

MOBILE APP HARDENING

Against Reverse Engineering

Vikas Gupta

Gautam Arvind Pandian

Thales DIS

SINCON 2021, Singapore Virtual

November 2021

■ **Vikas Gupta**

- Security Researcher and Pentester at *Thales DIS*
- M.Sc in Information Security, OSCP certified
- Co-Author contributor for *OWASP Mobile Security Testing Guide (MSTG)*
- Interests: Reverse Engineering, Obfuscation, Crypto
- Github: [@su-vikas](#)

■ **Gautam Arvind Pandian**

- Security Researcher and Architect at *Thales DIS*
- CTF creator in r2con2020 conference - R2Pay.
- Speaker at various conferences - *Android Security Symposium 2020*, *Sincon 2020*
- Interests: Crypto, Hardening Mobile Apps
- Github: [@darvincisec](#)

- **Objective:** Harden mobile apps against RE and without using commercial tools
- What is Mobile App Hardening?
- Various app hardening techniques
 - Build Settings
 - Code Hardening
 - Data
 - Cryptography
- Case Study - R2Pay
- Discussion and Conclusion

1. App Hardening

2. Build Settings
Techniques

3. Code Hardening
Techniques

4. Data Hardening

5. Crypto Hardening

6. Discussion

How to improve the security skills of mobile app developers? Comparing and contrasting expert views

Charles Weir
Security Lancaster
Lancaster University, UK
+44-7876-027350
c.weir1@lancaster.ac.uk

Awais Rashid
Security Lancaster
Lancaster University, UK
+44-1524-510316
a.rashid@lancaster.ac.uk

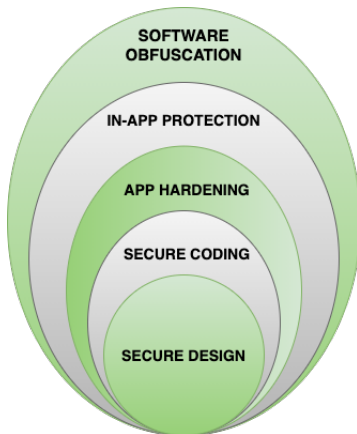
James Noble
Victoria University
Wellington, NZ
+64-4-4635233
kjj@ecs.vuw.ac.nz

Programmers' lack of knowledge and ability in secure development threatens everyone who uses mobile apps. There's no consensus on how to empower app programmers to get that knowledge. Based on interviews with twelve industry experts we argue that the discipline of secure app development is still at an early stage. Only once industry and academia have produced effective app developer motivation and training approaches shall we begin to see the kinds of secure apps we need to combat crime and privacy invasions.

- "We found little existing work about how programmers learn application security."
- In summary, limited resources to motivate and train developers for application security.

- *App hardening* is used in multiple contexts
- Hardening against exploitation
 - Stack Canaries, PIE code
- Self protection for a mobile app
 - RASP, Obfuscation
- Ensure the security of an app even on a hostile or breached OS/device.
- In this presentation, app hardening against reverse engineering is discussed.
 - Against both static and dynamic analysis.

- Mobile app security can be implemented in a multi-layered approach - onion layers.



- Absence of hardening doesn't make an app insecure.

- Threat modeling is necessary to determine depth of hardening needed.
 - If RE is not a risk, then hardening against RE is not required

- Having critical business assets.
 - Protection of IP
 - Sensitive data

- Providing sensitive services
 - Payment
 - Digital Banking
 - Govt services - Digital Identity (e.g: driving licenses)

- There are multitude of commercial tools
 - In-app protections, Code obfuscation, Symbol obfuscation
 - Arxan, Dexguard, Preemptive etc.
- Before using commercial tools, many techniques can be applied by developers.
- Techniques discussed not always present in commercial tools.
- Goal is to slow down RE attacks
- Most ideas are language agnostic, applicable for all platforms
 - Until mentioned otherwise
 - Many native code techniques applicable to ObjC and Swift code.

Static Analysis

- Understanding working of a binary without running it.
- Tools: Jadx, IDA Pro, Ghidra

Dynamic Analysis

- Understanding workings of a binary by executing it.
- Tools: Debuggers, FRIDA, EdXposed

1. App Hardening

2. Build Settings
Techniques

3. Code Hardening
Techniques

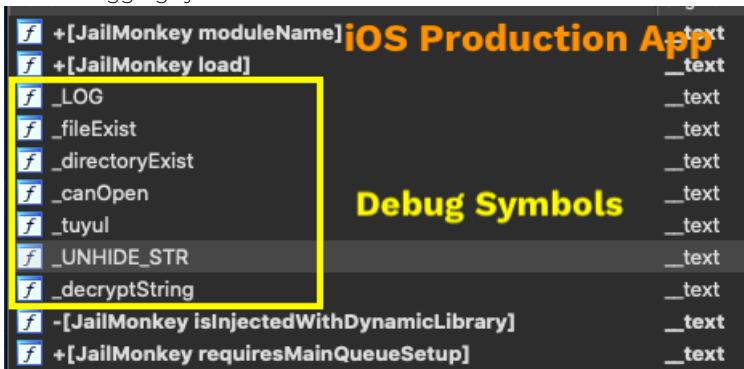
4. Data Hardening

5. Crypto Hardening

6. Discussion

Stripping Symbols

- Remove all the information not needed for execution of the binary
 - Debugging symbols



Stripping Symbols

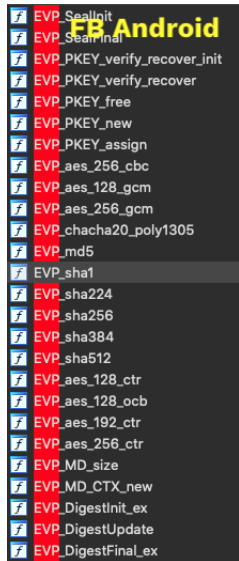
- ELF binary: strip *.comment*, *.strtab*, *.symtab* sections
 - *APKiD* uses information from these sections

```
$ apkid app-release_without_specialstrip.apk      apkid can detect obfuscators if
[+] APKiD 2.1.2 :: from RedNaga :: rednaga.io    not properly stripped
[*] app-release_without_specialstrip.apk!classes.dex
|-> compiler : r8
[+] app-release_without_specialstrip.apk!lib/arm64-v8a/libnative-lib.so
|-> obfuscator : Obfuscator-LLVM version 4.0
[*] app-release_without_specialstrip.apk!lib/armeabi-v7a/libnative-lib.so
|-> obfuscator : obfuscator-LLVM version 4.0
```

```
$ apkid app-release_with_specialstrip.apk
[+] APKiD 2.1.2 :: from RedNaga :: rednaga.io
[*] app-release_with_specialstrip.apk!classes.dex
|-> anti_vm : Build.FINGERPRINT check, Build.MANUFACTURER check
|-> compiler : r8
```

Visibility Hidden Flag

- For native code
- Remove symbols that are private to shared library.
- `-fvisibility=hidden` make **all** symbols hidden by default.
 - Explicitly mark the symbols to be exported by setting visible attribute.



Static Linking Libraries

- Binary merging or having one monolith binary with minimal symbols
- Symbols will be statically linked.
 - Functions called directly using address
 - No exported symbols
- In Android, easier to obfuscate symbols with Proguard
- Only works if a module's source code is available.
 - Or compiled as static lib for native code.

- In Java if app is compiled with *BouncyCastle's* code

```
import d.a.a.c.c;  
import java.nio.ByteBuffer;  
import java.security.MessageDigest;  
import java.security.NoSuchAlgorithmException;  
import javax.crypto.Mac;  
import javax.crypto.spec.SecretKeySpec;  
import me.zhanghai.android.materialprogressbar.BuildConfig;  
  
private static byte[] h(a aVar, byte[] bArr, byte[] bArr2) {  
    String str = "Hmac" + aVar.toString();  
    Mac instance = Mac.getInstance(str);  
    instance.init(new SecretKeySpec(bArr, str));  
    return instance.doFinal(bArr2);  
}
```

Without Static Linking BouncyCastle

```
import d.b.a.d.c;  
import d.b.a.d.d;  
import d.b.a.d.e;  
import d.b.a.d.f;  
import java.nio.ByteBuffer;  
  
private static byte[] h(b bVar, byte[] bArr, byte[] bArr2) {  
    int i = a.f3790a(bVar.ordinal());  
    d.b.a.e.a aVar = i != 1 ? i != 2 ? i != 3 ? null : new d.b.a.e.a(new f()) : new d.b.a.e.a(new e()) : new d.b.a.e.a(new d());  
    aVar.d(new d.b.a.f.a(bArr));  
    aVar.e(bArr2, 0, bArr2.length);  
    byte[] bArr3 = new byte[aVar.c()];  
    aVar.a(bArr3, 0);  
    return bArr3;  
}
```

With Static Linking BouncyCastle

Symbols Obfuscation

The image displays the Android Studio interface with the 'Function name' list on the left and the assembly code view on the right. The function list contains various symbols, including `sub_6CF1AC` through `sub_6D6DD8`. A yellow arrow points from the text 'Obfuscated Func Names' to the list of function names. Another yellow arrow points from the text 'BARCLAYS ANDROID' to the function list. A third yellow arrow points from the text 'Obfuscation too' to the assembly code view.

Function name

- sub_6CF1AC
- sub_6CF1D4
- sub_6CF268
- sub_6CF448
- Java_vcvuysrmbboolalb_dhhhhhd_A9jFNx
- sub_6CFA60
- sub_6D0EA8
- sub_6D0EF8
- Java_vcvuysrmbboolalb_dhhhhhd_RvOaKP
- sub_6D10F0
- sub_6D11B8
- sub_6D11F0
- sub_6D1440
- sub_6D1588
- sub_6D15C0
- sub_6D1688
- sub_6D16A4
- Java_vcvuysrmbboolalb_dhhhhhd_qLEpT_1
- sub_6D17F0
- sub_6D18C8
- sub_6D1908
- sub_6D1924
- sub_6D1A6C
- sub_6D1AC8
- sub_6D5048
- sub_6D511C
- sub_6D516C
- sub_6D5284
- sub_6D5314
- sub_6D5370
- sub_6D5C04
- sub_6D5E0C
- sub_6D5E44
- Java_vcvuysrmbboolalb_dhhhhhd_FYTt0t
- sub_6D5FC4
- sub_6D6DD8

BARCLAYS ANDROID

Obfuscated Func Names

Obfuscation too

Assembly code snippet:

```
; Attributes: bp-based frame
EXPORT Java_vcvuysrmbboolalb_dhhhhhd_qLEpT_1
Java_vcvuysrmbboolalb_dhhhhhd_qLEpT_1
var_74= -0x74
var_70= -0x70
var_20= -0x20
var_20= -0x20
var_10= -0x10
var_s0= 0
; unwind {
SUB SP, #0xC0
STR X21, [SP, #0x80+var_20]
STP X20, X19, [SP, #0x80+var_10]
STP X20, X30, [SP, #0x80+var_s0]
ADD X29, SP, #0x80
MOV X19, X2
MOV X20, X1
MOV X21, X0
MRS X8, TPIDR_EL0
XB, [X8, #0x28]
STR X29, #var_28
WB, [SP, #0x80+var_74]
WB, WB, #0
MOV WB, #3
CSINC WB, WB, WZR, EQ
STR X8, [SP, #0x80+var_70]
ADR X9, sub_6D17F0
MVN X8, X8
ANDS X9, X9, X8
SUB X9, X9, X10
B.EQ loc_6D17E4
```

Assembly code snippet (loc_6D17E4):

```
SUB SP, SP, #0
loc_6D17E8
SUB X9, X9, X10
```

Assembly code snippet (loc_6D17E8):

```
loc_6D17E8
ADD X9, X9, X10
BR X9
```

1. App Hardening

2. Build Settings
Techniques

3. Code Hardening
Techniques

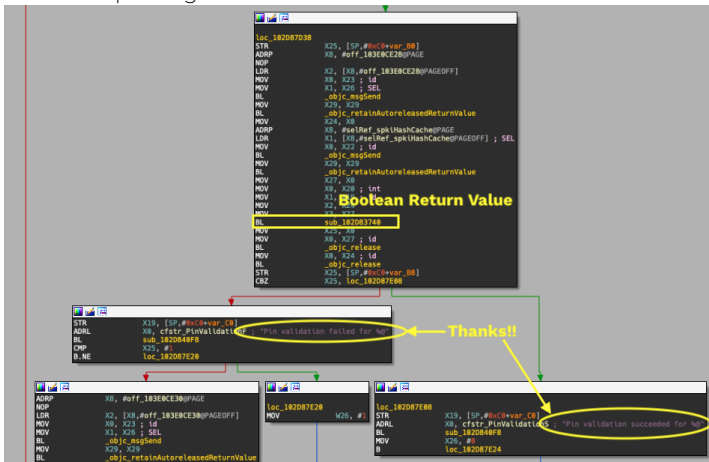
4. Data Hardening

5. Crypto Hardening

6. Discussion

Non-simple Return Types

- To prevent simple hooking or patching
 - Avoid *Boolean* return values
- Paypal iOS has simple boolean return value
 - For SSL pinning check



Non-simple Return Types

- Bypassing with FRIDA in couple of lines of JS

```
Interceptor.attach(  
    ObjC.classes.IRoot['- jailbroken'].implementation,  
    {  
        onLeave: function(retval){  
            retval.replace(0x0);  
        }  
    });
```

Anti-Replay Return Values

- Both, non-simple return type and anti-replay protection

General registers

X0 6C4342D1F3386A1D

X1 000000760657A800

X2 0000007606400000

X3 0000000000000008

X4 0000000000000017A

X5 0000007604090808

X6 000000760657A700

X7 000000760657A680

X8 6DBA8645DA15059F

X9 6DBA8645DA15059F

X10 0000000000000017A

X11 00000000000000000

X12 000000760657A300

X13 000000760657A080

X14 00000000FFFFFFF

X15 0000000000000004

X16 00000076A35D3C48

X17 00000076A357D4B8

X18 0000000068DF4E1

X19 0000000000000000

X20 00000076182E7100

X21 00000076182E7080

X22 00000000FFFFFFF

X23 0000000000000004

X24 00000076A35D3C48

X25 00000076A357D4B8

X26 0000000068DF4E1

X27 0000000000000000

X28 0000007606431000

X29 6DBA8645DA15059F

X30 6DBA8645DA15059F

X31 0000000000000000

X32 000000760657A080

X33 000000760657A680

X34 00000000FFFFFFF

X35 0000000000000004

X36 00000076A35D3C48

X37 00000076A357D4B8

X38 0000000068DF4E1

X39 0000000000000000

X40 0000007606431000

X41 6DBA8645DA15059F

X42 6DBA8645DA15059F

X43 0000000000000000

X44 000000760657A080

X45 000000760657A680

X46 00000000FFFFFFF

X47 0000000000000004

X48 00000076A35D3C48

X49 00000076A357D4B8

X50 0000000068DF4E1

X51 0000000000000000

X52 0000007606431000

X53 6DBA8645DA15059F

X54 6DBA8645DA15059F

X55 0000000000000000

X56 000000760657A080

X57 000000760657A680

X58 00000000FFFFFFF

X59 0000000000000004

X60 00000076A35D3C48

X61 00000076A357D4B8

X62 0000000068DF4E1

X63 0000000000000000

X64 0000007606431000

X65 6DBA8645DA15059F

X66 6DBA8645DA15059F

X67 0000000000000000

X68 000000760657A080

X69 000000760657A680

X70 00000000FFFFFFF

X71 0000000000000004

X72 00000076A35D3C48

X73 00000076A357D4B8

X74 0000000068DF4E1

X75 0000000000000000

X76 0000007606431000

X77 6DBA8645DA15059F

X78 6DBA8645DA15059F

X79 0000000000000000

X80 000000760657A080

X81 000000760657A680

X82 00000000FFFFFFF

X83 0000000000000004

X84 00000076A35D3C48

X85 00000076A357D4B8

X86 0000000068DF4E1

X87 0000000000000000

X88 0000007606431000

X89 6DBA8645DA15059F

X90 6DBA8645DA15059F

X91 0000000000000000

X92 000000760657A080

X93 000000760657A680

X94 00000000FFFFFFF

X95 0000000000000004

X96 00000076A35D3C48

X97 00000076A357D4B8

X98 0000000068DF4E1

X99 0000000000000000

X100 0000007606431000

X101 6DBA8645DA15059F

X102 6DBA8645DA15059F

X103 0000000000000000

X104 000000760657A080

X105 000000760657A680

X106 00000000FFFFFFF

X107 0000000000000004

X108 00000076A35D3C48

X109 00000076A357D4B8

X110 0000000068DF4E1

X111 0000000000000000

X112 0000007606431000

X113 6DBA8645DA15059F

X114 6DBA8645DA15059F

X115 0000000000000000

X116 000000760657A080

X117 000000760657A680

X118 00000000FFFFFFF

X119 0000000000000004

X120 00000076A35D3C48

X121 00000076A357D4B8

X122 0000000068DF4E1

X123 0000000000000000

X124 0000007606431000

X125 6DBA8645DA15059F

X126 6DBA8645DA15059F

X127 0000000000000000

X128 000000760657A080

X129 000000760657A680

X130 00000000FFFFFFF

X131 0000000000000004

X132 00000076A35D3C48

X133 00000076A357D4B8

X134 0000000068DF4E1

X135 0000000000000000

X136 0000007606431000

X137 6DBA8645DA15059F

X138 6DBA8645DA15059F

X139 0000000000000000

X140 000000760657A080

X141 000000760657A680

X142 00000000FFFFFFF

X143 0000000000000004

X144 00000076A35D3C48

X145 00000076A357D4B8

X146 0000000068DF4E1

X147 0000000000000000

X148 0000007606431000

X149 6DBA8645DA15059F

X150 6DBA8645DA15059F

X151 0000000000000000

X152 000000760657A080

X153 000000760657A680

X154 00000000FFFFFFF

X155 0000000000000004

X156 00000076A35D3C48

X157 00000076A357D4B8

X158 0000000068DF4E1

X159 0000000000000000

X160 0000007606431000

X161 6DBA8645DA15059F

X162 6DBA8645DA15059F

X163 0000000000000000

X164 000000760657A080

X165 000000760657A680

X166 00000000FFFFFFF

X167 0000000000000004

X168 00000076A35D3C48

X169 00000076A357D4B8

X170 0000000068DF4E1

X171 0000000000000000

X172 0000007606431000

X173 6DBA8645DA15059F

X174 6DBA8645DA15059F

X175 0000000000000000

X176 000000760657A080

X177 000000760657A680

X178 00000000FFFFFFF

X179 0000000000000004

X180 00000076A35D3C48

X181 00000076A357D4B8

X182 0000000068DF4E1

X183 0000000000000000

X184 0000007606431000

X185 6DBA8645DA15059F

X186 6DBA8645DA15059F

X187 0000000000000000

X188 000000760657A080

X189 000000760657A680

X190 00000000FFFFFFF

X191 0000000000000004

X192 00000076A35D3C48

X193 00000076A357D4B8

X194 0000000068DF4E1

X195 0000000000000000

X196 0000007606431000

X197 6DBA8645DA15059F

X198 6DBA8645DA15059F

X199 0000000000000000

X200 000000760657A080

X201 000000760657A680

X202 00000000FFFFFFF

X203 0000000000000004

X204 00000076A35D3C48

X205 00000076A357D4B8

X206 0000000068DF4E1

X207 0000000000000000

X208 0000007606431000

X209 6DBA8645DA15059F

X210 6DBA8645DA15059F

X211 0000000000000000

X212 000000760657A080

X213 000000760657A680

X214 00000000FFFFFFF

X215 0000000000000004

X216 00000076A35D3C48

X217 00000076A357D4B8

X218 0000000068DF4E1

X219 0000000000000000

X220 0000007606431000

X221 6DBA8645DA15059F

X222 6DBA8645DA15059F

X223 0000000000000000

X224 000000760657A080

X225 000000760657A680

X226 00000000FFFFFFF

X227 0000000000000004

X228 00000076A35D3C48

X229 00000076A357D4B8

X230 0000000068DF4E1

X231 0000000000000000

X232 0000007606431000

X233 6DBA8645DA15059F

X234 6DBA8645DA15059F

X235 0000000000000000

X236 000000760657A080

X237 000000760657A680

X238 00000000FFFFFFF

X239 0000000000000004

X240 00000076A35D3C48

X241 00000076A357D4B8

X242 0000000068DF4E1

X243 0000000000000000

X244 0000007606431000

X245 6DBA8645DA15059F

X246 6DBA8645DA15059F

X247 0000000000000000

X248 000000760657A080

X249 000000760657A680

X250 00000000FFFFFFF

X251 0000000000000004

X252 00000076A35D3C48

X253 00000076A357D4B8

X254 0000000068DF4E1

X255 0000000000000000

X256 0000007606431000

X257 6DBA8645DA15059F

X258 6DBA8645DA15059F

X259 0000000000000000

X260 000000760657A080

X261 000000760657A680

X262 00000000FFFFFFF

X263 0000000000000004

X264 00000076A35D3C48

X265 00000076A357D4B8

X266 0000000068DF4E1

X267 0000000000000000

X268 0000007606431000

X269 6DBA8645DA15059F

X270 6DBA8645DA15059F

X271 0000000000000000

X272 000000760657A080

X273 000000760657A680

X274 00000000FFFFFFF

X275 0000000000000004

X276 00000076A35D3C48

X277 00000076A357D4B8

X278 0000000068DF4E1

X279 0000000000000000

X280 0000007606431000

X281 6DBA8645DA15059F

X282 6DBA8645DA15059F

X283 0000000000000000

X284 000000760657A080

X285 000000760657A680

X286 00000000FFFFFFF

X287 0000000000000004

X288 00000076A35D3C48

X289 00000076A357D4B8

X290 0000000068DF4E1

X291 0000000000000000

X292 0000007606431000

X293 6DBA8645DA15059F

X294 6DBA8645DA15059F

X295 0000000000000000

X296 000000760657A080

X297 000000760657A680

X298 00000000FFFFFFF

X299 0000000000000004

X300 00000076A35D3C48

X301 00000076A357D4B8

X302 0000000068DF4E1

X303 0000000000000000

X304 0000007606431000

X305 6DBA8645DA15059F

X306 6DBA8645DA15059F

X307 0000000000000000

X308 000000760657A080

X309 000000760657A680

X310 00000000FFFFFFF

X311 0000000000000004

X312 00000076A35D3C48

X313 00000076A357D4B8

X314 0000000068DF4E1

X315 0000000000000000

X316 0000007606431000

X317 6DBA8645DA15059F

X318 6DBA8645DA15059F

X319 0000000000000000

X320 000000760657A080

X321 000000760657A680

X322 00000000FFFFFFF

X323 0000000000000004

X324 00000076A35D3C48

X325 00000076A357D4B8

X326 0000000068DF4E1

X327 0000000000000000

X328 0000007606431000

X329 6DBA8645DA15059F

X330 6DBA8645DA15059F

X331 0000000000000000

X332 000000760657A080

X333 000000760657A680

X334 00000000FFFFFFF

X335 0000000000000004

X336 00000076A35D3C48

X337 00000076A357D4B8

X338 0000000068DF4E1

X339 0000000000000000

X340 0000007606431000

X341 6DBA8645DA15059F

X342 6DBA8645DA15059F

X343 0000000000000000

X344 000000760657A080

X345 000000760657A680

X346 00000000FFFFFFF

X347 0000000000000004

X348 00000076A35D3C48

X349 00000076A357D4B8

X350 0000000068DF4E1

X351 0000000000000000

X352 0000007606431000

X353 6DBA8645DA15059F

X354 6DBA8645DA15059F

X355 0000000000000000

X356 000000760657A080

X357 000000760657A680

X358 00000000FFFFFFF

X359 0000000000000004

X360 00000076A35D3C48

X361 00000076A357D4B8

X362 0000000068DF4E1

X363 0000000000000000

X364 0000007606431000

X365 6DBA8645DA15059F

X366 6DBA8645DA15059F

X367 0000000000000000

X368 000000760657A080

X369 000000760657A680

X370 00000000FFFFFFF

X371 0000000000000004

X372 00000076A35D3C48

X373 00000076A357D4B8

X374 0000000068DF4E1

X375 0000000000000000

X376 0000007606431000

X377 6DBA8645DA15059F

X378 6DBA8645DA15059F

X379 0000000000000000

X380 000000760657A080

X381 000000760657A680

X382 00000000FFFFFFF

X383 0000000000000004

X384 00000076A35D3C48

X385 00000076A357D4B8

X386 0000000068DF4E1

X387 0000000000000000

X388 0000007606431000

X389 6DBA8645DA15059F

X390 6DBA8645DA15059F

X391 0000000000000000

X392 000000760657A080

X393 000000760657A680

X394 00000000FFFFFFF

X395 0000000000000004

X396 00000076A35D3C48

X397 00000076A357D4B8

X398 0000000068DF4E1

X399 0000000000000000

X400 0000007606431000

X401 6DBA8645DA15059F

X402 6DBA8645DA15059F

X403 0000000000000000

X404 000000760657A080

X405 000000760657A680

X406 00000000FFFFFFF

X407 0000000000000004

X408 00000076A35D3C48

X409 00000076A357D4B8

X410 0000000068DF4E1

X411 0000000000000000

X412 0000007606431000

X413 6DBA8645DA15059F

X414 6DBA8645DA15059F

X415 0000000000000000

X416 000000760657A080

X417 000000760657A680

X418 00000000FFFFFFF

X419 0000000000000004

X420 00000076A35D3C48

X421 00000076A357D4B8

X422 0000000068DF4E1

X423 0000000000000000

X424 0000007606431000

X425 6DBA8645DA15059F

X426 6DBA8645DA15059F

X427 0000000000000000

X428 000000760657A080

X429 000000760657A680

X430 00000000FFFFFFF

X431 0000000000000004

X432 00000076A35D3C48

X433 00000076A357D4B8

X434 0000000068DF4E1

X435 0000000000000000

X436 0000007606431000

X437 6DBA8645DA15059F

X438 6DBA8645DA15059F

X439 0000000000000000

X440 000000760657A080

X441 000000760657A680

X442 00000000FFFFFFF

X443 0000000000000004

X444 00000076A35D3C48

X445 00000076A357D4B8

X446 0000000068DF4E1

X447 0000000000000000

X448 0000007606431000

X449 6DBA8645DA15059F

X450 6DBA8645DA15059F

X451 0000000000000000

X452 000000760657A080

X453 000000760657A680

X454 00000000FFFFFFF

X455 0000000000000004

X456 00000076A35D3C48

X457 00000076A357D4B8

X458 0000000068DF4E1

X459 0000000000000000

X460 0000007606431000

X461 6DBA8645DA15059F

X462 6DBA8645DA15059F

X463 0000000000000000

X464 000000760657A080

X465 000000760657A680

X466 00000000FFFFFFF

X467 0000000000000004

X468 00000076A35D3C48

X469 00000076A357D4B8

X470 0000000068DF4E1

X471 0000000000000000

X472 0000007606431000

X473 6DBA8645DA15059F

X474 6DBA8645DA15059F

X475 0000000000000000

X476 000000760657A080

X477 000000760657A680

X478 00000000FFFFFFF

X479 0000000000000004

X480 00000076A35D3C48

X481 00000076A357D4B8

X482 0000000068DF4E1

X483 0000000000000000

X484 0000007606431000

X485 6DBA8645DA15059F

X486 6DB

System APIs

- Avoid using libc except for memory management
 - replace libc calls with open source memcpy, memset, string functions and **inline** them
 - replace file operations with **syscalls** for critical operations

```
__attribute__((always_inline))
static inline
void* my_memset(void* dst, int c, size_t n)
{
    char* q = (char*)dst;
    char* end = q + n;
    for (;;) {
        if (q >= end) break; *q++ = (char) c;
        if (q >= end) break; *q++ = (char) c;
        if (q >= end) break; *q++ = (char) c;
        if (q >= end) break; *q++ = (char) c;
    }
    return dst;
}

__attribute__((always_inline))
static inline int
my_strcmp(const char *s1, const char *s2)
{
    while (*s1 == *s2++)
        if (*s1++ == 0)
            return (0);
    return (*(unsigned char *)s1 - *(unsigned char *)--s2);
}
```

Custom impl of libc

```
__attribute__((always_inline))
static inline int my_openat(int __dir_fd, const void* __path, int __flags, int __mode) {
    return (int) __syscall4(__NR_openat, __dir_fd, (long) __path, __flags, __mode);
}

__attribute__((always_inline))
static inline ssize_t my_read(int __fd, void* __buf, size_t __count) {
    return __syscall3(__NR_read, __fd, (long) __buf, (long) __count);
}

__attribute__((always_inline))
static inline int my_close(int __fd) {
    return (int) __syscall1(__NR_close, __fd);
}
```

syscall alternative for libc file operations

System APIs

- Use reflection for accessing Java System APIs such as Keystore.
 - further requires string obfuscation

String Data Structures

- Don't use immutable data structures for storing critical values
 - Memory cannot be zero'd
- Avoid **Strings** in Java, **NSData/NSString** in ObjC
 - Need to be dependent on GC or ref counting for variable to be destroyed
- Use byte arrays, as its easy to manipulate the lifetime.
 - XOR masking when kept in memory
 - Zero it when no more used

Reduce Variables Memory Lifetime

- Critical values wiped with 0s or random before *free-ing* the memory
- Example: passwords, keys

Mask Sensitive Values when in Memory

- XOR mask the values.
- Unmask when to be used and mask them back again.
- Example: crypto keys

1. App Hardening

2. Build Settings
Techniques

3. Code Hardening
Techniques

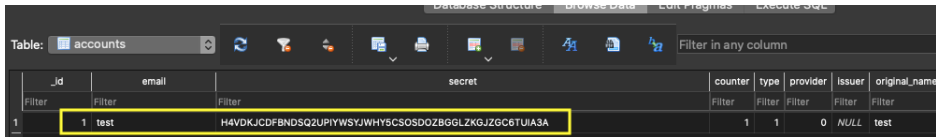
4. Data Hardening

5. Crypto Hardening

6. Discussion

- Data at rest gives juicy information whether any sensitive or personal info is stored even if encrypted
- A malware or root application can dump the sandbox data and analyze the contents
- While protecting data at rest, assume sandbox is broken

- Famous OTP apps such as Google Authenticator, Microsoft Authenticator stores sensitive data in clear.
- **Google Authenticator** SQLite DB.



_id	email	secret	counter	type	provider	issuer	original_name
1	test	H4VDKJCDFBNDsq2UPIYWSYJWHY5CSOSDOZBGGLZKGJZGC6TUIA3A	1	1	0	NULL	test

- *counter* and *secret* for HOTP algo in clear.
- Column names very descriptive as well.

Obfuscated database						
_id	_a	_b	_d	_e	_f	_g
Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	33d1e99028fc25b7	3a0e3a911aa...	b54406d42ebaa455cdbabc1f723524b7	b54406d42ebaa455cdbab...	ea94f189599658...	da34c0e2-1692-4666-aa9f-

- Security sensitive apps use certificate pinning to ensure the app is communicating with intended server
 - Easy to bypass certificate pinning in the absence of RASP
 - If Certificate pinning is bypassed, all data in transit is exposed
- Strengthening the obfuscation of data in transit
 - Obscure key names in JSON payload
 - Create a tunnel on top of TLS
 - Provisioning crypto keys, user sensitive information (credit card number) requires high protection

```
"adID":"4490a706c1735198",
"adrIMEI":"4490a706c1735198",
"adrIMSI":"","
"adrMEID":"","
"adrSERIAL":"unknown",
"adrUDID":"76d85db4-a9ce-42b5-8114-c0112b97260b",
"advertisingID":"0979449c-e32f-49e1-86bd-2bb538703dca",
"applicationVersion":"5.170.0(51700000) Build ; Build 27844510",
"challengeID":"090eae8b-5c07-4db2-bd69-df9c67ec6fb",
"cli":"","
"countryCode":"in",
"deviceManufacturer":"Google",
"deviceModel":"Pixel",
"iosUDID":"","
"latitude":0.0,
"longitude":0.0,
"otp":"356512",
"phoneNumber":"+919845040726",
"source":"android",
"sourceID":"","
"tmSessionID":"","
"tpToken":{"
"type":"Google",
"value":"eyJhbGciOiJSUzI1NiIsImtpZCI6ImFrZWdhbGVWInJleSU0TUONGFmNTNaOTM3MTJhNjRlMaUyYmaYmY5NDMzMmNTIiLCJOeXAiOiA1KXVlQ1fQ..eyJpc3MiOiJodHRwczovL2FjY2S1bnRsLmdvbCdsZS5jbC01LkY0Y0Y0WqOjJC2Sk5Iiwic1wiOiJldmVyc2SiImh0dHB0eS2KXvZXZjbjB2SO2W50LmhvdS9hLOFBVFhBndqO0dLSVp5SV9VRopBTf4DLmJTY1ZFBGPdPmKEZINDXTWSkYattPXM5nNiljIiwic2IiZW50ZShhcPrvsVd-IspahyBQPwlfs3ZP4gbRJzmHQ4xcwXPg3j4NbYco17QPIJjarfp0aEdsu2FFaCaQBHbVTILDRLpvc2SskTIPHhmZVO45IirSh08_e_MF5AnvnLGA"
```

1. App Hardening

2. Build Settings
Techniques

3. Code Hardening
Techniques

4. Data Hardening

5. Crypto Hardening

6. Discussion

Inline sensitive functions

- Inline sensitive functions
 - No separate function in final binary
 - Xref analysis won't work either
- Compiler directive `__attribute__((always_inline))` to inline sensitive functions in C

System APIs

- Avoid using system APIs for crypto
 - Leaks info on crypto mechanisms with simple static analysis
 - Use 3rd party libraries, and statically link them
- On iOS, use *OpenSSL* or other libs, build with source code.
 - Instead of using *CryptoKit*, *CCCrypt* APIs
 - Link statically

```

loc_1EF24
ADRP      X1, #aMbedtls@PAGE ; "MBEDTLS"
ADRP      X2, #aIEncryptedInto@PAGE ; "[i] Encrypted into buffer:"
ADD       X1, X1, #aMbedtls@PAGEOFF ; "MBEDTLS"
ADD       X2, X2, #aIEncryptedInto@PAGEOFF ; "[i] Encrypted into buffer:"
MOV       W0, #2
BL        __android_log_print
ADD       X0, SP, #0x2F0+var_1E8
BL        .mbedtls_gcm_init
ADD       X0, SP, #0x2F0+var_1E8
ADD       X2, SP, #0x2F0+var_200
MOV       W1, #2
MOV       W3, #0x80
BL        .mbedtls_gcm_setkey
ADD       X0, SP, #0x2F0+var_1E8 ; int
ADD       X2, SP, #0x2F0+var_210
MOV       W1, #1
MOV       W3, #0x10
MOV       X4, XZR
MOV       X5, XZR
BL        .mbedtls_gcm_starts
ADD       X0, SP, #0x2F0+var_1E8 ; int
ADD       X3, SP, #0x2F0+var_250
MOV       W1, #0x20 ; ''
MOV       X2, X19
ADD       X24, SP, #0x2F0+var_250
BL        .mbedtls_gcm_update
ADD       X0, SP, #0x2F0+var_1E8
BL        .mbedtls_gcm_free
ADRL      X0, aHelloWorld ; "hello world"
MOV       W1, #0xC
BL        __strlen_chk
CBZ       X0, loc_1F008

```

Symbols show up when
crypto library is not built
with visibility-hidden

```

loc_9374
ADRP      X1, #aMbedtls@PAGE ; "MBEDTLS"
ADRP      X2, #aIEncryptedInto@PAGE ; "[i] Encrypted into buffer:"
ADD       X1, X1, #aMbedtls@PAGEOFF ; "MBEDTLS"
ADD       X2, X2, #aIEncryptedInto@PAGEOFF ; "[i] Encrypted into buffer:"
MOV       W0, #2
BL        __android_log_print
ADD       X0, SP, #0x2F0+var_1E8
BL        sub_28070
ADD       X0, SP, #0x2F0+var_1E8
ADD       X2, SP, #0x2F0+var_200
MOV       W1, #2
MOV       W3, #0x80
BL        sub_280AC
ADD       X0, SP, #0x2F0+var_1E8 ; int
ADD       X2, SP, #0x2F0+var_210
MOV       W1, #1
MOV       W3, #0x10
MOV       X4, XZR
MOV       X5, XZR
BL        sub_282B4
ADD       X0, SP, #0x2F0+var_1E8 ; int
ADD       X3, SP, #0x2F0+var_250
MOV       W1, #0x20 ; ''
MOV       X2, X19
ADD       X24, SP, #0x2F0+var_250
BL        sub_285C0
ADD       X0, SP, #0x2F0+var_1E8
BL        sub_28AF0
ADRL      X0, aHelloWorld ; "hello world"
MOV       W1, #0xC
BL        __strlen_chk
CBZ       X0, loc_9458

```

Crypto symbols are
stripped out

System APIs

- On Android use BouncyCastle and build with source code
 - Instead of using *java.crypto* APIs
 - Call BC APIs directly, instead of via *Java Provider* route

Uses System APIs

```
private static byte[] generateHash(HashAlgorithm algorithm, byte[] key,
    throws NoSuchAlgorithmException, InvalidKeyException {
    String algo = "Hmac" + algorithm.toString();

    Mac mac = Mac.getInstance(algo);
    mac.init(new SecretKeySpec(key, algo));

    return mac.doFinal(data);
}
```

Cannot be obfuscated

```
import d.a.a.a.c.c;
import java.nio.ByteBuffer;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import me.zhanghai.android.materialprogressbar.BuildConfig;
```

```
private static byte[] h(a aVar, byte[] bArr, byte[] bArr2) {
    String str = "Hmac" + aVar.toString();
    Mac instance = Mac.getInstance(str);
    instance.init(new SecretKeySpec(bArr, str));
    return instance.doFinal(bArr2);
}
```

Without Static Linking BouncyCastle

Uses Bouncycastle APIs

```
private static byte[] generateHash(HashAlgorithm algorithm, byte[] key, byte[] data) {
    HMac hMac = null;
    byte[] out;
    switch(algorithm) {
        case SHA1:
            hMac = new HMac(new SHA1Digest());
            break;
        case SHA256:
            hMac = new HMac(new SHA256Digest());
            break;
        case SHA512:
            hMac = new HMac(new SHA512Digest());
            break;
    }
    hMac.init(new KeyParameter(key));
    hMac.update(data, 0, data.length);
    out = new byte[hMac.getMacSize()];
    hMac.doFinal(out, 0);
    return out;
}
```

Can be obfuscated

```
import d.b.a.d.c;
import d.b.a.d.d;
import d.b.a.d.e;
import d.b.a.d.f;
import java.nio.ByteBuffer;
```

```
private static byte[] h(b bVar, byte[] bArr, byte[] bArr2) {
    int i = a.f3799a(bVar.ordinal());
    d.b.a.e.a aVar = i != 1 ? i != 2 ? i != 3 ? null : new d.b.a.e.a(new f()) : new d.b.a.e.a(new e()) : new d.b.a.e.a(new d());
    aVar.e(new d.b.a.f.a(bArr));
    aVar.e(bArr2, 0, bArr2.length);
    byte[] bArr3 = new byte[aVar.c()];
    aVar.a(bArr3, 0);
    return bArr3;
}
```

With Static Linking BouncyCastle

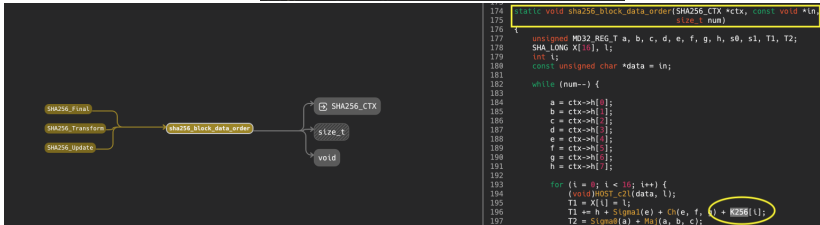
Crypto Constants

- Constants are used by crypto algorithms.
- SHA256 uses the following as one of the constant

```

139 #ifndef SHA256_ASM
140 static const SHA_LONG K256[64] = {
141     0x428a2f98UL, 0x71374491UL, 0xb5c0fbcFUL, 0xe9b5dba5UL,
142     0x3959c25bUL, 0x59f111f1UL, 0x923f82a4UL, 0xab1c5ed5UL,
143     0xd807aa98UL, 0x12835b01UL, 0x243185beUL, 0x550c7dc3UL,
144     0x72be5d74UL, 0x80deb1feUL, 0x9bdc06a7UL, 0xc19bf174UL,
145     0xe49b69c1UL, 0xefbe4786UL, 0x0fc19dc6UL, 0x240ca1ccUL,
146     0x2de92c6fUL, 0x4a7484aaUL, 0x5cb0a9dcUL, 0x76f988daUL,
147     0x983e5152UL, 0xa831c66dUL, 0xb00327c8UL, 0xbf597fc7UL,
148     0xc6e00bf3UL, 0xd5a79147UL, 0x06ca6351UL, 0x14292967UL,
149     0x27b70a85UL, 0x2e1b2138UL, 0x4d2c6dfcUL, 0x53380d13UL,
150     0x650a7354UL, 0x766a0abbUL, 0x81c2c92eUL, 0x92722c85UL,
151     0xa2bfe8a1UL, 0xa81a664bUL, 0xc24b8b70UL, 0xc76c51a3UL,
152     0xd192e819UL, 0xd6990624UL, 0xf40e3585UL, 0x106aa070UL,
153     0x19a4c116UL, 0x1e376c08UL, 0x2748774cUL, 0x34b0bcb5UL,
154     0x391c0cb3UL, 0x4ed8aa4aUL, 0x5b9cca4fUL, 0x682e6ff3UL,
155     0x748f82eeUL, 0x78a5636fUL, 0x84c87814UL, 0x8cc70288UL,
156     0x90beffaUL, 0xa4b86ceUL, 0xbef9a3f7UL, 0xc67178f2UL
157 };

```



- FB app uses SHA1 and SHA256 in libcoldstart.so
- Can help us in finding the API entry points, or critical sections of the algorithm

```

SHA1_update
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
00000000 38 02 06 92 38 02 05 7a a0 03 02 01 02 02 10 06 0...0...Zm...0
00000010 06 06 d3 18 cc 45 16 90 04 5a 57 cd fc a6 fc 38 .....2m...0
00000020 6d 06 09 2a 85 40 86 f7 6d 01 01 06 05 00 30 70 ...8...89
00000030 31 0b 30 09 06 03 55 04 06 13 02 55 53 31 15 30 1...0...U...U51,0
00000040 13 06 03 55 04 0a 13 0c 44 69 67 69 43 65 72 74 ...U...Digicert
00000050 28 49 66 63 31 19 30 17 06 03 55 04 06 13 10 77 Incl.0...U...U
00000060 77 77 26 04 09 07 09 03 05 72 74 26 03 6f 6d 31 mw.digicert.com
00000070 24 30 2d 04 03 55 04 03 12 26 44 09 07 69 42 62 0...U...Digicert
00000080 72 74 20 53 40 41 32 20 40 69 67 6d 20 41 73 73 SHA2_High_Asi
00000090 75 72 61 66 63 65 20 53 65 72 76 65 72 20 43 41 urance Server CA
000000a0 30 1e 17 04 32 31 30 39 30 30 30 30 30 30 30 30 0...210900000000
000000b0 5a 17 04 32 31 31 32 30 38 32 33 35 39 35 39 5a 2...2112002359592
000000c0 30 69 31 06 30 09 06 03 55 04 06 13 02 55 53 31 013...0...U...U51
000000d0 13 30 11 06 03 55 04 06 13 0a 43 61 6c 69 66 6f 0...U...Califo
000000e0 72 66 69 61 31 13 30 11 06 03 55 04 07 13 0a 4d rmal.0...U...H
000000f0 65 66 6c 6f 20 50 61 72 6b 31 17 30 15 06 03 55 emlo Park.0...U
00000100 04 0a 13 06 46 61 63 05 62 6f 6f 6b 2c 20 49 66 ...Facebook, In
00000110 63 26 31 17 30 15 06 03 55 04 03 0c 0e 2a 2e 46 c2.0...U...F
00000120 61 63 65 62 6f 6f 6b 2c 63 6f 6d 30 50 30 13 06 acbook.com?U
00000130 07 2a 16 48 ce 3d 02 01 06 00 2a 16 48 ce 3d 03 ...F.H...*H...
00000140 01 07 03 42 00 04 c4 14 16 6f 2c 45 20 d3 bf a6 ...B...0,E...
00000150 0a 97 44 ab 40 96 49 08 43 65 1b 09 42 44 22 d7 ...0.0.T.Co...D
00000160 7c 0f 0b 90 2c 6a 33 c1 a0 bc 8a 95 53 ac 5e 0b [...]3...2...
00000170 0b 10 26 79 ba 82 80 df 21 94 01 fa 6b 80 75 7f ...0p...h...h...
00000180 c5 36 1d fe 0e cd a3 82 03 f0 30 82 03 f4 30 1f 6...0...0...0
00000190 06 03 55 14 23 04 1a 30 16 00 14 51 6b ff 90 af ...U.#...0...0
000001a0 82 07 75 3c cc 09 65 64 62 42 12 ba 59 72 3b 38 ...uc...edh...Yr.0
000001b0 14 06 03 55 1d 06 04 16 04 14 71 3c 25 1d 07 0a ...U...gc...
000001c0 91 47 2c f1 1f cc 96 ac 0a 4a 26 c4 9d 30 81 ...U...0A...0
000001d0 b5 06 03 55 1d 11 04 81 ad 30 81 aa 82 0e 2a 2e ...U...0...0...
000001e0 66 61 63 65 62 6f 6f 6b 2c 63 6f 6d 82 0e 2a 2e facebook.com.*
000001f0 06 61 63 65 62 6f 6f 6b 2c 65 65 74 82 0b 2a 2e facebook.net.*
00000200 62 63 64 66 2a 66 65 74 82 0b 2a 2e 66 63 65 65 fcdm.net.*.xy
00000210 62 70 63 6f 6d 82 10 2a 2e 6d 2a 66 63 65 65 bx.com.*.a face
00000220 62 6f 6f 6b 2c 63 6f 6d 82 0f 2a 2e 6d 65 73 73 book.com.*.mess
00000230 65 66 67 65 72 2a 63 6f 6d 82 0e 2a 2e 78 78 2a enger.com.*.xx
00000240 06 62 63 64 66 2a 66 65 74 82 0e 2a 2e 78 79 2a fcdm.net.*.xy
00000250 66 62 63 64 66 2a 66 65 74 82 0e 2a 2e 78 79 2a fcdm.net.*.xz
00000260 06 62 63 64 66 2a 66 65 74 82 0c 66 63 63 65 62 fcdm.net.*face
00000270 6f 6f 6b 2c 63 6f 6d 82 0d 6d 65 73 73 65 6e 6f ook.com.*messeng
00000280 65 72 2a 63 6f 6d 30 06 06 03 55 1d 0f 01 01 ff er.com...U...
00000290 04 04 03 02 07 00 30 1d 06 03 55 1d 25 04 16 38 ...0...U...0
000002a0 14 06 03 55 1d 06 01 05 07 03 01 06 62 2b 06 01 ...0...U...0
000002b0 05 05 07 03 02 30 75 06 03 55 1d 1f 04 06 30 6c ...0b...U...0
000002c0 30 34 a0 32 a0 30 86 2e 68 74 74 70 3a 2f 2f 63 04 2.0...http://c
000002d0 72 6c 33 2a 64 69 67 69 63 65 72 74 2a 63 6f 6d rl3.digicert.com
000002e0 2f 73 68 61 32 2a 04 61 2d 73 65 72 76 65 72 2d /sha2-ha-server-
000002f0 67 30 2a 63 72 6c 30 34 a0 32 a0 30 86 2e 68 74 sp/crl2.0...h
00000300 74 70 3a 2f 2f 63 72 6c 34 2a 64 69 67 69 63 65 rt.com/sha2-ha-s
00000310 72 74 2a 63 6f 6d 2f 73 68 61 32 2d 68 61 2d 73 rt.com/sha2-ha-s
00000320 65 72 76 65 72 2d 67 36 2a 63 72 6c 30 3a 06 03 erver-g6.crlB
00000330 55 1d 20 04 37 30 35 36 31 06 06 6f 01 0c 01 02 U...70503.g...

```

- Obfuscate crypto constants with simple XOR masks.

- Perform critical code in native, harder to reverse
- Use *Openssl* or *mbedtls*
 - Use *-fvisibility=hidden*
 - Use static linking
- Diffing or pattern matching can help in reversing popular crypto libraries
 - Crypto constants can easily indicate the matching functions
 - Use OLLVM to obfuscate the control flow and protect the strings

Similarity	Confid	Change	EA Primary	Name Primary	EA Secondary	Name Secondary	Co	Algorithm	Matched B	Basic Block	Basic Block	Matched Inst	Instructions F	Instruct
1.00	0.99	-----	0003E304	mbdctl_gcm_finish	00028754	sub_00028754		Edges Flow Graph MD Index	15	15	15	117	117	117
0.97	0.99	G---E-C	0001C810	mbdctl_gcm_auth_decrypt	000289B4	sub_000289B4		Call Reference	16	17	16	79	83	79
0.97	0.99	G---E--	0001D520	mbdctl_gcm_starts	000282B4	sub_000282B4		Call Reference	23	24	23	125	129	125
0.97	0.99	G---E--	0001CFC0	mbdctl_gcm_update	000285C0	sub_000285C0		Call Reference	24	25	24	101	105	101
0.96	0.99	G---E-C	0001D6A0	mbdctl_gcm_setkey	000280AC	sub_000280AC		Call Reference	10	11	10	130	134	130
0.94	0.99	G---E--	0001CBB0	mbdctl_gcm_crypt_and_tag	00028928	sub_00028928		Call Sequence (Sequence)	19	21	19	152	160	152
0.83	0.96	G---E--	0001D680	mbdctl_gcm_free	00028AF0	sub_00028AF0		Call Reference	6	8	6	19	27	19
0.59	0.98	G---E--	0001D8C0	mbdctl_gcm_init	00028070	sub_00028070		Call Reference	1	2	1	15	19	15

1. App Hardening

2. Build Settings
Techniques

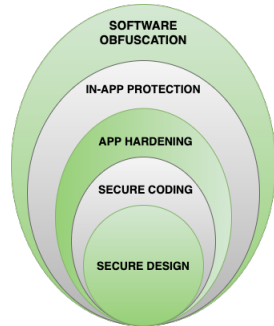
3. Code Hardening
Techniques

4. Data Hardening

5. Crypto Hardening

6. Discussion

- Often the mobile app hardening discussion starts and stops at RASP
- RASP is a good starting point.
 - But not necessarily always sufficient.
 - Depends on the threat model.
- RASP techniques are often standard and known
 - Without additional hardening, easy to defeat
- App hardening against RE will enhance RASP protection.



1. App Hardening
2. Build Settings Techniques
3. Code Hardening Techniques
4. Data Hardening
5. Crypto Hardening
6. Discussion

- Open source CTF code R2Pay incorporates the above discussed points
- Code obfuscation
 - proguard for Java
 - OLLVM obfuscator for C
- Code hardening
 - Inlining of sensitive code
 - RASP calls are inlined as part of the core logic
 - libc apis replaced with custom code and syscalls
 - byte arrays used for all sensitive data
 - memory wiped just after use

■ Crypto obfuscation

- Symbols removed
- no traces of which crypto library being used
- Crypto constants derived at runtime from arbitrary constants
- Open source WBC to hide crypto key

DEMO

- Discussed what is app hardening and its importance
- Discussed with hardening techniques
 - Build Settings
 - Code
 - Data
 - Crypto
- Case study on R2Pay
- One technique in itself will not be sufficient, use all of them for maximum benefit.
- It is an iterative process
- Determine your threat model and use accordingly.

- Slides at: <https://github.com/su-vikas/Presentations>

