blackbox Android

Breaking "Enterprise Class" Applications and Secure Containers

Marc Blanchou
Mathew Solnik

iSEC PARTNERS

10/13/2011

Agenda

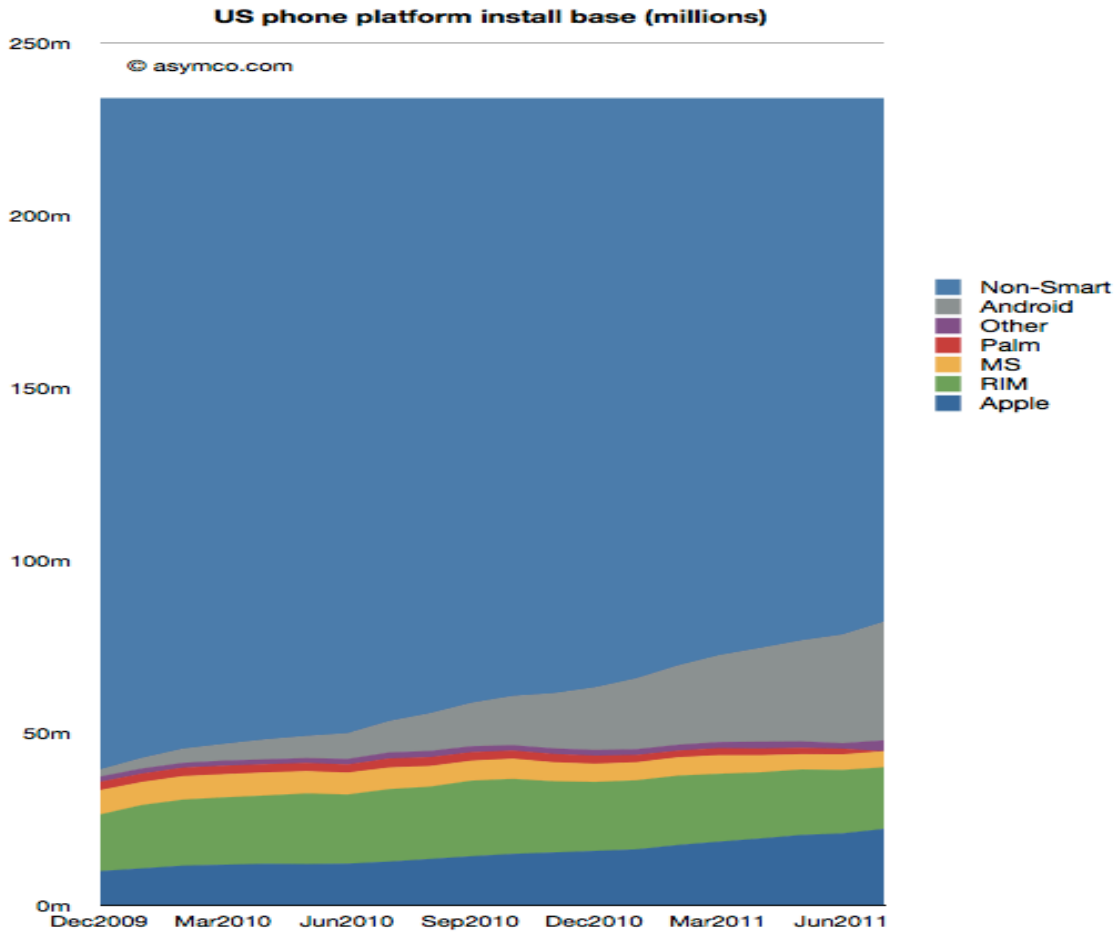
- Background
- Enterprise Class Applications
- Threats to Mobile Data
- Ways to verify we are protected
- Conclusion

Background

- What is this talk about and why?
- Android Overview

What is this talk about and why?

- Android Applications that focus on the Enterprise



iSEC PARTNERS

What is this talk about and why?

- Real world threats and the attack vectors
  - Old data security models may no longer be so relevant
- Reviewing the security marketing claims of Enterprise Class applications
  - How do they compare to current threats and attack vectors?
- Exploring how to verify these claims to determine if you are "secure".

Android Overview

- What is Android?
  - Linux + Java + Google Magic = Android! (sort of)

=

Android Overview

What are a couple of the major differences between Android and Linux?

- Dalvik VM
- Application "Sandboxes": Privilege separation
  - Every Application gets its own UID and GID
  - Permissions are set when applications are installed
- Definitely not all the differences!

Enterprise Class Applications

- Android in the Enterprise
- “Enterprise Class” Applications
- Secure Containers
- Security Marketing Claims

Android in the Enterprise

- Currently focused towards the Consumer Market
  - Lacks central management features
  - Lacks full disk encryption
  - Changing in Android 3.0

- Relies on ActiveSync for Enterprise Mobile Mail
  - Limited security features supported
  - Relies on the Android certificate store if using SSL
  - Emails and data are unencrypted

“Enterprise Class” Applications

- Attempt to improve the security and manageability of the devices
  - Device Analytics
  - Remote Wipe
  - Password Management
  - Application Whitelisting/Blacklisting
  - Credential Offloading
- Communications to and from server protected by SSL
- Take cues from Blackberry Enterprise Server

Secure Containers

- Attempt to provide data segregation
- Encrypt sensitive data such as mail, contacts, and calendar
- Some provide full disk encryption
- Usually protected by a PIN that's separate from the main Android PIN

Security Marketing Claims

- Governments have tested the product and approved it for their **most sensitive deployments**.

- Several security-conscious enterprises have approved the use after **rigorous internal or third party penetration testing**.

- Helps the government agencies comply with DoD Directive 8100.2, Homeland Security Presidential Directive 12 and the Federal Information Security Management Act.

Security Marketing Claims

- Performs remote wipe of enterprise data only

Security Marketing Claims

- Corporate data is **highly secure**
- Over-the-air transmissions and enterprise data at rest on the devices are secured with industry-leading AES-192 encryption
- Leverages a FIPS 140-2 certified cryptographic module to protect data-at-rest and data-in-transit

Security Marketing Claims

What is "industry-leading" encryption?

- Successfully tested by NIST-approved labs and certified to be compliant with FIPS 140-2 Level 1

- Different levels of FIPS
  - Level 1
    - A least one approved algorithm
    - Self contained
    - Approved and reviewed Software module
  - Level 2
    - Show evidence of tampering
  - Level 3
    - Detect and respond to attempts at physical access
  - Level 4
    - Detects fluctuations in environmental conditions

Threats to Mobile Data

- Threats From Who
- What are they after?
- Where does the data reside?
- How they might steal the data and can it be protected

Threats from Who

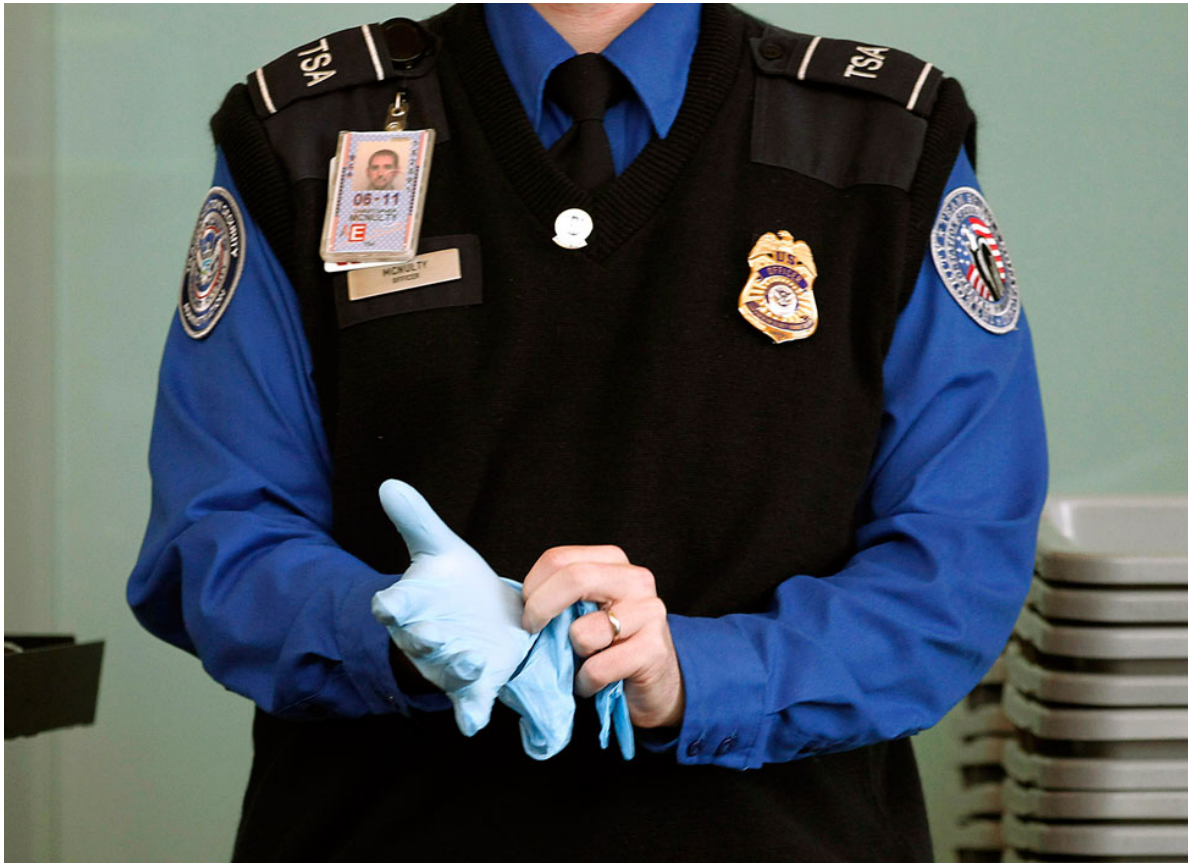
- Malicious Hackers

Threats from Who

- Corporate Espionage

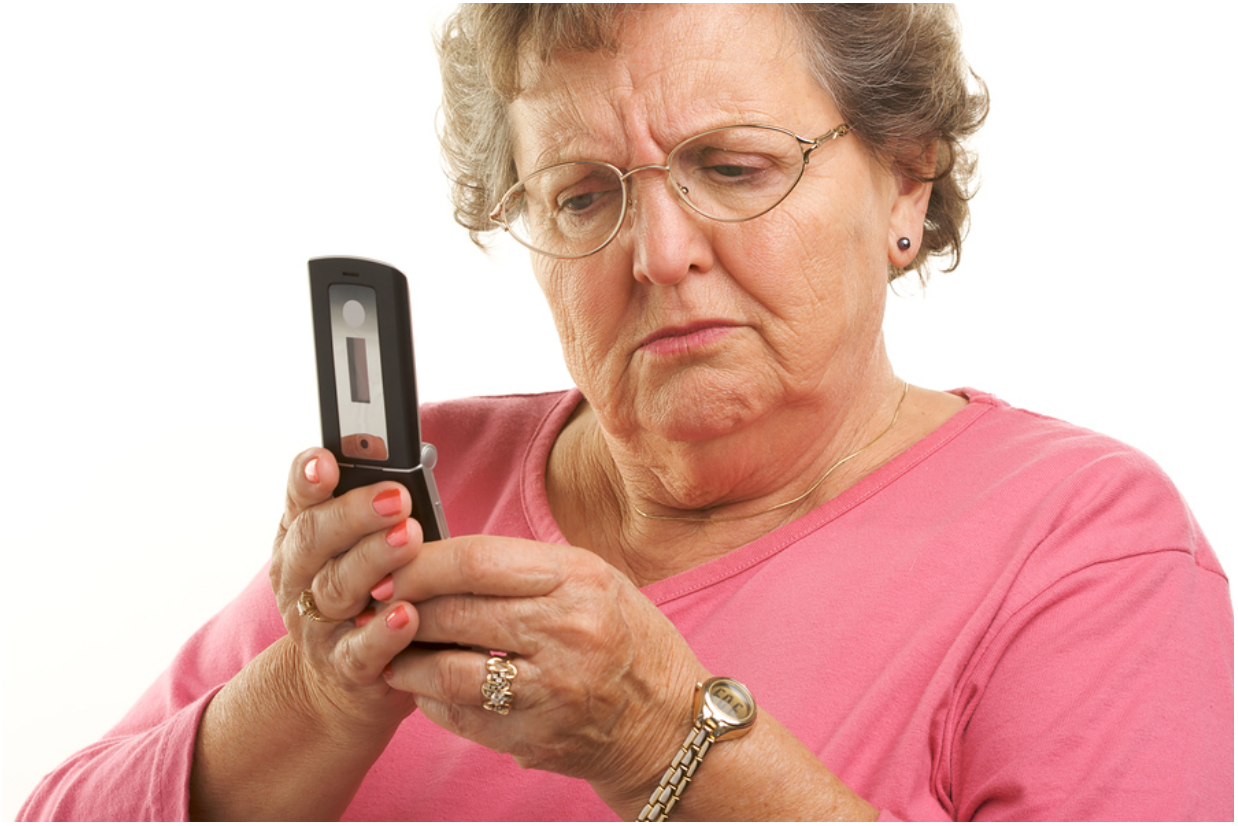
Threats from Who

- Government Entities



Threats from Who

- Your Significant Other



What are they after?

- Emails (including attachments)
- Sensitive Documents and Files
- Credentials
- Text Messages
- Contact Lists
- Call logs
- Calendars

Where does the data reside?

- On the Server
  - Not our focus right now
- In the Air
  - Sensitive data is floating all around you right now
- On the Device
  - Flash
  - RAM

How they might obtain the data

Device is stolen while Powered ON –

- RAM and Flash may be available!
- Ways to obtain the RAM contents
  - Using Android OS
    - USB Debugging Enabled?
      - Root Access
    - Lock Screen bypass
  - Cold Boot Attacks
    - Software based
    - Hardware based
  - JTAG (maybe)

Can the data be protected?

Device is stolen while Powered ON –

- Relevant Marketing Claims
  - "Data is highly secure"
- Can your sensitive data be protected?
  - In some circumstances
    - If nothing sensitive is in RAM
- How might it be properly secured?
  - Sensitive data zeroized

How they might obtain the data

Device is stolen while Powered OFF –

- Only Flash may be available
- Ways to obtain the Flash contents
  - Using Android OS
    - USB Debugging Enabled?
      - Root Access
    - Lock Screen bypass
  - Hardware based
    - Physical flash reading
    - Over USB (UFED[1])

1. UFED - http://www.cellebrite.com/forensic-products/forensic-products/ufed-physical-analyzer-2.html

Can the data be protected?

Device is stolen while Powered OFF –

- Relevant Claims
  - "Data is secure at rest"
- Can your sensitive data be protected?
  - Yes
- How might it be properly secured?
  - Strong password based key derivation
  - Server side key

How they might obtain the data

Device is "borrowed"

- While you are "sleeping" (passed out?)
- Stopped at the border
- Stopped while driving
  - Searching phones is "legal" in some states
  - UFEDs are used by Police
- Similar attack vectors as stolen device
  - Time limited – less likelihood of chip focused attacks
  - BUT device may come back to you (modified)

Can the data be protected?

Device is "borrowed" –

- Can your sensitive data be protected?
  - In some circumstances
    - If nothing sensitive is in RAM
    - If the device is not used afterwards
- How might it be properly secured?
  - Sensitive data zeroized
  - Full Disk Encryption (with separate boot PIN)
  - Tamper detection

How they might obtain the data

Network Based Attacks

- Man in the middle or Passive interception
  - Control of the Cellular Providers
  - Cellular Protocol Vulnerabilities (GSM/WiMAX/etc)
  - Femtocells
  - WIFI
  - SSL
- Baseband Exploits
  - Potential remote root access
  - Potential for persistent backdoor
- Firmware over the Air
  - Push a backdoored firmware

Can the data be protected?

Network Based Attacks – MITM or Passive interception

- Relevant claims
  - Patented end-to-end security
  - AES-192
- Can your sensitive data be protected?
  - Yes
- How?
  - Implement SSL to prevent most attacks
  - Certificate fingerprinting – Don't rely entirely on external root CAs.

How they might obtain the data

Application based attacks

- Malware
  - App Markets
  - Android Vulnerabilities (Browser, PDF, Mail, etc)
  - Userland
    - Contained within Android security boundaries
  - Root
    - Can access RAM and Flash
- Standard application vulnerabilities may also be present
  - Permissions issues
  - Buffer Overflows
  - Misconfigured settings

iSEC PARTNERS

Can the data be protected?

Malware - Userland

- Relevant Claims
  - Corporate and Application data remain separate
- Can your sensitive data be protected?
  - Yes
- How might it be secured?
  - Application white/black listing
  - Granular OS Controls
  - Intent filtering

What data can't be protected from

Anything that obtains higher level permissions then the Secure container

- Malware with Root access
- Baseband exploits
- Firmware over the Air exploits

Verifying if we are protected

- Application Reconnaissance
- Reverse Engineering
- Memory Analysis
- Modifying an Application
- Dynamics Analysis

Application reconnaissance, the APK

How is the application organized

- Android Manifest
- Signature files
- classes.dex
- Libraries
- Resources

Reverse Engineering

Understanding the application's inner workings

- Pull the application off the phone
  - Connect to adb server,
  - Pull off the apk and unzip it
- Convert the DEX file into more readable code
  - Smali
    - Using backsmali to decompile it
    - Can re-compile it with smali
  - Dedexer
  - Java
    - Using dex2jar and JD

Decompiling: From dex to Smali

- Smali

```
.method protected Test()Z
    .registers 3
    .prologue
    iget v0, p0, ###/###/###;->value:I
    const/16 v1, 0x10
    if-ne v0, v1, :cond_8
    const/4 v0, 0x1
    :goto_7
    return v0
    :cond_8
    const/4 v0, 0x0
    goto :goto_7
```

- Java equivalent

```
protected boolean Test() {
    return (this.value == 16 ? true : false);
}
```

Decompiling: From dex to Java

- Smali

```
.method protected Test()Z
    .registers 3
    .prologue
    iget v0, p0, ###/###/###;->value:I
    const/16 v1, 0x10
    if-ne v0, v1, :cond_8
    const/4 v0, 0x1
    :goto_7
    return v0
    :cond_8
    const/4 v0, 0x0
    goto :goto_7
```

- Java equivalent

```
protected boolean Test() {
    return (this.value == 16 ? true : false);
}
```

- From dex to Java
  - With dex2jar and JD

```
protected boolean Test()
{
  if (this.value == 16);
  for (int i = 1; ; i = 0)
    return i;
}
```

iSEC PARTNERS

Differences between DVM and JVM

- Core of Android is Dalvik virtual machine (DVM)
  - Dx (dexer) processes Java .class into Dalvik format
  - One big DEX file (share methods, fields, tables)

- Dalvik is register-based as opposed to JVM
  - Virtual registers
  - Register frames: new set of registers for each method
  - Slightly different opcodes
    - Leading to less code

The difficulty in disassembling DEX

- Known JVM app structure?
  - Multiple constant pools
  - Class definitions
  - Data segment
- Dalvik VM differences
  - Single pool and inlining
  - Different control flow structure
  - Ambiguous types

Converting to Java

- Smali
  - Rather accurate
  - Parameters
  - Type issue
  - Different control flow structure
  - Harder to read
- Java with dex2jar
  - Decompiling is inconsistent
  - Some errors
  - Easily readable for big apps

Converting to Java

- Find common inconsistencies in Java code
  - Examples

```
while (true)
{
  return i;
  i = 0;
  continue;
  byte[] arrayOfByte = computeHash(paramString);
  boolean bool = Arrays.equals(this.passwordHash, arrayOfByte);
}
```

```
public boolean isPwdSet()
{
  if (this.passwordHash != null);
  for (int i = 1; ; i = 0)
    return i;
}
```

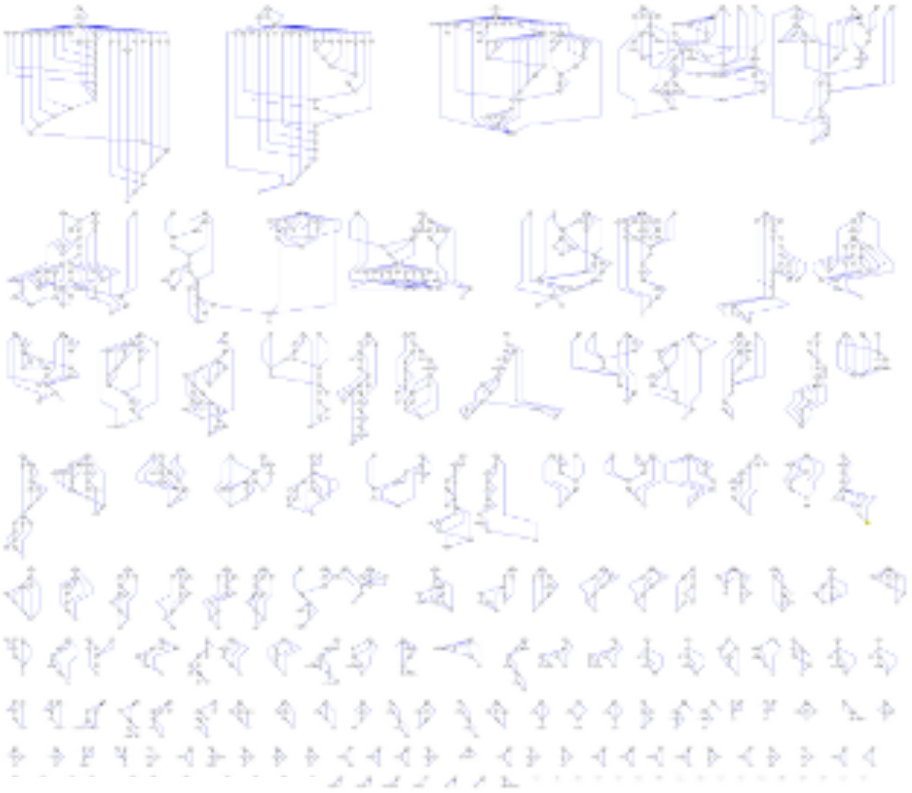
```
for (this.passwordHash = null; !passwordMatch(paramString); this.passwordHas
{
  throw new EncryptionException("password check failed");
  byte[] arrayOfByte1 = *.getRandomKey(8);
  this.passwordSalt = arrayOfByte1;
  arrayOfByte2 = computeHash(paramString);
}
```

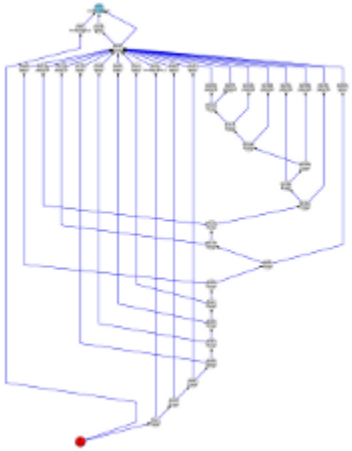
- Replace inconsistencies with Smali code
- Replace parameter names
- Spot most used methods
  - Replace them with converted code

iSEC PARTNERS

Visualizations

- Using Androguard





Verifying if we are protected

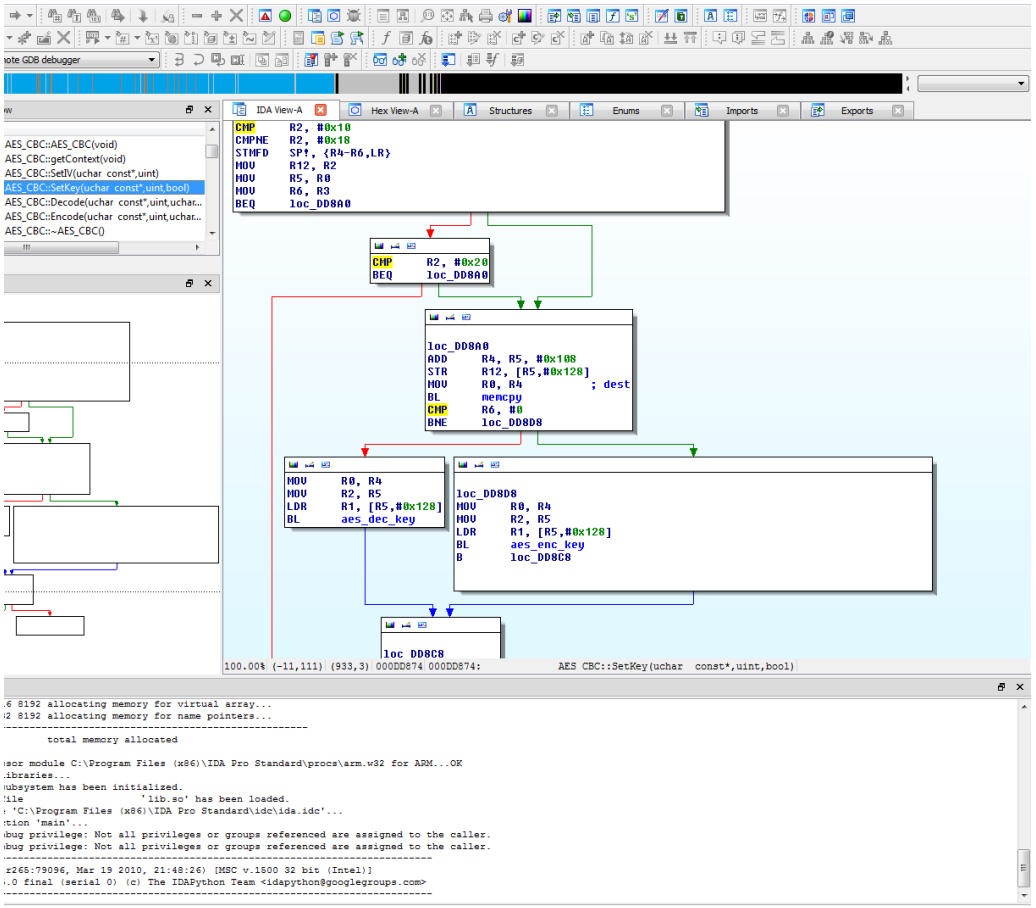
- Protection schema and encryption design
  - Example of issues
- How are cryptographic materials handled
  - Good way to do it, are there still issues to look for?
- Specifics to secure containers
- Appropriate to all attack vectors

Secure container

Reverse Engineering Native Code

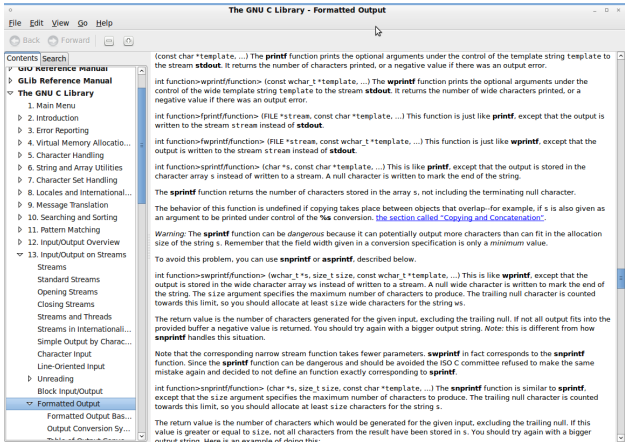
- Using IDA Pro

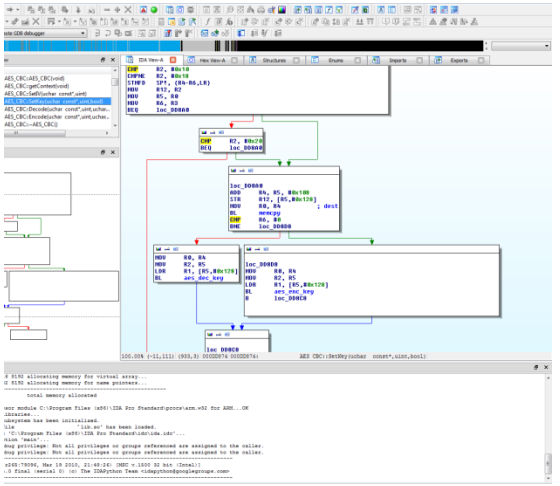
- JNI
- ARM
- THUMB



Native code: What to look for?

- How is data encrypted?
  - More likely to use external libraries
- Documentation
  - Known Open source libraries?
  - Commercial libraries using open-source base?





Memory Analysis

What is available in memory and when

- When device is ON and locked
  - Emails, contacts, etc.
  - Keys, passwords, passwords hashes
- Unstructured
  - String, specifics
- Structured
  - Retrieve structures and objects in memory
  - Java object type, list etc.
  - Follow pointers
- Operating system level
  - Open FD: files, network connections

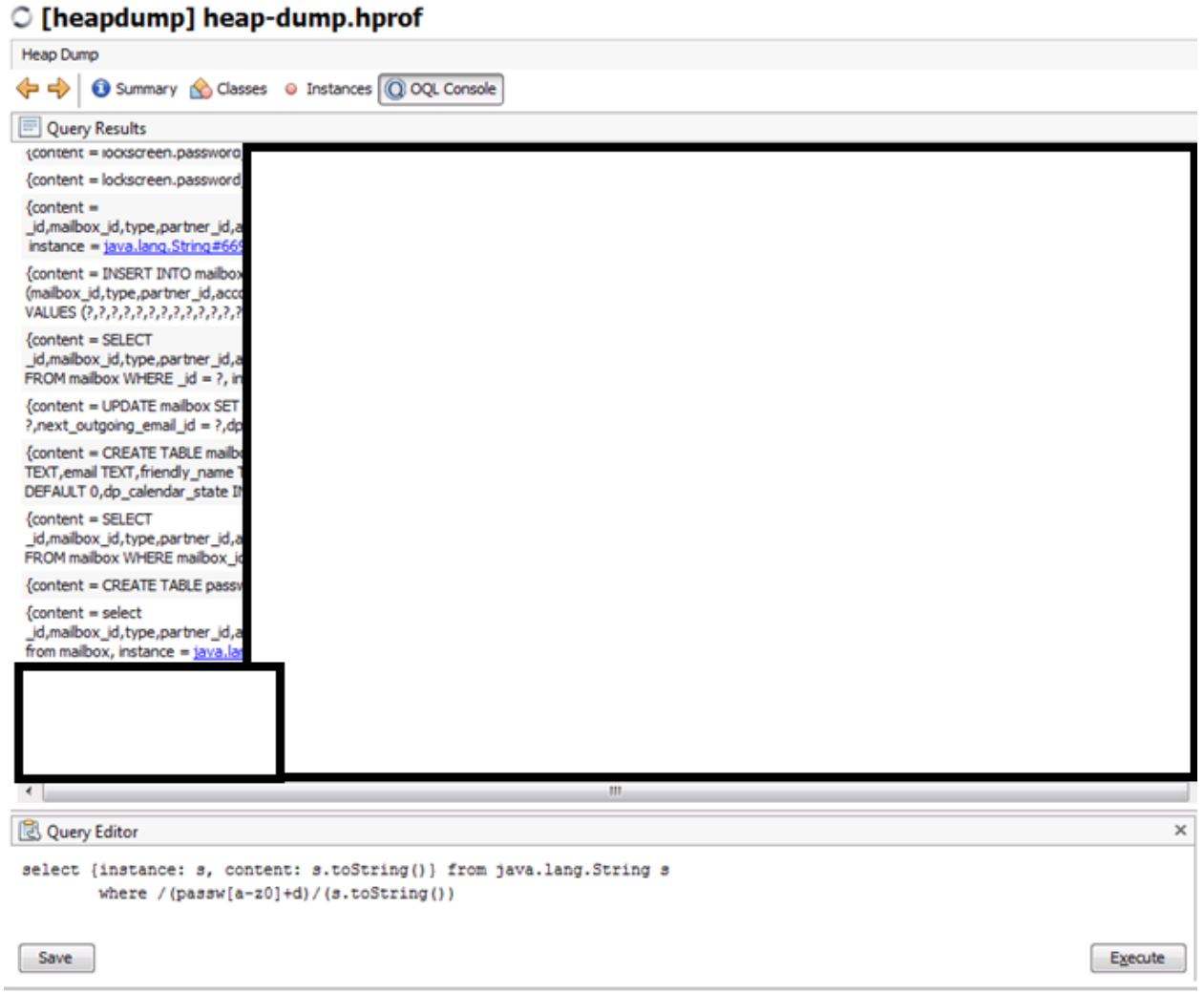
Acquiring memory dump

- Using Linux proc filesystem (procfs)
  - /proc/uid
- Using memfetch
- Get it from the Garbage Collector
  - Send a SIGUSR1 to the process (kill -10 pid)
  - Forcing the GC to dump a hprof
  - Memory dump shows in /data/misc/*.hprof
  - Strip the dump off Dalvik specific data
  - Read it with Java memory dump tools

Reading the memory heap

- Unstructured
  - Looking for a String, part of a key
  - Dalvik strings are UTF-16
- Structured
  - Looking for an object
  - Using JHAT, Jprofiler or VisualVM

Reading the memory heap



Modifying the Application

To help us audit an application and see what an attack can do

- Bypass checks
  - SSL, validate all certificates
  - Specifics
- Bypass very dense obfuscation that would take a while to understand
  - Making obfuscation pointless
- Dump data deobfuscated by the app.
  - Keys, password hash

- Help us debug an application
  - Add debug Logs
  - Add Stacktraces

- Strip SSL if server allows it

Modifying the Application

- Using backsmali then smali
- Apktool
- Java code?
  - Converted to dex with dx then back to smali

- Re-sign it
  - Apk are self-signed
  - Create a key and sign the apk (keytool and jarsigner)

Dynamic Analysis

- Strace
  - Recompiled and available
  - Tracking file reading/writing
    - Why is my application writing my password hash in the system logs?
  - Network IOs

  - Hook on the process
  - When the application starts?
    - Hook on the zygote process and wait for the fork()

- GDB on native parts
- DDMS and Traceview, dmtracedump

Dynamic Analysis

- JDB
  - Not really supported
- JDWP
- AndBug

- Injecting code
  - Java dynamic proxy?
  - Injecting bytecode on the fly?

Conclusion

Be weary of marketing claims

- Claims may be 100% true… But are they relevant?

- Determine what being secure means to you
  - Secure from a random thief? Or from a Government Entity? BIG difference
  - Make your own threat models!
- Ask for penetration test results and threat models!
  - Companies should backup their claims.

Test it yourself!

- You have the tools now.
  - Are things really as secure at they claim?
- Watch that RAM!
  - Insecure RAM usage can lead to many bad things
- Review the crypto!
  - Verify proper key usage
  - Lookout for outdated crypto implementations
  - Check for no salt/IV
- Do not rely on Android protections to keep you safe
  - Lock screen can be bypassed
  - Flash can be read without debugging enabled

Be careful with your sensitive data!

- Nothing is 100% secure
- The more attack vectors the harder something is to secure
- Your phone has a very large threat surface compared to most other devices.

Special Thanks

- iSEC Partners!
- Alex Stamos
- Justine Osborne
- Jesse Burns
- David Thiel
- Paul Youn
- Aaron Grattafiori

iSEC PARTNERS