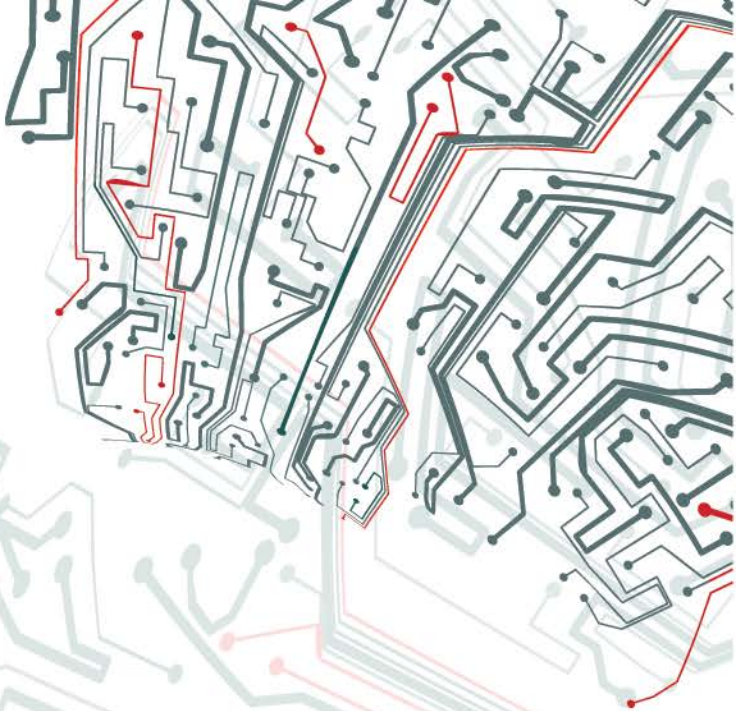
Beyond The ‘Cript: Practical iOS Reverse Engineering

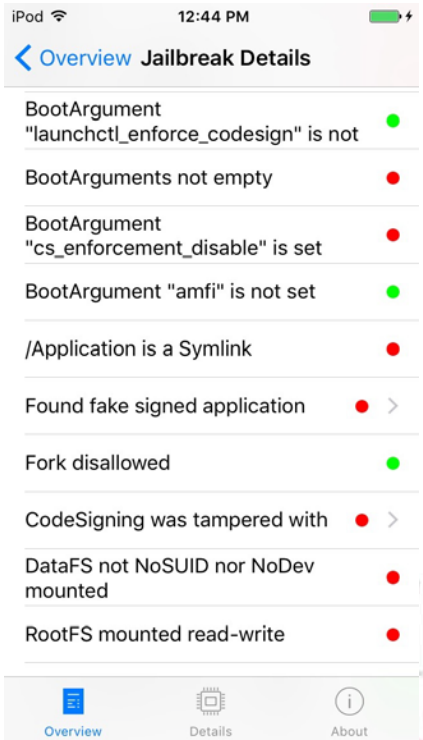
Michael Allen (@_dark_knight_)
Security Consultant



IOActive
Hardware | Software | Wetware
SECURITY SERVICES

LASCON
LONESTAR APPLICATION SECURITY CONFERENCE

Identifying and bypassing Simple Jailbreak Detection Routines Case Study



IOActive™

Case Study: Viewing File System Activity

- Using filemon -l
- Creates hard links to temporary files

```
Auto-linked /private/var/mobile/Containers/Data/Application/CB61B9D0-3C4D-4643-B499-443D3666A0E9/Library/Caches/Snapsho
ts/de.sektioneins.sysinfo/251357E1-92A5-4D5F-82F1-B2ACAFD53527@2x.png to /private/var/tmp/filemon/251357E1-92A5-4D5F-82
F1-B2ACAFD53527@2x.png.filemon.15
   92 SpringBoard       Created        /private/var/mobile/Containers/Data/Application/CB61B9D0-3C4D-4643-B499-443D3666
A0E9/Library/Caches/Snapshots/de.sektioneins.sysinfo/251357E1-92A5-4D5F-82F1-B2ACAFD53527@2x.png
   92 SpringBoard       Modified       /private/var/mobile/Containers/Data/Application/CB61B9D0-3C4D-4643-B499-443D3666
A0E9/Library/Caches/Snapshots/de.sektioneins.sysinfo/251357E1-92A5-4D5F-82F1-B2ACAFD53527@2x.png
   92 SpringBoard       Deleted        /private/var/mobile/Containers/Data/Application/CB61B9D0-3C4D-4643-B499-443D3666
A0E9/Library/Caches/Snapshots/de.sektioneins.sysinfo/E9396C40-CF92-48DE-8834-13BDB25819A3@2x.png
   92 SpringBoard       Deleted        /private/var/mobile/Containers/Data/Application/CB61B9D0-3C4D-4643-B499-443D3666
A0E9/Library/Caches/Snapshots/de.sektioneins.sysinfo/downscaled/CDA55A72-AF2A-4C4B-B72B-4F36B664FADE@2x.png
Auto-linked /private/var/mobile/Containers/Data/Application/CB61B9D0-3C4D-4643-B499-443D3666A0E9/Library/Caches/Snapsho
ts/de.sektioneins.sysinfo/downscaled/1422BF63-E775-4BB6-B662-BD193F4CF1F0@2x.png to /private/var/tmp/filemon/1422BF63-E
775-4BB6-B662-BD193F4CF1F0@2x.png.filemon.16
```


IOActive™

Case Study: Viewing Logs

- Using idevicesyslog [libimobiledevice]

```
Oct 24 12:37:19 Neo SysSecInfo[1183] <Warning>: Get fact for de.sektioneins.artefact.fileDoesNotExists./panguaxe.installed
Oct 24 12:37:33 Neo SysSecInfo[1183] <Warning>: Get fact for de.sektioneins.artefact.fileDoesNotExists./xuanyuansword.installe
d
Oct 24 12:37:33 Neo SysSecInfo[1183] <Warning>: Get fact for de.sektioneins.artefact.fileDoesNotExists./System/Library/LaunchD
aemons/io.pangu.axe.untether.plist
Oct 24 12:37:33 Neo SysSecInfo[1183] <Warning>: Get fact for de.sektioneins.artefact.fileDoesNotExists./panguaxe.installed
Oct 24 12:37:34 Neo SysSecInfo[1183] <Warning>: Get fact for de.sektioneins.artefact.fileDoesNotExists./System/Library/Caches/
com.apple.dyld/enable-dylibs-to-override-cache
Oct 24 12:37:34 Neo SysSecInfo[1183] <Warning>: Get fact for de.sektioneins.artefact.fileDoesNotExists./System/Library/LaunchD
aemons/com.evad3rs.evasi0n7.untether.plist
Oct 24 12:37:34 Neo SysSecInfo[1183] <Warning>: Get fact for de.sektioneins.artefact.fileDoesNotExists./bin/bash
Oct 24 12:37:34 Neo SysSecInfo[1183] <Warning>: Get fact for de.sektioneins.artefact.fileDoesNotExists./bin/sh
Oct 24 12:37:35 Neo SysSecInfo[1183] <Warning>: Get fact for de.sektioneins.artefact.fileDoesNotExists./Applications/Cydia.app
/Cydia
Oct 24 12:37:35 Neo SysSecInfo[1183] <Warning>: Get fact for de.sektioneins.artefact.fileDoesNotExists./usr/sbin/sshd
Oct 24 12:37:36 Neo SysSecInfo[1183] <Warning>: BootArgs: cs_enforcement_disable=1
Oct 24 12:37:38 Neo SysSecInfo[1183] <Warning>: BAD: BootArg "cs_enforcement_disable" is set.
Oct 24 12:37:41 Neo kernel[0] <Notice>: Sandbox: SysSecInfo(1183) deny(1) process-fork
Oct 24 12:37:41 Neo SysSecInfo[1183] <Warning>: DATAFS allows suid and dev files
Oct 24 12:37:41 Neo SysSecInfo[1183] <Warning>: ROOTFS is read-write mounted
Oct 24 12:37:41 Neo SysSecInfo[1183] <Warning>: Testing tfp0
Oct 24 12:37:42 Neo SysSecInfo[1183] <Warning>: task_for_pid0 scheint nicht gepatcht
```


IOActive™

Case Study: Obtaining The Binary

- Dump the binary (facilitated by DYLD and DYLD_INSERT_LIBRARIES environment variable)

```
root@Neo (/var/root)# DYLD_INSERT_LIBRARIES=dumpdecrypted.dylib /var/mobile/Containers/Bundle/Application
/25A72B67-9342-471F-ADF9-FC80C5F21B22/SysSecInfo.app/SysSecInfo
mach-o decryption dumper

DISCLAIMER: This tool is only meant for security research purposes, not for application crackers.

[+] detected 64bit ARM binary in memory.
[+] offset to cryptid found: @0x1000b0ac8(from 0x1000b0000) = ac8
[+] Found encrypted data at address 00004000 of length 770048 bytes - type 1.
[+] Opening /private/var/mobile/Containers/Bundle/Application/25A72B67-9342-471F-ADF9-FC80C5F21B22/SysSec
Info.app/SysSecInfo for reading.
[+] Reading header
[+] Detecting header type
[+] Executable is a FAT image - searching for right architecture
[+] Correct arch is at offset 835584 in the file
[+] Opening SysSecInfo.decrypted for writing.
[+] Copying the not encrypted start of the file
[+] Dumping the decrypted data into the file
[+] Copying the not encrypted remainder of the file
[+] Setting the LC_ENCRYPTION_INFO->cryptid to 0 at offset ccac8
[+] Closing original file
[+] Closing dump file
```


IOActive™

Case Study: Obtaining Symbols

- Dump the symbols along with dylib's to which they belong

```
DarkKnight:Cript michael$ jtool -v -S SysSecInfo-arm64
0xf6338 0000000100000000 T __mh_execute_header
0xf6348                  U _CFStringGetCStringPtr: /System/Library/Frameworks/CoreFoundation.framework/CoreFoundation
0xf6358                  U _NSLog: /System/Library/Frameworks/Foundation.framework/Foundation
0xf6368                  U _NSSearchPathForDirectoriesInDomains: /System/Library/Frameworks/Foundation.framework/Foundation
0xf6378                  U _NSStringFromClass: /System/Library/Frameworks/Foundation.framework/Foundation
0xf6388                  U _OBJC_CLASS_$_NSArray: /System/Library/Frameworks/CoreFoundation.framework/CoreFoundation
0xf6398                  U _OBJC_CLASS_$_NSBundle: /System/Library/Frameworks/Foundation.framework/Foundation
0xf63a8                  U _OBJC_CLASS_$_NSData: /System/Library/Frameworks/CoreFoundation.framework/CoreFoundation
0xf63b8                  U _OBJC_CLASS_$_NSDate: /System/Library/Frameworks/CoreFoundation.framework/CoreFoundation
0xf63c8                  U _OBJC_CLASS_$_NSDateFormatter: /System/Library/Frameworks/Foundation.framework/Foundation
0xf63d8                  U _OBJC_CLASS_$_NSDictionary: /System/Library/Frameworks/CoreFoundation.framework/CoreFoundation
0xf63e8                  U _OBJC_CLASS_$_NSLocale: /System/Library/Frameworks/CoreFoundation.framework/CoreFoundation
0xf63f8                  U _OBJC_CLASS_$_NSLock: /System/Library/Frameworks/Foundation.framework/Foundation
0xf6408                  U _OBJC_CLASS_$_NSMutableArray: /System/Library/Frameworks/CoreFoundation.framework/CoreFoundation
0xf6418                  U _OBJC_CLASS_$_NSMutableDictionary: /System/Library/Frameworks/CoreFoundation.framework/CoreFoundation
0xf6428                  U _OBJC_CLASS_$_NSNotificationCenter: /System/Library/Frameworks/Foundation.framework/Foundation
0xf6438                  U _OBJC_CLASS_$_NSNumber: /System/Library/Frameworks/Foundation.framework/Foundation
0xf6448                  U _OBJC_CLASS_$_NSObject: /usr/lib/libobjc.A.dylib
0xf6458                  U _OBJC_CLASS_$_NSPredicate: /System/Library/Frameworks/Foundation.framework/Foundation
0xf6468                  U _OBJC_CLASS_$_NSProcessInfo: /System/Library/Frameworks/Foundation.framework/Foundation
```


IOActive™

Case Study: Extracting strings

- Any interesting strings?
- Dump **cstring section** (same as running strings)
  - Knowledge of SEGMENTS and sections important

```
DarkKnight:Cript michael$ jtool -d __TEXT.__cstring SysSecInfo-arm64

Dumping C-Strings from address 0x100092cad (Segment: __TEXT.__cstring)..
Address : 0x100092cad = Offset 0x92cad
0x100092cad: de.sektioneins.artefact.symlinksApplication
0x100092cd9: /Application is a Symlink
0x100092cf3: /Application is no Symlink
0x100092d0e: /Applications
0x100092d1c: Error - Cannot access /Applications\r
0x100092d41: hash
0x100092d46: TQ,R
0x100092d4b: superclass
0x100092d56: T#,R
0x100092d5b: description
0x100092d67: T@"NSString",R,C
0x100092d78: debugDescription
0x100092d89: de.sektioneins.infoItem.ProcessList
0x100092dad: Processes
0x100092db7: (len not supplied?)
0x100092db8: ProcessList
0x100092dc4: ProcessListSegue
0x100092dd5: type
0x100092dda: T@"NSNumber",&,Vtype
0x100092def: v8@?0
```


IOActive™

Case Study: Extracting DYLIB'S

- procexp <pid> regions

```
MALLOC_LARGE metadata  0x595af457 00000001002cc000-00000001002d4000 [  32K] rw-/rwx PRV
(0)                    0x5939c6bf 00000001002d4000-00000001002d8000 [  16K] r-x/rwx COW /Library/Frameworks/CydiaSubstrate.framework/Libra
ries/SubstrateBootstrap.dylib
(0)                    0x5939c6bf 00000001002d8000-00000001002dc000 [  16K] rw-/rwx COW /Library/Frameworks/CydiaSubstrate.framework/Libra
ries/SubstrateBootstrap.dylib
(0)                    0x5939c6bf 00000001002dc000-00000001002e0000 [  16K] r--/rwx COW /Library/Frameworks/CydiaSubstrate.framework/Libra
ries/SubstrateBootstrap.dylib
MALLOC guard page      0x00000000 00000001002e0000-00000001002e4000 [  16K] ---/rwx NUL
MALLOC metadata        0x5a8b1ff7 00000001002e4000-00000001002fc000 [  96K] rw-/rwx PRV
MALLOC guard page      0x00000000 00000001002fc000-0000000100300000 [  16K] ---/rwx NUL
MALLOC guard page      0x00000000 0000000100300000-0000000100304000 [  16K] ---/rwx NUL
```

Dump the library with lldb

```
(lldb) me r -o /tmp/tmpFile -b --force 0x00000001002d4000 0x00000001002d8000
16384 bytes written to '/tmp/tmpFile'
(lldb)
```


IOActive™

Case Study: Extracting DYLIB'S

```
DarkKnight:tmp michael$ file tmpFile
tmpFile: Mach-O 64-bit dynamically linked shared library
DarkKnight:tmp michael$ jtool -d tmpFile

Warning: companion file ./tmpFile.ARM64.3134CFB2-F722-310E-A2C7-42AE4DC131AB not found
Disassembling from file offset 0x3da8, Address 0x3da8
        3da8    STP     X29, X30, [SP,#-16]!      ;
        3dac    ADD     X29, SP, #0        X29 = 0x3db0 -|
        3db0    ADR     X0, #248          "MSExitZero"     ; ->R0 = 0x3ea8
        3db4    NOP                       ;
        3db8    BL      0x3e54
; // if (0x..  != 0) then goto 0x3e04
        3dbc    CBNZ    X0, 0x3e04        ;
        3dc0    ADR     X0, #305          "/System/Library/Frameworks/Security.framework/Security"
        ; ->R0 = 0x3ef1
        3dc4    NOP                       ;
        3dc8    MOVZ    W1, 0x11                  ; ->R1 = 0x11
        3dcc    BL      0x3e48
; // if (0x..  == 0) then goto 0x3dfc
        3dd0    CBZ     X0, 0x3dfc        ;
        3dd4    BL      0x3e3c
        3dd8    ADR     X0, #336          "/System/Library/Frameworks/CoreFoundation.framework/Core
Foundation"     ; ->R0 = 0x3f28
        3ddc    NOP                       ;
        3de0    MOVZ    W1, 0x9                   ; ->R1 = 0x9
```

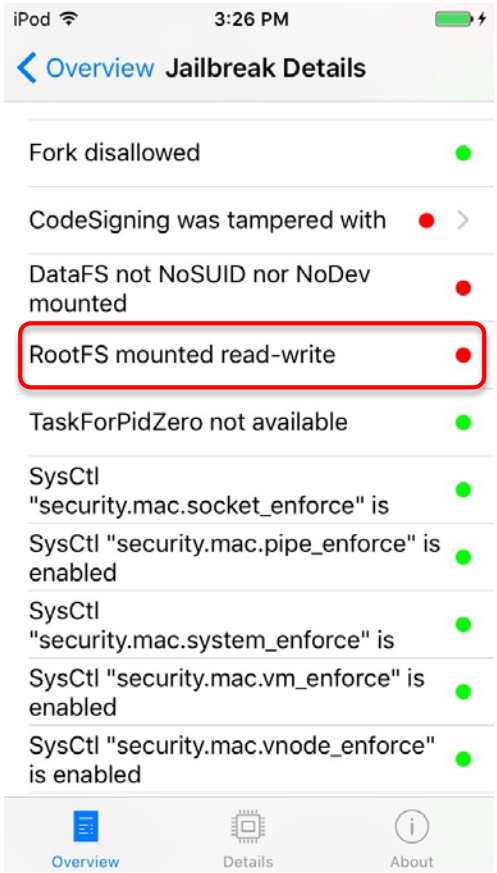

IOActive™

Case Study: Obtaining Classes

```
DarkKnight:Cript michael$ jtool -v -d objc SysSecInfo-arm64
Warning: companion file ./SysSecInfo-arm64.ARM64.8DBD392A-5FA8-3710-9E93-CB78D4BAF640 not found
// Dumping class 0 (ApplicationSymlinkArtefact)
@interface ApplicationSymlinkArtefact : Artefact
// 4 properties:
@property (readonly) unsigned long long hash;
@property (readonly) Class superclass;
@property (readonly,copy) NSString description;
@property (readonly,copy) NSString debugDescription;
// 1 class methods
 /* 0 */ 0x100005c0c - load;  // Protocol 1186a73f2
// 5 instance methods
 /* 0 */ 0x100005c60 - uuid;  // Protocol @16@0:8
 /* 1 */ 0x100005c8c - titleFactFailed;  // Protocol @16@0:8
 /* 2 */ 0x100005cb8 - titleFactPassed;  // Protocol @16@0:8
 /* 3 */ 0x100005ce4 - getFact;  // Protocol B16@0:8
 /* 4 */ 0x100005d44 - orderNr;  // Protocol @16@0:8
@end
// Dumping class 1 (ProcessListInfoItem)
@interface ProcessListInfoItem : InfoItem
// 4 properties:
@property (readonly) unsigned long long hash;
@property (readonly) Class superclass;
@property (readonly,copy) NSString description;
@property (readonly,copy) NSString debugDescription;
```


IOActive™

Case Study: Bypassing RootFS Check



IOActive™

Case Study: Bypassing RootFS Check

```
-[RootFSMountOptionArtefacts getFact]:
    10000698c     STP     X20, X19, [SP,#-32]!      ;
    100006990     STP     X29, X30, [SP,#16]        ;
    100006994     ADD     X29, SP, #16          ; $$ R29 = SP + 0x10
    100006998     SUB     SP, SP, 2176                ; SP -= 0x880 (stack frame)
    10000699c     NOP                           ;
    1000069a0     LDR     X19, #759416          ; X19 = *(1000c0018) = -libSystem.B.dylib::___stack_chk_guard-
    1000069a4     LDR     X19, [X19, #0]        ;       R19 = *(libSystem.B.dylib::___stack_chk_guard)
    1000069a8     STUR    X19, X29, #-24        ;= X19  0x0
    1000069ac     ADD     X0, SP, #0              ; $$ R0 = SP + 0x0
    1000069b0     MOVZ    W1, 0x878                   ; ->R1 = 0x878
    1000069b4     BL      libSystem.B.dylib::_bzero         ; 0x10009050c
;   R0 = libSystem.B.dylib::_bzero(SP + 0x0,2168);
    1000069b8     ADR     X0, #575060           "/"     ; ->R0 = 0x10009300c
    1000069bc     NOP                           ;
    1000069c0     ADD     X1, SP, #0              ; $$ R1 = SP + 0x0
    1000069c4     BL      libSystem.B.dylib::_statfs        ; 0x10009077c
; // if (libSystem.B.dylib::_statfs  == 0) then goto 0x1000069d8
    1000069c8     CBZ     X0, 0x1000069d8  ;
    1000069cc     ADR     X0, #841148           @"ERROR BAD ERROR BAD - maybe we need another symbol for UNEXPI
E OF TESTCASE"          ; ->R0 = 0x1000d3f88
    1000069d0     NOP                           ;
    1000069d4     B       0x1000069e8
    1000069d8     LDRB    W8, [SP, #64]     ;--R8 = *(SP + 64) = 0x0 ... (null)?..
; // if (R8  != 0) then goto 0x1000069f4
    1000069dc     TBNZ    W8, #0, 0x1000069f4       ;
    1000069e0     ADR     X0, #841096           @"ROOTFS is read-write mounted" ; ->R0 = 0x1000d3f68
    1000069e4     NOP                           ;
```

statfs argument

statfs func call

Patch here

IOActive™

Case Study: Bypassing RootFS Check

- Patch register w8

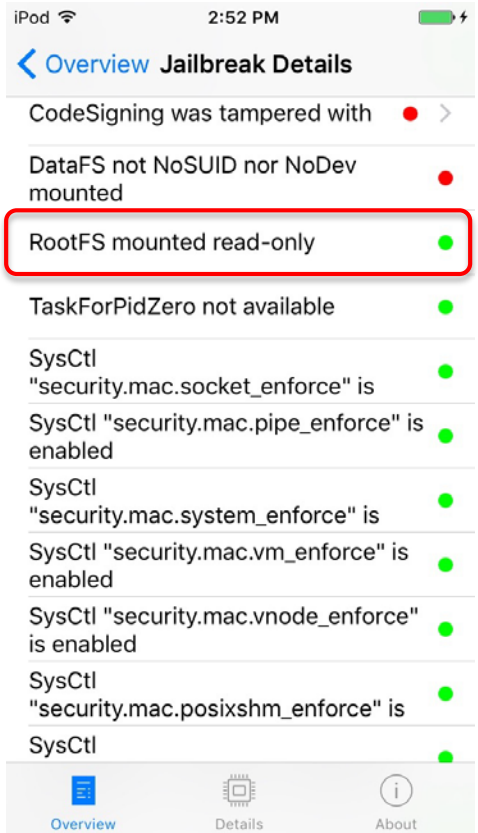
```
SysSecInfo`___lldb_unnamed_function59$$SysSecInfo:
->  0x1000269dc <+80>: tbnz   w8, #0, 0x1000269f4       ; <+104>
    0x1000269e0 <+84>: adr    x0, #841096               ; @"ROOTFS is read-write mounted"
    0x1000269e4 <+88>: nop
    0x1000269e8 <+92>: bl     0x1000b0284               ; symbol stub for: NSLog
(lldb) reg read w8
      w8 = 0x00000000
(lldb) reg write w8 1
(lldb) c
```

Patch here

```
; Foundation::_NSLog(@"ROOTFS is read-write mounted");
    1000069ec     MOVZ    W0, 0x0                      ; ->R0 = 0x0
    1000069f0     B       0x100006a04
    1000069f4     ADR     X0, #841044          @"ROOTFS is read-only mounted"
    1000069f8     NOP                          ;
    1000069fc     BL      Foundation::_NSLog           ; 0x100090284
```

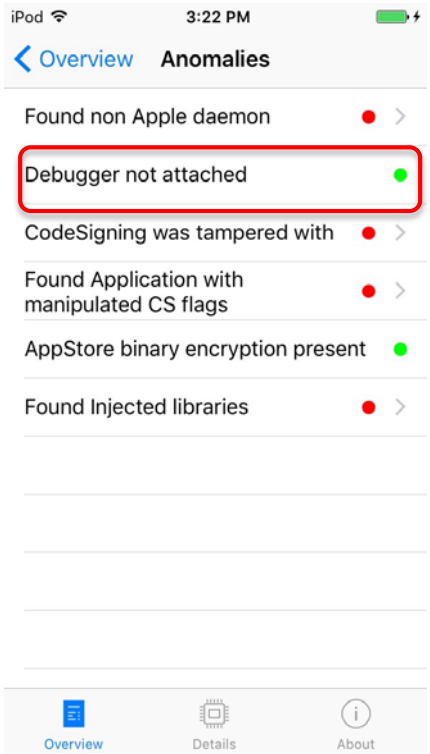

IOActive

Case Study: Bypassing RootFS Check

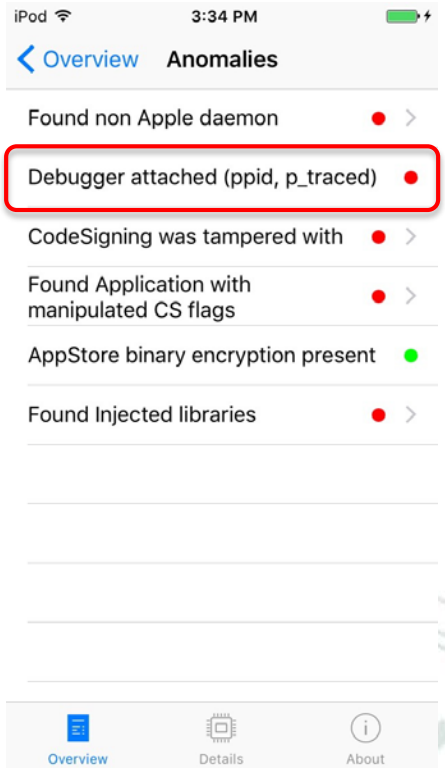


IOActive™

Case Study: Bypassing Debugger Checks



Changes when debugger attached



IOActive™

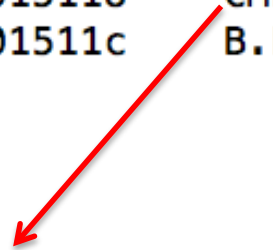
Case Study: Bypassing Debugger Checks (ppid)

```
-[DebugPPIDArtefact getFact]:
   1000150d4    STP    X24, X23, [SP,#-64]!      ;
   1000150d8    STP    X22, X21, [SP,#16]        ;
   1000150dc    STP    X20, X19, [SP,#32]        ;
   1000150e0    STP    X29, X30, [SP,#48]        ;
   1000150e4    ADD    X29, SP, #48         ; $$ R29 = SP + 0x30
   1000150e8    SUB    SP, SP, 688                ; SP -= 0x2b0 (stack frame)
   1000150ec    MOV    X19, X0           ; --X19 = X0 = ARG0
   1000150f0    NOP                      ;
   1000150f4    LDR    X22, #700196      ; X22 = *(1000c0018) = -libSystem.B.dylib::___stack_chk_guard-
   1000150f8    LDR    X22, [X22, #0]    ;       R22 = *(libSystem.B.dylib::___stack_chk_guard)
   1000150fc    STUR   X22, X29, #-56    ;= X22  0x0
   100015100    NOP                      ;
   100015104    LDRSW  X23, 1000df798
        ;
   100015108    -LDR   X0, [X19, X23 ...]        ;       R0 = *(ARG0)
   10001510c    STR    X31, [X19, xX23] ..       ;       R31 = *(ARG0)= 0x0
   100015110    BL     libobjc.A.dylib::_objc_release    ; 0x1000903c8
   100015114    BL     libSystem.B.dylib::_getppid       ; 0x100090614
   100015118    CMP    W0, #1            ;
   10001511c    B.EQ   0x100015174       ;
   100015120    -LDR   X20, [X19, X23 ...]       ; -R0 = *(R0 + 0) = .. *(0x0, no sym) = 0x100000cfeedfacf
. (null)?..
; // if (R20  == 0) then goto 0x100015158
   100015124    CBZ    X20, 0x100015158 ;
   100015128    NOP                      ;
   10001512c    LDR    X1, #827980       ; X1 = *(1000df378) = -stringByAppendingString:-
   100015130    ADR    X2, #794072      @", ppid"         ; ->R2 = 0x1000d6f08
   100015134    NOP                      ;
   100015138    MOV    X0, X20           ; --X0 = X20 = 0x0
   10001513c    BL     libobjc.A.dylib::_objc_msgSend    ; 0x1000903a4
; [? stringByAppendingString:@", ppid"]
   100015140    MOV    X29, X29          ; --X29 = X29 = 0x60
```


IOActive™

Case Study: Bypassing Debugger Checks (ppid)

```
    ;
100015108     -LDR    X0, [X19, X23 ...]        ;       R0 = *(ARG0)
10001510c     STR     X31, [X19, xX23] ..       ;       R31 = *(ARG0)= 0x0
100015110     BL      libobjc.A.dylib::_objc_release    ; 0x1000903c8
100015114     BL      libSystem.B.dylib::_getppid       ; 0x100090614
100015118     CMP     W0, #1                    ;
10001511c     B.EQ    0x100015174               ;
```



Patch here

ppid func call

IOActive™

Case Study: Bypassing Debugger Checks (ppid)

```
SysSecInfo`___lldb_unnamed_function353$$SysSecInfo:
->  0x100059118 <+68>: cmp    w0, #1
    0x10005911c <+72>: b.eq   0x100059174
    0x100059120 <+76>: ldr    x20, [x19, x23]
    0x100059124 <+80>: cbz    x20, 0x100059158
(lldb) reg read w0
w0 = 0x0000072a
(lldb) reg write w0 1
(lldb) c
```

Patch here

- parent process id of calling process

```
root@Neo (/var/root)# ps aux | grep 1834
root      1834   0.0  0.3   687360   2652 s000  S+    6:42PM   0:00.46 debugserver -x backboard *:4444
 /var/mobile/Containers/Bundle/Application/25A72B67-9342-471F-ADF9-FC80C5F21B22/SysSecInfo.app/SysSecI
nfo
```


IOActive

Case Study: Bypassing Debugger Checks (p_traced)

```
   100015194    ORR    W8, WZR, #0x1              ; ->R8 = 0x1
   100015198    STUR   X8, X29, #-64     ;= X8  0x1
   10001519c    BL     libSystem.B.dylib::_getpid          ; 0x100090608
   1000151a0    STUR   X0, X29, #-60     ;= X0  0x0
   1000151a4    SUB    X0, X29, #72       X0 = 0x18 -|
   1000151a8    ORR    W1, WZR, #0x4              ; ->R1 = 0x4
   1000151ac    ADD    X2, SP, #16        X2 = 0x1000151c0 -|
   1000151b0    ADD    X3, SP, #8         X3 = 0x1000151bc -|
   1000151b4    MOVZ   X4, 0x0                    ; ->R4 = 0x0
   1000151b8    MOVZ   X5, 0x0                    ; ->R5 = 0x0
   1000151bc    BL     libSystem.B.dylib::_sysctl          ; 0x100090818
; // if (libSystem.B.dylib::_sysctl  != 0) then goto 0x100015220
   1000151c0    CBNZ   X0, 0x100015220  ;
   1000151c4    LDRB   W8, [SP, #49]    ; -R8 = *(R31 + 49) = .. *(0x1000151f9, no sym) = 0x0 ... ?..
; // if (R8  == 3) then goto 0x100015220
   1000151c8    TBZ    W8, #3, 0x100015220        ;
   1000151cc    -LDR   X20, [X19, X23 ...]        ; -R0 = *(R0 + 0) = .. *(0x0, no sym) = 0x100000cfeed1
. (null)?..
; // if (R20  == 0) then goto 0x100015204
   1000151d0    CBZ    X20, 0x100015204 ;
   1000151d4    NOP                     ;
   1000151d8    LDR    X1, #827808       ; X1 = *(1000df378) = -stringByAppendingString:-
   1000151dc    ADR    X2, #793964      @", p_traced"    ; ->R2 = 0x1000d6f48
   1000151e0    NOP                     ;
   1000151e4    MOV    X0, X20           ; --X0 = X20 = 0x0
   1000151e8    BL     libobjc.A.dylib::_objc_msgSend    ; 0x1000903a4
; [? stringByAppendingString:@", p_traced"]
```

```
int name[4];
struct kinfo_proc info;
size_t info_size = sizeof(info);
int ret;

info.kp_proc.p_flag = 0;

name[0] = CTL_KERN; // kernel-specific information
name[1] = KERN_PROC; // return a struct with process entries.
name[2] = KERN_PROC_PID; // target process selected based on PID
name[3] = getpid(); // PID of process

if(sysctl(name, 4, &info, & info_size, NULL, 0) == -1){
    return 0;
}
```

sysctl func call

Patch here

IOActive™

Case Study: Bypassing Debugger Checks (p_traced)

```
->  0x1000e11bc <+232>: bl     0x10015c818               ; symbol stub for: sysctl
    0x1000e11c0 <+236>: cbnz   w0, 0x1000e1220           ; <+332>
    0x1000e11c4 <+240>: ldrb   w8, [sp, #49]
    0x1000e11c8 <+244>: tbz    w8, #3, 0x1000e1220       ; <+332>
(lldb) n
Process 2060 stopped
* thread #1: tid = 0x340a2, 0x00000001000e11c0 SysSecInfo`___lldb_unnamed_function35
 = 'com.apple.main-thread', stop reason = instruction step over
    frame #0: 0x00000001000e11c0 SysSecInfo`___lldb_unnamed_function353$$SysSecInfo
SysSecInfo`___lldb_unnamed_function353$$SysSecInfo:
->  0x1000e11c0 <+236>: cbnz   w0, 0x1000e1220           ; <+332>
    0x1000e11c4 <+240>: ldrb   w8, [sp, #49]
    0x1000e11c8 <+244>: tbz    w8, #3, 0x1000e1220       ; <+332>
    0x1000e11cc <+248>: ldr    x20, [x19, x23]
(lldb) reg read w0
      w0 = 0x00000000
(lldb) reg write w0 1
```

Patch here

IOActive™

Case Study: Bypassing Fork Check

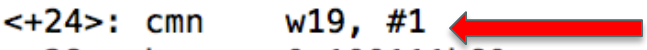
```
-[ForkArtefact getFact]:
   100015b18    STP     X20, X19, [SP,#-32]!      ;
   100015b1c    STP     X29, X30, [SP,#16]        ;
   100015b20    ADD     X29, SP, #16          ; $$ R29 = SP + 0x10
   100015b24    SUB     SP, SP, 16            ; -[ForkArtefact getFact]
   100015b28    BL      libSystem.B.dylib::_fork ; 0x1000905a8
   100015b2c    MOV     X19, X0               ; --X19 = X0 = 0x0
   100015b30    CMN     W19, #1                X0 = 0xffffffffffffffff -|
   100015b34    B.EQ    0x100015b60           ;
   100015b38    ADR     X0, #791856           @"fork allowed" ; ->R0 = 0x10
   100015b3c    NOP                           ;
   100015b40    BL      Foundation::_NSLog        ; 0x100090284
; Foundation::_NSLog(@"fork allowed");
; // if (R19  == 0) then goto 0x100015b74
   100015b44    CBZ     X19, 0x100015b74 ;
   100015b48    STR     X19, [SP, #0]             ; *(SP + 0x0) =
   100015b4c    ADR     X0, #791868           @"parent: childid: %d"  ; ->R
   100015b50    NOP                           ;
   100015b54    BL      Foundation::_NSLog        ; 0x100090284
; Foundation::_NSLog(@"parent: childid: %d");
   100015b58    MOVZ    W0, 0x0                   ; ->R0 = 0x0
   100015b5c    B       0x100015b64
   100015b60    ORR     W0, WZR, #0x1             ; ->R0 = 0x1
   100015b64    SUB     X31, X29, #16             ; SP = R29 - 0x10
   100015b68    LDP     X29, X30, [SP,#16]        ;
   100015b6c    LDP     X20, X19, [SP],#32        ;
   100015b70    RET                           ;
   100015b74    MOVZ    W0, 0x0                   ; ->R0 = 0x0
   100015b78    BL      libSystem.B.dylib::__exit         ; 0x1000904dc
```

Call to fork
Return value in X0
Patch CMN W19, #1

IOActive™

Case Study: Bypassing Fork Check

```
SysSecInfo`___lldb_unnamed_function371$$SysSecInfo:
->  0x100111b30 <+24>: cmn    w19, #1
    0x100111b34 <+28>: b.eq   0x100111b60                ; <+72>
    0x100111b38 <+32>: adr    x0, #791856                ; @"fork allowed"
    0x100111b3c <+36>: nop
(lldb) reg read w19
     w19 = 0xffffffff
```



Patch here

IOActive™

Conclusion

- Common bugs being closed
- A “***new”*** approach and break from the norm is required for in depth assessments
- Assembly knowledge a **MUST** for Reversing Engineering
  - Low level assembly allows you to bypass many security protections, discover hidden gems and then some
- Knowledge of iOS architecture will not only improve your assessments but also provide a launching pad for other research
- Disassemblers are your friends (IDA, Hopper, Jtool …..)
- **Add the reverse engineering skillset to your arsenal !!!**

IOActive™

References

- Books:
  - Mac OS X and iOS Internals To the Apple's Core (Jonathan Levin)
  - The Mobile Application Hacker's Handbook (Dominic Chell, Tyrone Erasmus et al.)
  - Hacking and Securing iOS Applications (Jonathan Zdziarski)
  - iOS Application Security: The Definitive Guide for Hackers and Developers (David Thiel)

- Blogs and Tools:
  - processor_set_tasks() - http://newosxbook.com/articles/PST2.html
  - procexp – http://newosxbook.com/tools/procexp.html
  - iOSBinaries - http://newosxbook.com/tools/iOSBinaries.html
  - jtool - http://newosxbook.com/tools/jtool.html
  - filemon - http://newosxbook.com/tools/filemon.html
  - AmIBeingDebugged - https://developer.apple.com/library/mac/qa/qa1361/_index.html
  - Frida - http://www.frida.re/
  - Cycript - http://www.cycript.org/
  - iFunBox - http://www.i-funbox.com/
  - SSL Kill Switch – https://github.com/iSECPartners/ios-ssl-kill-switch
  - BurpSuite - https://portswigger.net/burp/
  - IDA - https://www.hex-rays.com/products/ida/
  - Hopper - https://www.hopperapp.com/
  - Idb - http://www.idbtool.com/
  - PT_DENY_ATTACH - https://www.theiphonewiki.com/wiki/Bugging_Debuggers
  - ARM - http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.den0024a/ch05s01s03.html
  - SQLite-parser - https://github.com/mdegrazia/SQLite-Deleted-Records-Parser
  - SQLite Deletion - http://www.zdziarski.com/blog/?p=6143
  - lsdtrip - http://newosxbook.com/src.jl?tree=listings&file=ls.m#dumpURL

IOActive™