SektionEins
http://www.sektioneins.de

iOS 8: Containers, Sandboxes and Entitlements

Stefan Esser <stefan.esser@sektioneins.de>

Ruxcon 2014 / Melbourne, 11th October 2014

Who am I?

Stefan Esser

- from Cologne / Germany
- in information security since 1998
- PHP core developer from 2001-20xx
- Months of PHP Bugs and Suhosin
- since 2010 focused on iPhone security (ASLR/jailbreak)
- founder of SektionEins GmbH

Disclaimer

- title and abstract did not fully match
- iOS 8 plugins research is work in progress
- this talk will focus on code signing and sandboxes
- plus some extra in the end

Agenda

- iOS Codesigning 101
- Mac Policy Framework / TrustedBSD in iOS
- Sandbox & Sandbox Profiles
- a new mitigation sneaked into iOS 8
- KASLR in iOS 8

iOS Codesigning 101

iOS Codesigning 101

- every executable file on iOS requires code signing information
- stored as binary blobs in mach-o files
- LC_CODE_SIGNATURE with data usually at the end
- code-signing works on per memory page basis
- SHA1 check of memory pages

iOS Codesigning 101 - Codesigning Blobs

- consist of blob directory followed by blobs
- usually
  - SuperBlob
    - Entitlements
    - **optionally a** signature
    - additional optional elements

iOS Codesigning 101 - SuperBlob

- defines name etc… of signed object
- defines from where to where the file is signed
- contains SHA1 hashes
  - for every memory page inside
  - for the other parts (e.g. entitlements)
- signature only signs the SuperBlob

iOS Codesigning 101 - Entitlements

```
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>application-identifier</key>
    <string>com.apple.mobileslideshow</string>
    <key>backupd-connection-initiate</key>
    <true/>
    <key>checklessPersistentURLTranslation</key>
    <true/>
    ...
    <key>com.apple.private.MobileGestalt.AllowedProtectedKeys</key>
    <array>
        <string>EthernetMacAddress</string>
        <string>WifiAddressData</string>
        <string>WifiAddress</string>
        <string>UniqueDeviceID</string>
    </array>
    <key>com.apple.private.accounts.allaccounts</key>
    <true/>
    <key>com.apple.private.accounts.bypassguestmoderestrictions</key>
    <true/>
    ...
    <key>keychain-access-groups</key>
    <array>
        <string>com.apple.youtube.credentials</string>
        <string>com.apple.videouploadplugins.credentials</string>
        <string>apple</string>
        <string>com.apple.airplay</string>
    </array>
    <key>platform-application</key>
    <true/>
    <key>seatbelt-profiles</key>
    <array>
        <string>MobileSlideShow</string>
    </array>
</dict>
</plist>
```

- XML snippets that define non default permissions of applications
- some boolean flags
- some more complicated structures
- keychain access groups
- selected sandbox profiles

iOS Codesigning 101 - Optional Signature

- for all built-in applications there is no digital signature attached
- instead AMFI driver contains SHA1 whitelist of SuperBlobs
- other apps contain digital signature
- signature verification actually performed by user space daemon

Mac Policy Framework / TrustedBSD

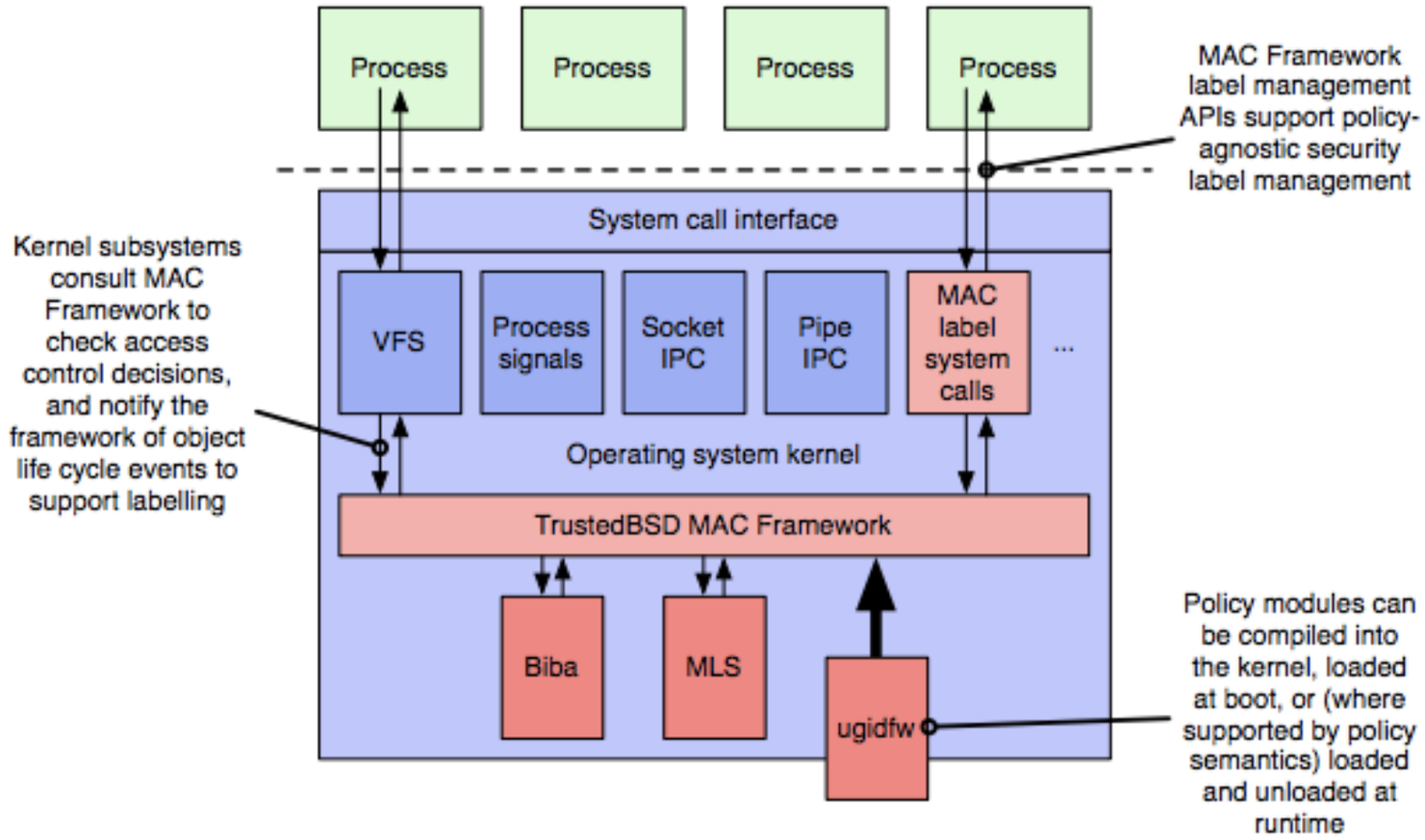
Mandatory Access Control (MAC) Framework

- based on TrustedBSD Mac Framework
- implements mandatory access controls throughout the kernel
- iOS sandbox, entitlement and code-signing security is based on this
- partially documented in

  "New approaches to operating system security extensibility"
  by Robert N. M. Watson

  http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-818.pdf

Mandatory Access Control (MAC) Framework



- the MAC framework and interfaces are inside the XNU source code
- policies come from policy modules inside kernel extensions
- iOS ships with only two modules
    - sandbox - responsible for the whole sandboxing of iOS apps
    - AppleMobileFileIntegrity (AMFI) - responsible for code signatures and entitlement handling

Integration into the Kernel

- throughout the kernel you will see calls from kernel services to the framework
- here is an example from mmap() for protecting MAP_JIT

```
int
mmap(proc_t p, struct mmap_args *uap, user_addr_t *retval)
{
    ...
    if ((flags & MAP_JIT) && ((flags & MAP_FIXED) || (flags & MAP_SHARED) || !(flags & MAP_ANON))){
        return EINVAL;
    }
    ...
    if (flags & MAP_ANON) {

        maxprot = VM_PROT_ALL;
#if CONFIG_MACF
        /*
         * Entitlement check.
         */
        error = mac_proc_check_map_anon(p, user_addr, user_size, prot, flags, &maxprot);
        if (error) {
            return EINVAL;
        }
#endif /* MAC */
```

MAC hooks

- implementation of MAC hooks usually check if policy checking is not disabled
- and then they call the policy modules to check the policy

```
int
mac_proc_check_map_anon(proc_t proc, user_addr_t u_addr,
    user_size_t u_size, int prot, int flags, int *maxprot)
{
    kauth_cred_t cred;
    int error;

    if (!mac_vm_enforce ||
        !mac_proc_check_enforce(proc, MAC_VM_ENFORCE))
        return (0);

    cred = kauth_cred_proc_ref(proc);
    MAC_CHECK(proc_check_map_anon, proc, cred, u_addr, u_size, prot, flags, maxprot);
    kauth_cred_unref(&cred);

    return (error);
}
```

check if vm policy checks are globally disabled

check if vm policy checks are disabled for this process only

MAC Policy Operations

```
/*
 * Policy module operations.
 *
 * Please note that this should be kept in sync with the check assumptions
 * policy in bsd/kern/policy_check.c (policy_ops struct).
 */
#define MAC_POLICY_OPS_VERSION 24 /* inc when new reserved slots are taken */
struct mac_policy_ops {
    mpo_audit_check_postselect_t        *mpo_audit_check_postselect;
    mpo_audit_check_preselect_t     *mpo_audit_check_preselect;

    mpo_bpfdesc_label_associate_t       *mpo_bpfdesc_label_associate;
    mpo_bpfdesc_label_destroy_t     *mpo_bpfdesc_label_destroy;
    mpo_bpfdesc_label_init_t        *mpo_bpfdesc_label_init;
    mpo_bpfdesc_check_receive_t     *mpo_bpfdesc_check_receive;

    mpo_cred_check_label_update_execve_t    *mpo_cred_check_label_update_execve;
    mpo_cred_check_label_update_t       *mpo_cred_check_label_update;
    mpo_cred_check_visible_t        *mpo_cred_check_visible;
    mpo_cred_label_associate_fork_t     *mpo_cred_label_associate_fork;
    mpo_cred_label_associate_kernel_t   *mpo_cred_label_associate_kernel;
    mpo_cred_label_associate_t      *mpo_cred_label_associate;
    mpo_cred_label_associate_user_t     *mpo_cred_label_associate_user;
    mpo_cred_label_destroy_t        *mpo_cred_label_destroy;
    mpo_cred_label_externalize_audit_t  *mpo_cred_label_externalize_audit;
    mpo_cred_label_externalize_t        *mpo_cred_label_externalize;
    mpo_cred_label_init_t           *mpo_cred_label_init;
    mpo_cred_label_internalize_t        *mpo_cred_label_internalize;
    mpo_cred_label_update_execve_t      *mpo_cred_label_update_execve;
    mpo_cred_label_update_t         *mpo_cred_label_update;

    mpo_devfs_label_associate_device_t  *mpo_devfs_label_associate_device;
    mpo_devfs_label_associate_directory_t   *mpo_devfs_label_associate_directory;
    mpo_devfs_label_copy_t          *mpo_devfs_label_copy;
    mpo_devfs_label_destroy_t       *mpo_devfs_label_destroy;
    mpo_devfs_label_init_t          *mpo_devfs_label_init;
    mpo_devfs_label_update_t        *mpo_devfs_label_update;

    mpo_file_check_change_offset_t      *mpo_file_check_change_offset;
    mpo_file_check_create_t         *mpo_file_check_create;
    mpo_file_check_dup_t            *mpo_file_check_dup;
    mpo_file_check_fcntl_t          *mpo_file_check_fcntl;
    ...
```

more than 200 defined MAC policy operations

defined in */security/mac_policy.h*

MAC Policy Operations - Documentation

```
/**
  @brief Access control check for pipe read
  @param cred Subject credential
  @param cpipe Object to be accessed
  @param pipelabel The label on the pipe

  Determine whether the subject identified by the credential can
  perform a read operation on the passed pipe.  The cred object holds
  the credentials of the subject performing the operation.

  @return Return 0 if access is granted, otherwise an appropriate value for
  errno should be returned.

*/
typedef int mpo_pipe_check_read_t(
    kauth_cred_t cred,
    struct pipe *cpipe,
    struct label *pipelabel
);
```

purpose and parameters op operations are well documented in */security/mac_policy.h*

MAC Policy Registration

- new policies get added to the kernel by calling **mac_policy_register()**

```
int mac_policy_register(struct mac_policy_conf *mpc, mac_policy_handle_t *handlep, void *xd);
```

- the policies are defined by the struct **mac_policy_conf**

```
struct mac_policy_conf {
    const char      *mpc_name;      /** policy name */
    const char      *mpc_fullname;      /** full name */
    const char      **mpc_labelnames;   /** managed label namespaces */
    unsigned int         mpc_labelname_count;   /** number of managed label namespaces */
    struct mac_policy_ops   *mpc_ops;       /** operation vector */
    int          mpc_loadtime_flags;    /** load time flags */
    int         *mpc_field_off;     /** label slot */
    int          mpc_runtime_flags; /** run time flags */
    mpc_t            mpc_list;      /** List reference */
    void            *mpc_data;      /** module data */
};
```

Finding iOS Policies

IDA – /Users/sesser/Downloads/iPhone4,1_8.0_12A365/kernelcache.iPhone4,1_8.0_12A365.decrypted

No debugger

Library function | Data | Regular function | Unexplored | Instruction | External symbol

Functions win…

Function name
- sub_80042028
- sub_800420FC
- sub_800421B4
- sub_800422E8
- sub_80042584

IDA … | Pseudo… | Strings … | Hex … | Stru… | E… | I… | E…

```
__TEXT:__text:80329994
__TEXT:__text:80329994 ; Attributes: bp-based frame
__TEXT:__text:80329994
__TEXT:__text:80329994                  EXPORT _mac_policy_register
__TEXT:__text:80329994 _mac_policy_register                     ; DATA XREF: com.apple.driver.AppleMobileFileIntegrity:_
__TEXT:__text:80329994                                          ; com.apple.security.sandbox:__nl_symbol_ptr:80CCC130↓o
__TEXT:__text:80329994
__TEXT:__text:80329994 var_2C          = -0x2C
__TEXT:__text:80329994 var_28          = -0x28
__TEXT:__text:80329994 var_24          = -0x24
__TEXT:__text:80329994 var_20          = -0x20
__TEXT:__text:80329994 var_1C          = -0x1C
```

xrefs to _mac_policy_register

Direction	Type	Address	Text
…	o	com.apple.driver.AppleMobileFileIntegrity:__nl_s…	DCD _mac_policy_register+1
…	o	com.apple.security.sandbox:__nl_symbol_ptr:80…	DCD _mac_policy_register+1

Help | Search | Cancel | OK

Line 1 of 2

```
__TEXT:__text:803299C4                  BL      _panic
```

00328994 80329994: _mac_policy_register

_fre

Output window

```
803E3144: using guessed type int dword_803E3144;
8009ECA8: using guessed type int __fastcall sub_8009ECA8(_DWORD);
803E3160: using guessed type int dword_803E3160;
803E3164: using guessed type int dword_803E3164;
803E3168: using guessed type int dword_803E3168;
803E316C: using guessed type int dword_803E316C;
803F5160: using guessed type int dword_803F5160;
803F5168: using guessed type int dword_803F5168;
Caching 'Choose segment to jump'... ok
```

Python

AU: idle Down Disk: 180GB

AMFI Policy Register

```
77    dword_80731B44 = 0;
78    dword_80731B50 = sub_807228B4(dword_80731B60);
79    AMFI_policy_ops.mpo_vnode_check_signature = (void *(*)(void))sub_80721A4C;
80    AMFI_policy_ops.mpo_vnode_check_exec = (void *(*)(void))sub_80721D84;
81    AMFI_policy_ops.mpo_proc_check_get_task = (void *(*)(void))sub_80721DBC;
82    AMFI_policy_ops.mpo_proc_check_run_cs_invalid = (void *(*)(void))sub_80721E30;
83    AMFI_policy_ops.mpo_cred_label_init = (void *(*)(void))sub_80721EBC;
84    AMFI_policy_ops.mpo_cred_label_associate = (void *(*)(void))sub_80721EE4;
85    AMFI_policy_ops.mpo_cred_check_label_update_execve = (void *(*)(void))sub_80721F28;
86    AMFI_policy_ops.mpo_cred_label_update_execve = (void *(*)(void))&loc_80721F2C;
87    AMFI_policy_ops.mpo_cred_label_destroy = (void *(*)(void))sub_807223B4;
88    AMFI_policy_ops.mpo_r18 = (void *(*)(void))sub_80722474;
89    AMFI_policy_ops.mpo_proc_check_map_anon = (void *(*)(void))sub_80722434;
90    AMFI_policy_ops.mpo_priv_grant = (void *(*)(void))sub_807224A8;
91    AMFI_mac_policy_conf.mpc_name = "AMFI";
92    AMFI_mac_policy_conf.mpc_fullname = "Apple Mobile File Integrity";
93    AMFI_mac_policy_conf.mpc_labelnames = (const char **)&off_80731A60;
94    AMFI_mac_policy_conf.mpc_labelname_count = 1;
95    AMFI_mac_policy_conf.mpc_ops = (struct mac_policy_conf::mac_policy_ops *)&AMFI_policy_ops;
96    AMFI_mac_policy_conf.mpc_loadtime_flags = 0;
97    AMFI_mac_policy_conf.mpc_field_off = &dword_80731B64;
98    AMFI_mac_policy_conf.mpc_runtime_flags = 0;
99    if ( mac_policy_register(&AMFI_mac_policy_conf, &AMFI_policy_handle, 0) )
100   {
101     IOLog("%s: mac_policy_register failed: %d\n", "kern_return_t _initializeAppleMobileFileIntegrity()");
102     panic("\"AMFI mac policy could not be registered!\"");
103   }
104   v2 = dword_80731B58;
105 }
```

AMFI mpo_proc_check_get_task

```
 1 signed int __fastcall sub_80721DBC(int a1, int a2)
 2 {
 3   int v2; // r5@1
 4   signed int v3; // r4@1
 5   char v5; // [sp+0h] [bp-Ch]@1
 6
 7   v2 = a1;
 8   v3 = 0;
 9   v5 = 0;
10   sub_80720E68(a2, "get-task-allow", &v5);
11   if ( !(v5 & 1) )
12   {
13     sub_80720F14(v2, "task_for_pid-allow", &v5);
14     v3 = 0;
15     if ( !(v5 & 1) )
16     {
17       v3 = 0;
18       if ( sub_807225C8() != 1 )
19       {
20         v3 = 1;
21         if ( dword_80731B48 )
22         {
23           IOLog("AMFI: task_for_pid() not allowed\n");
24           v3 = 1;
25         }
26       }
27     }
28   }
29   return v3;
30 }
```

checks for presence of various entitlements

get-task-allow
task_for_pid-allow

```
kern_return_t task_for_pid(struct task_for_pid_args *args)
{
    ...
#if CONFIG_MACF
        error = mac_proc_check_get_task(kauth_cred_get(), p);
        if (error) {
            error = KERN_FAILURE;
            goto tfpout;
        }
#endif
```

Sandbox Policy Register

- unlike AMFI the sandbox extension uses a pre-filled structure from the __DATA segment for registering the policy

```
1 int __fastcall sub_8073D600()
2 {
3   int result; // r0@1
4
5   result = sub_8073FF78();
6   if ( !result )
7     result = _mach_policy_register(&sandbox_policy_conf, algn_80742C34, 0);
8   return result;
9 }
```

Sandbox Policy Conf

```
y.sandbox:__data:80742320                                       ; "sb"
y.sandbox:__data:80742324 ; mac_policy_conf sandbox_policy_conf
y.sandbox:__data:80742324 sandbox_policy_conf mac_policy_conf <aSandbox_0, aSeatbeltSandbo, off_80742320, 1, \ ; "Sandbox"
y.sandbox:__data:80742324                                       stru_8074234C, 0, dword_80742C38, 0, 0, 0>
y.sandbox:__data:8074234C ; struct mac_policy_ops stru_8074234C
y.sandbox:__data:8074234C stru_8074234C   mac_policy_ops <0, 0, 0, 0, 0, 0, \
y.sandbox:__data:8074234C                                       ; DATA XREF: com.apple.security.sandbox:__data:sandbox_p
y.sandbox:__data:8074234C                                     sandbox_mpo_cred_check_label_update_execve+1, \
y.sandbox:__data:8074234C                                     sandbox_mpo_cred_check_label_update+1, 0, 0, 0, \
y.sandbox:__data:8074234C                                     sandbox_mpo_cred_label_associate+1, 0, \
y.sandbox:__data:8074234C                                     sandbox_mpo_cred_label_destroy+1, 0, 0, 0, 0, \
y.sandbox:__data:8074234C                                     sandbox_mpo_cred_label_update_execve+1, \
y.sandbox:__data:8074234C                                     sandbox_mpo_cred_label_update+1, 0, 0, 0, 0, 0, 0, 0, \
y.sandbox:__data:8074234C                                     sandbox_mpo_cred_label_update_execve+1, \
y.sandbox:__data:8074234C                                     sandbox_mpo_cred_label_update+1, 0, 0, 0, 0, 0, 0, 0, \
y.sandbox:__data:8074234C                                     0, 0, sandbox_mpo_file_check_fcntl+1, 0, 0, 0, 0, 0, \
y.sandbox:__data:8074234C                                     0, 0, 0, sandbox_mpo_file_check_set+1, 0, 0, 0, 0, 0, \
y.sandbox:__data:8074234C                                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \
y.sandbox:__data:8074234C                                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \
y.sandbox:__data:8074234C                                     0, 0, 0, 0, sandbox_mpo_mount_check_fsctl+1, 0, 0, \
y.sandbox:__data:8074234C                                     sandbox_mpo_mount_check_mount+1, \
y.sandbox:__data:8074234C                                     sandbox_mpo_mount_check_remount+1, 0, 0, \
y.sandbox:__data:8074234C                                     sandbox_mpo_mount_check_umount+1, 0, 0, 0, 0, 0, 0, 0,\
y.sandbox:__data:8074234C                                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \
y.sandbox:__data:8074234C                                     sandbox_mpo_policy_init+1, \
y.sandbox:__data:8074234C                                     sandbox_mpo_policy_initbsd+1, \
y.sandbox:__data:8074234C                                     sandbox_mpo_policy_syscall+1, 0, 0, 0, 0, 0, 0, 0, 0, \
y.sandbox:__data:8074234C                                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \
```

Sandbox & Sandbox Profiles

Apple Sandbox

- closed source sandboxing
- reversed by Dionysus Blazakis
- great paper “The Apple Sandbox” in January 2011
- later chapter in iOS Hacker’s Handbook
- released demos, scripts and tools
- AFAIK no public research on anything newer than iOS 5

Sandbox Profiles

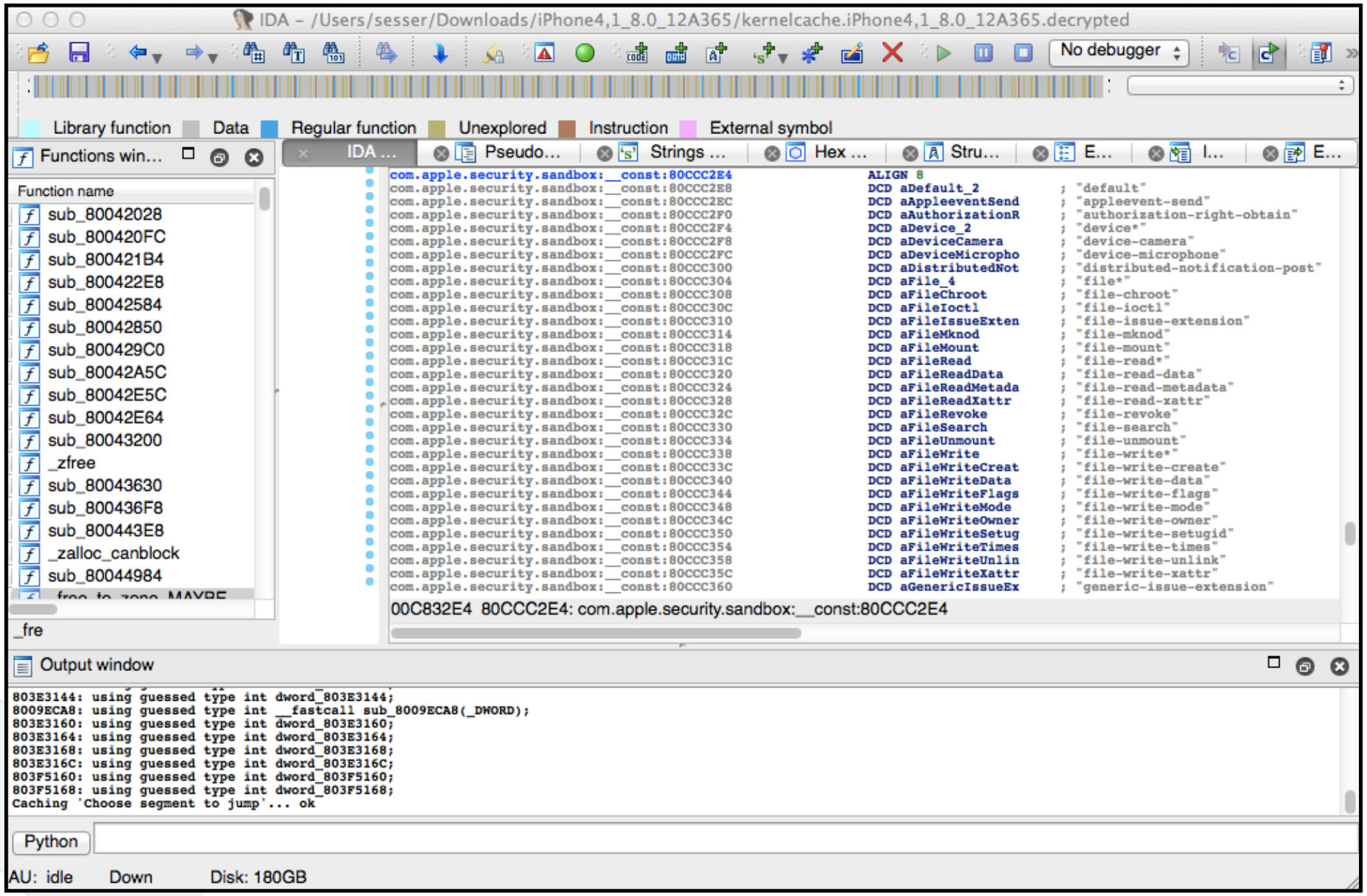
```
;;
;; syslogd - sandbox profile
;; Copyright (c) 2007 Apple Inc.  All Rights reserved.
;;
;; WARNING: The sandbox rules in this file currently constitute
;; Apple System Private Interface and are subject to change at any time and
;; without notice. The contents of this file are also auto-generated and not
;; user editable; it may be overwritten at any time.
;;
(version 1)
(debug deny)

(import "bsd.sb")

(deny default)

(allow file-write* file-read-data file-read-metadata
    (regex #"^(/private)?/var/run/syslog$"
           #"^(/private)?/var/run/syslog\.pid$"
           #"^(/private)?/var/run/asl_input$"))

(allow file-write* file-read-data file-read-metadata
    (regex #"^(/private)?/dev/console$"
           #"^(/private)?/var/log/.*\.log$"
           #"^(/private)?/var/log/asl\.db$"))

(allow file-read-data file-read-metadata
    (regex #"^(/private)?/dev/klog$"
           #"^(/private)?/etc/asl\.conf$"
           #"^(/private)?/etc/syslog\.conf$"
           #"^/usr/lib/asl/.*\.so$"))
(allow mach-lookup (global-name "com.apple.system.notification_center"))
```

Sandbox operations

Sandbox Profiles

- iOS does not come with plaintext profiles
- profiles are binary blobs without any kind of documentation
- built-in blobs are hidden in Sandbox daemon
- decompilation requires extracting system specific operation names
- current iOS uses different binary format from what Dion reversed

Extracting Sandbox Operations



- sandbox kernel extension contains list of all sandbox operations
- could be different with every new kernel version

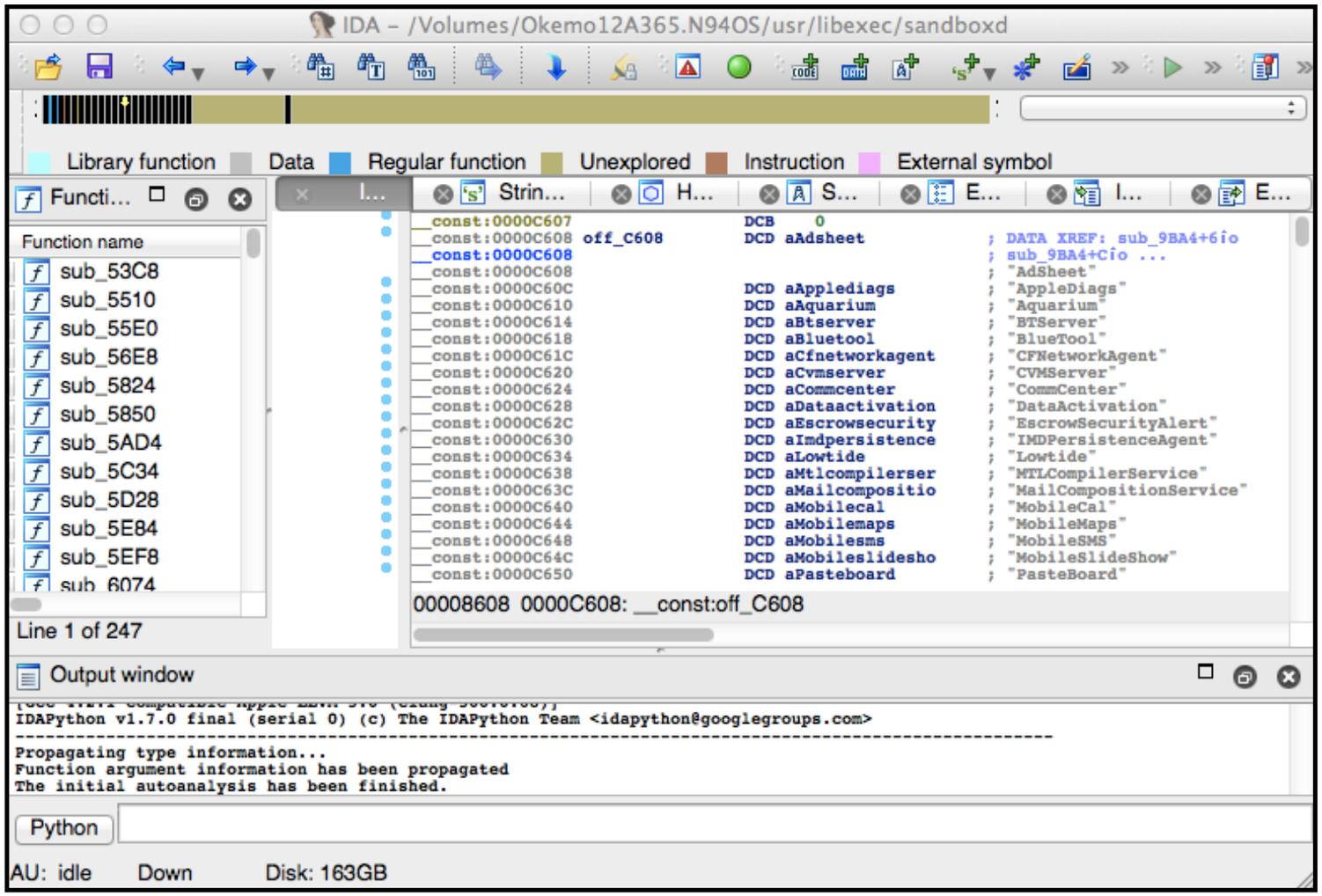
Sandbox Operations in iOS 8.0

default
appleevent-send
authorization-right-obtain
device*
device-camera
device-microphone
distributed-notification-post
file*
file-chroot
file-ioctl
file-issue-extension
file-mknod
file-mount
file-read*
file-read-data
file-read-metadata
file-read-xattr
file-revoke
file-search
file-unmount
file-write*
file-write-create
file-write-data
file-write-flags
file-write-mode
file-write-owner
file-write-setugid
file-write-times
file-write-unlink
file-write-xattr
generic-issue-extension
qtn-user
qtn-download
qtn-sandbox
hid-control
iokit*
iokit-issue-extension
iokit-open
iokit-set-properties
iokit-get-properties
ipc*
ipc-posix*
ipc-posix-issue-extension
ipc-posix-sem
ipc-posix-shm*
ipc-posix-shm-read*
ipc-posix-shm-read-data
ipc-posix-shm-read-metadata
ipc-posix-shm-write*
ipc-posix-shm-write-create
ipc-posix-shm-write-data
ipc-posix-shm-write-unlink
ipc-sysv*
ipc-sysv-msg
ipc-sysv-sem
ipc-sysv-shm
job-creation
load-unsigned-code
lsopen
mach*
mach-bootstrap
mach-issue-extension
mach-lookup
mach-per-user-lookup
mach-priv*
mach-priv-host-port
mach-priv-task-port
mach-register
mach-task-name
network*
network-inbound
network-bind
network-outbound
user-preference*
user-preference-read
user-preference-write
process*
process-exec*
process-exec-interpreter
process-fork
process-info*
process-info-listpids
process-info-pidinfo
process-info-pidfdinfo
process-info-pidfileportinfo
process-info-setcontrol
process-info-dirtycontrol
process-info-rusage
pseudo-tty
signal
sysctl*
sysctl-read
sysctl-write
system*
system-acct
system-audit
system-chud
system-debug
system-fsctl
system-info
system-kext*
system-kext-load
system-kext-unload
system-lcid
system-mac-label
system-nfssvc
system-privilege
system-reboot
system-sched
system-set-time
system-socket
system-suspend-resume
system-swap
system-write-bootstrap

114 operations

Extracting Sandbox Profiles

- binary profiles in ***/usr/libexec/sandboxd***
- ***__const*** section contains built in profile names
- table of name followed by pointers to offset + length of profiles
- extracting script by Dion can be adjusted to work on iOS 8



Builtin Sandbox Profiles in iOS 8.0

AdSheet
AppleDiags
Aquarium
BTServer
BlueTool
CFNetworkAgent
CVMServer
CommCenter
DataActivation
EscrowSecurityAlert
IMDPersistenceAgent
Lowtide
MTLCompilerService
MailCompositionService
MobileCal
MobileMaps
MobileSMS
MobileSlideShow
PasteBoard
Stocks
StreamingUnzipService
WebSheet
accessoryd
afcd
apsd
cloudphotod
com.apple.AssetCacheLocatorService
com.apple.GSSCred
com.apple.WebKit.Databases
com.apple.WebKit.Networking
com.apple.WebKit.WebContent
com.apple.assistant.assistantd
com.apple.bird
com.apple.cloudd
com.apple.datadetectors.AddToRecentsService
com.apple.discoveryd
com.apple.nehelper
com.apple.nesessionmanager
com.apple.quicklook.QLThumbnailsService
com.apple.rtcreportingd
com.apple.sandboxd
com.apple.snhelper
com.apple.tccd
com.apple.tzlinkd
com.apple.ubd
com.apple.xpcd
container
coresymbolicationd
cplogd
dataaccessd
debugserver
deleted
fmfd
ftp-proxy-embedded
gamed
geocorrectiond
geod
gputoolsd
healthd
iapd
keyboard
librariand
limitadtrackingd
lockdownd
mDNSResponder
mediaserverd
mobile-house-arrest
mobileassetd
nfcd
nlcd
nointernet
nsurlsessiond
nsurlstoraged
passd
pfd
printd
ptpd
quicklookd
racoon
reversetemplated
revisiond
routined
seld
sharingd
softwareupdated
streaming_zip_conduit
suggestd
syncdefaultsd
transitd
userfs_helper
userfsd
vibrationmanagerd
vpn-plugins
webinspectord
wifiFirmwareLoader

95 builtin profiles

Decompiling Sandbox Profiles

- public tools by Dion will fail on iOS 8 kernels
- and their output was more a helper than a decompiler
- Apple made some smaller changes to overall file format
- regular expressions binary format very different
- reversed the new format by trial and error

➡ goal a full profile decompiler

Sandbox Profile Binary Format

file header

- starts with unknown 2 bytes
- regular expression table start offset
- regular expression count
- offset to operation node offset table
- all offsets are file offset divided by 8

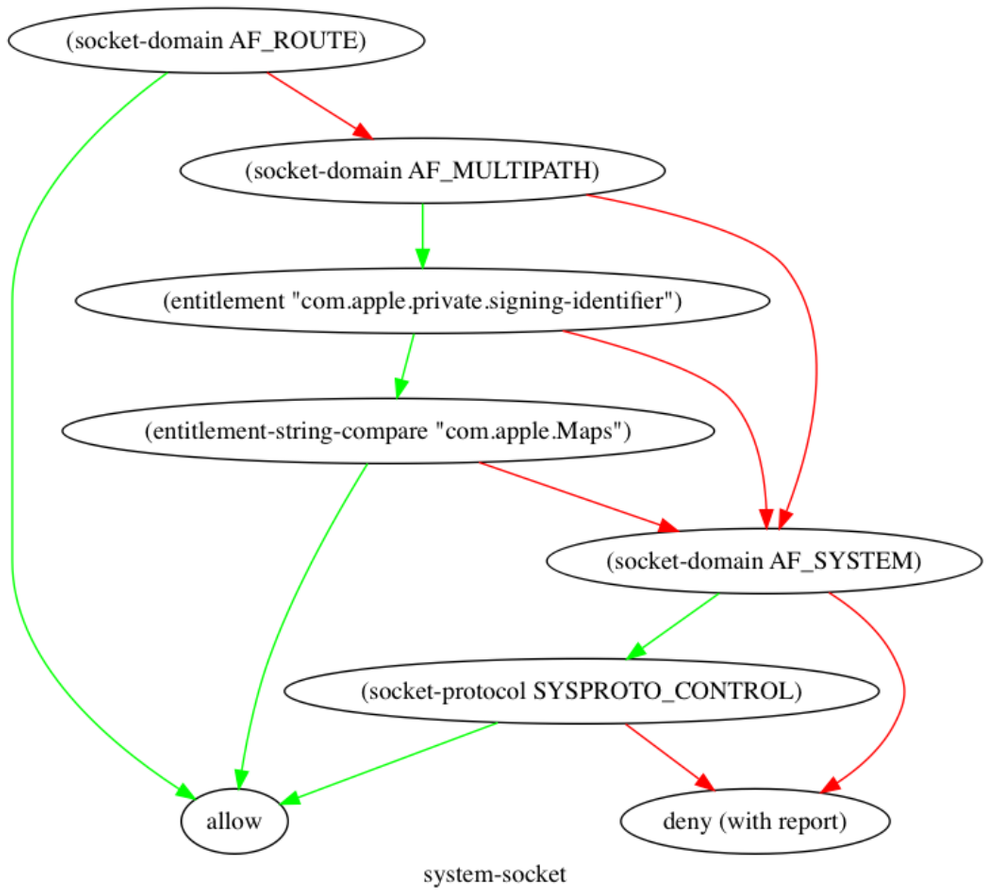
```
struct sb_profile_header {
    uint16_t unknown; /* seems to be 0 */
    uint16_t re_table_offset;
    uint16_t re_table_count;
    uint16_t op_table[SB_OP_TABLE_COUNT];
};
```

Sandbox Profile Binary Format

nodes

- are either decision or result nodes (marked by tag)
- result nodes specify allow / deny
- decision nodes specify filter decisions

```
struct node {
    uint8_t tag;
    union {
        struct result terminal;
        struct decision filter;
        uint8_t raw[7];
    } u;
}
```

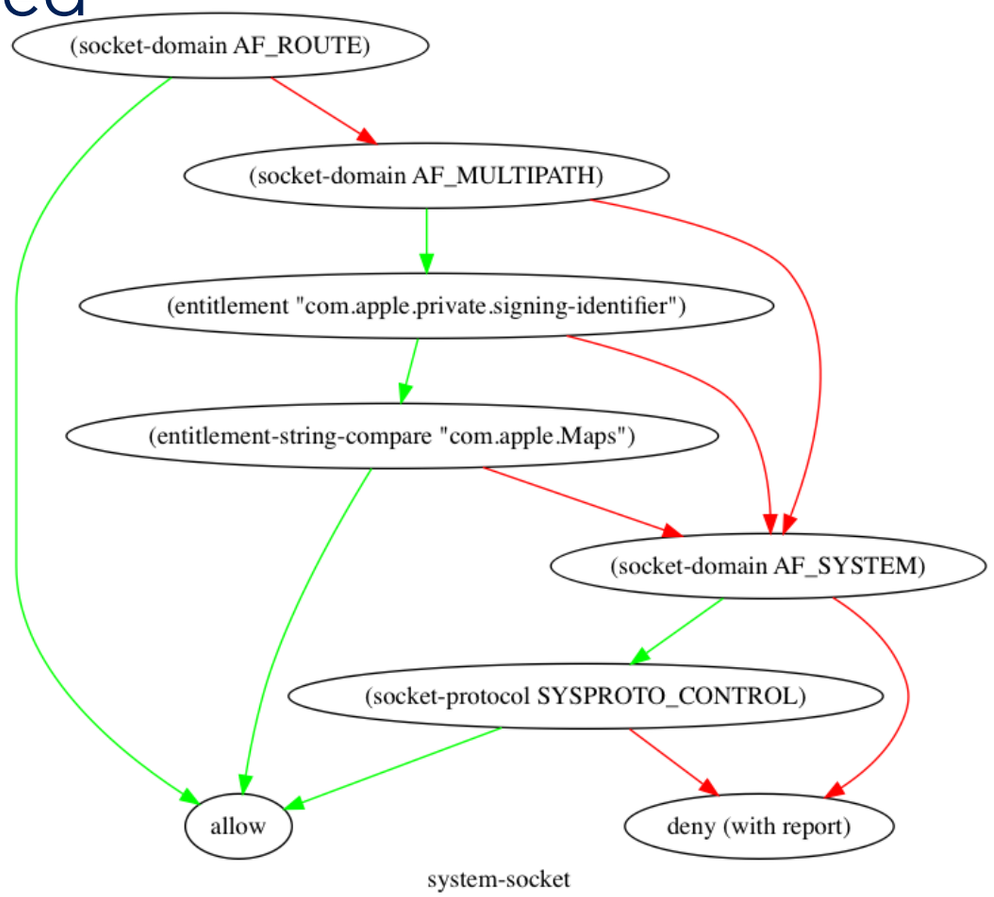


Sandbox Profile Binary Format

result node

- bit 0 decides if allow (1) or deny (0)
- other bits are modifiers
    - if bit 2 is set this result will be logged

```
struct result {
    uint8_t padding;
    uint16_t allow_or_deny;
}
```

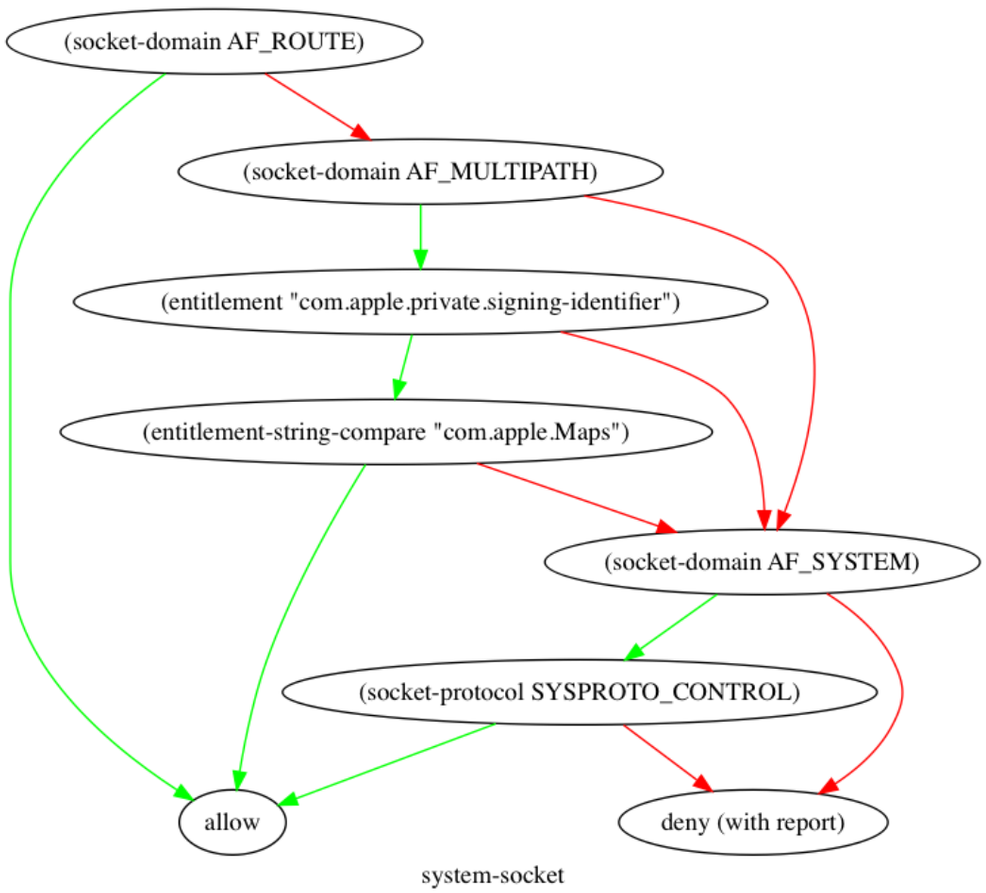


Sandbox Profile Binary Format

decision nodes

- filter type
- filter argument (number, enum, offset to string, number of regex)
- offset of next decision nodes in decision graph if filter matches or not

```
struct decision {
    uint8_t type;
    uint16_t arg;
    uint16_t match_next;
    uint16_t nomatch_next;
}
```



Sandbox Profile Binary Format

filters *(very incomplete list)*

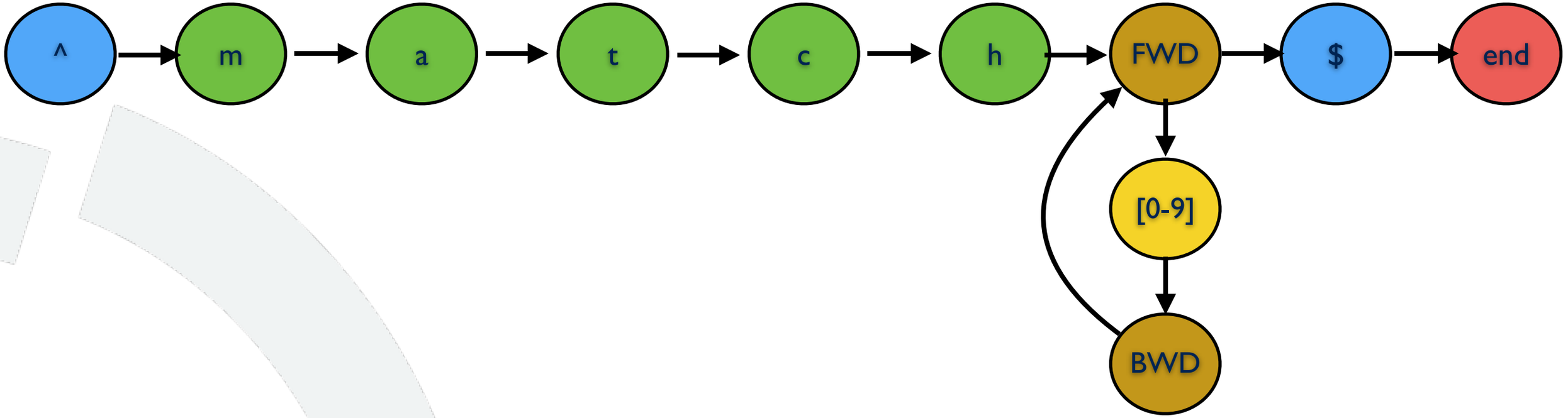
- `0x01 literal`
- `0x81 regex`
- `0x02 mount-relative`
- `0x82 mount-relative-regex`
- `0x0e target`
- `0x11 iokit-user-client-class`
- `0x14 device-major`
- `0x1c preference-domain`
- `0x1e require-entitlement`
- `0x21 kext-bundle-id`
- `...`

nowadays many more filters
that Dion either did not support in his tools
or that were introduced since then

numbering of filters also changed

Regular Expression Graphs

- example regular expression
    - ^match[0-9]*$



Sandbox Profile Binary Format

regular expression encoding
completely different from before

regular expressions *(graphs)*

- `0x02 uint8_t` character to match
- `0x?b uint8_t * 2 * ?` character class
- `0x09` any character `.`
- `0x19` anchor to beginning `^`
- `0x29` anchor to end `$`
- `0x1f uint8_t` end of graph
- `0x?a uint16_t` jump backward (offset from start)
- `0x2f uint16_t` jump forward (offset from start)

iOS 8 Sandbox Decompilation Quirks

regex parser

- [^…] negative character classes are compiled to character classes with wrap around - decompilation does not show this yet
- parser of graph of regular expression doesn't handle + / ? at the moment

filter parser

- require-all / require-any not fully working, yet

- **Addition: 29. Nov 2014 - most of these quirks fixed**

iOS 8 Sandbox Profile Decompilation Tool

- will be released as open source
- requires some fixes for known problems
- will be released on our company's github in next 2 weeks
- github.com/sektioneins

- **Addition: 29. Nov 2014 - Release will be for Christmas 2014**

queue handling in iOS 8

queue in Mach Kernel

- Mach kernel comes with queue implementation
- defined in */osfmk/kern/queue.h*
- used in IOKit and Mach part of kernel
- queue implemented as double linked list

```
struct queue_entry {
      struct queue_entry    *next;          /* next element */
      struct queue_entry    *prev;          /* previous element */
};

typedef struct queue_entry  *queue_t;
typedef struct queue_entry  queue_head_t;
typedef struct queue_entry  queue_chain_t;
typedef struct queue_entry  *queue_entry_t;
```

queue in Zone Allocator

- with iOS 7 Apple has added zone allocator page meta data
- page meta data is kept in double linked list using four queues
- one queue for each type of page full, empty, intermediate, foreign

```
struct zone_page_metadata {
     queue_chain_t              pages;
     struct zone_free_element *elements;
     zone_t                     zone;
     uint16_t                   alloc_count;
     uint16_t                   free_count;
};
```

Unsafe queue Operations

- queue operations not protected in iOS <= 7.1.x
- unlink operation vulnerable to memory corruptions
- iOS zone allocator became a feast for unlink exploits in iOS 7

```
static __inline__ void
remqueue(
        queue_entry_t  elt)
{
        elt->next->prev = elt->prev;
        elt->prev->next = elt->next;
        __DEQUEUE_ELT_CLEANUP(elt);
}
```

Safe Unlink for Mach Queues

- with iOS 8 beta 5 Apple sneaked in queue hardening
- added safe unlink check to **remqueue**() operation
- added **NULL** checks against queue head and elements
- violations will trigger **kernel panics**

```
static __inline__ void
remqueue(
	queue_entry_t	elt)
{
	if (!elt) panic("Invalid queue element %p", elt);

	if (!elt->next || !elt->prev)
	   panic("Invalid queue element pointers for %p: next %p prev %p", elt->next, elt->prev);

	if (elt->next->prev != elt || elt->prev->next != elt)
	   panic("Invalid queue element linkage for %p: next %p next->prev %p prev %p "
	    "prev->next %p", elt, elt->next, elt->next->prev, elt->prev, elt->prev->next);

	elt->next->prev = elt->prev;
	elt->prev->next = elt->next;
	__DEQUEUE_ELT_CLEANUP(elt);
}
```

kext_request() in iOS 8

kext_request()

- Mach API call
- allows applications to request information about kernel modules
- active operations are locked down *(load, unload, start, stop, …)*
- passive operations partially working from even within the sandbox
- Apple unslides load addresses to protect against KASLR leaks

kext_request() - Get Loaded Kext Info

- of special interest is a sub request called
  - ***`Get Loaded Kext Info`***
- returns a serialised dictionary with information about all loaded Kext
- information contained includes the mach-o headers
- Apple even modifies those headers to protect KASLR

kext_request() - Get Loaded Kext Info

```
mach_msg_type_number_t reqlen, resplen = 0, loglen = 0;
char *request, *response = NULL, *log = NULL;
kern_return_t kr;

request =
  "<dict><key>Kext Request Predicate</key><string>Get Loaded Kext Info</string></dict>";
reqlen = strlen(request) + 1;

kext_request(mach_host_self(), 0, request, reqlen,
                                  &response, &resplen,
                                  &log, &loglen, &kr);
```

kext_request() - iPhone 6plus / iOS 8.0.2 Output


```
<dict ID="0"><key>__kernel__</key><dict ID="1"><key>OSBundleMachOHeaders</key><data ID="2">z/rt/gwAAAEAAAAAAgAAAA
8AAABACwAAAQAgAAAAAAAZAAAAOAEAAF9fVEVYVAAAAAAAAAAAAAAAAIAACgP///wAgSAAAAAAAAAAAAAAAAAAIEgAAAAAAAUAAAAFA
AAAAwAAAAAAAAABfX3RleHQAAAAAAAAAAAAAX19URVhUAAAAAAAAAAAAAAAwAAKA////rCJCAAAAAAAAEAAADAAAAAAAAAAAAAAAAAQAgAAAAAA
AAAAAAAAAF9fY29uc3QAAAAAAAAAAABfX1RFWFQAAAAAAAAAAAAAwFJCAoD///9oYAIAAAAAAMAyQgAFAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAX19jc3RyaW5nAAAAAAAAAAF9fVEVYVAAAAAAAAAAAAAAos0QCgP///2h
+AwAAAAAAKJNEAAAAAAAAAAAAAAAAAAAAIAAAAAAAAAAAAAAAAAAAAAAAAZAAAAyAIAAF9fREFUQQAAAAAAAAAAAAAAQEgCgP///
wBACwAAAAAAACBIAAAAAAAAwAQAAAAAAAAMAAAADAAAACAAAAAAAAABfX21vZF9pbml0X2Z1bmMAX19EQVRBAAAAAAAAAAAAAABASAKA////
CAIAAAAAAAAAIEgAAwAAAAAAAAAAAAAACQAAAAAAAAAAAAAAAAAAAF9fbW9kX3Rlcm1fZnVuYwBfX0RBVEEAAAAAAAAAAAAAACEJIAoD///
8AAgAAAAAAAAgiSAADAAAAAAAAAAAAAAAAKAAAAAAAAAAAAAAAAAAAAAX19jb25zdAAAAAAAAAAAF9fREFUQQAAAAAAAAAAAAAQREgCgP///
yCWAQAAAAAAECRIAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABfX2RhdGEAAAAAAAAAAAAAX19EQVRBAAAAAAAAAAAAAAAAAASgKA////
aNICAAAAAAAA4EkADgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAF9fc2ZpX2NsYXNzX3JlZwBfX0RBVEEAAAAAAAAAAAAAAaNJMAoD///
8AAgAAAAAAAGiyTAADAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAX19zeXNjdGxfc2V0AAAAAAF9fREFUQQAAAAAAAAAAAAAABo1EwCgP///
3AcAAAAAAAAaLRMAAMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABfX2JzcwAAAAAAAAAAAAAX19EQVRBAAAAAAAAAAAAAAAAAATQKA////
kF8GAAAAAAAAAADAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAF9fY29tbW9uAAAAAAAAAAABfX0RBVEEAAAAAAAAAAAAAAAGBTAoD///
8YEQAAAAAAAAAAAAMAAAAAAAAAAAAAAAABAAAAAAAAAAAAAAAAAAAAAGQAAACgCAABfX0tMRAAAAAAAAAAAAAAAAAAAAIBTAoD///
8AIAAAAAAAAADwTAAAAAAAACAAAAAAAAADAAAAAwAAAAYAAAAAAAAAAX190ZXh0AAAAAAAAAAAAAF9fS0xEAAAAAAAAAAAAAAAAAAgFMCgP///
1ASAAAAAAAAPBMAAIAAAAAAAAAAAAAAAAAEAIAAAAAAAAAAAAAAAAAABfX2NzdHJpbmcAAAAAAAAAAX19LTEQAAAAAAAAAAAAAAAAFCSUwKA////
CAcAAAAAAAABQAk0AAAAAAAAAAAAAAAAAAAAgAAAAAAAAAAAAAAAAAAAAF9fY29uc3QAAAAAAAAAAABfX0tMRAAAAAAAAAAAAAAAAAAWJlTAoD///
9oAAAAAAAAAFgJTQADAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAX19tb2RfaW5pdF9mdW5jAF9fS0xEAAAAAAAAAAAAAAAAAAADAmVMCgP///
wgAAAAAAAAAwAlNAAMAAAAAAAAAAAAAAAAkAAAAAAAAAAAAAAAAAAAABfX21vZF90ZXJtX2Z1bmMAX19LTEQAAAAAAAAAAAAAAAAMiZUwKA////
CAAAAAAAAADICU0AAwAAAAAAAAAAAAAACgAAAAAAAAAAAAAAAAAAAF9fYnNzAAAAAAAAAAAAAABfX0tMRAAAAAAAAAAAAAAAAA0JlTAoD///
8BAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABAAAAAAAAAAAAAAAAAAAAAGQAAAOgAAABfX0xBU1QAAAAAAAAAAAAAAAAAAKBTAoD///
8AEAAAAAAAAAQTQAAAAAAAABAAAAAAAAADAAAAAwAAAAIAAAAAAAAAAX19tb2RfaW5pdF9mdW5jAF9fTEFTVAAAAAAAAAAAAAAAAAAoFMCgP///
wgAAAAAAAAABBNAAMAAAAAAAAAAAAAAAAkAAAAAAAAAAAAAAAAAAAAABfX2xhc3QAAAAAAAAAAAAAX19MQVNUAAAAAAAAAAAAAAAAAig8xKA////
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAABkAAACYAAAAX19QUkVMSU5LX1RFWFQAAAAwWQKA////
AJDTAAAAAAAAoFIAAAAAAAACQ0wAAAAAAAAwAAAAMAAAAABAAAAAAAAAAF9fdGV4dAAAAAAAAAAAAAABfX1BSRUxJTktfVEVYVAAAAADBZAoD///
8AkNMAAAAAAAACgUgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGQAAAOgAAABfX1BSRUxJTktfU1RBVEUAAALDzEoD///
8AAAAAAAAAAAgTQAAAAAAAAAAAAAAAAAAAADAAAAAwAAAAIAAAAAAAAAAX19rZXJuZWwAAAAAAAAAAAF9fUFJFTElOS19TVEFURQAAsPMSgP///
wAAAAAAAAAAACBNAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABfX2tleHRzAAAAAAAAAAAAAX19QUkVMSU5LX1NUQVRFAACw8xKA////
AAAAAAAAAAAAAIE0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABkAAACYAAAAX19QUkVMSU5LX0lORk8AAADAzBOA////
AFAIAAAAAAAAAAMCYBAAAAAAPhPCAAAAAAAAAAwAAAAMAAAAABAAAAAAAAAAF9faW5mbwAAAAAAAAAAAAAABfX1BSRUxJTktfSU5GTwAAAMDME4D////
4TwgAAAAAAAAAAwJgEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGQAAAEgAAABfX0xJTktFRElUAAAAAAAAAAALDzEoD///
94eQUAAAAAAAAAgTQAAAAAAAeHkFAAAAAAAAAAABAAAAAAQAAAAAAAAAAAAAAAAAAAgAAABgAAADwnU8AnhAAANCnUACo8QEACwAAAFAAAAAAAAAAAAAAAAAAAAAAACeEAAAn
hAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACBNAPVHAAAbAAAAGAAAAAPke8gf2KzpJlbujQoICCjclAAAAAEAAAAAAAA
...AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAWIANAoD///8AAAAAAAAAAACYAAAAQAAAAqF9PAEg+AAA=</data><key>OSBundleCPUType</
key><integer size="32" ID="3">0x100000c</integer><key>OSBundleCPUSubtype</key><integer size="32" ID="4">0x0</
integer><key>CFBundleIdentifier</key><string ID="5">__kernel__</string><key>CFBundleVersion</key><string
ID="6">14.0.0</string><key>OSBundleUUID</key><data ID="7">+R7yB/YrOkmVu6NCggIKNw==</data><key>OSKernelResource</
key><true/><key>OSBundleIsInterface</key><false/><key>OSBundlePrelinked</key><false/><key>OSBundleStarted</key><true/
><key>OSBundleLoadTag</key><integer size="32" ID="8">0x0</integer><key>OSBundleLoadAddress</key><integer size="64"
ID="9">0xffffff8002002000</integer><key>OSBundleLoadSize</key>
```

kext_request() - Mach-o Headerdump

```
Load command 0
      cmd LC_SEGMENT_64
  cmdsize 312
  segname __TEXT
   vmaddr 0xffffff8002002000
   vmsize 0x0000000000482000
  fileoff 0
 filesize 4726784 (past end of file)
  maxprot r-x
 initprot r-x
   nsects 3
    flags (none)
Section
  sectname __text
   segname __TEXT
      addr 0xffffff8002003000
      size 0x00000000004222ac
    offset 4096 (past end of file)
     align 2^12 (4096)
    reloff 0
    nreloc 0
      type S_REGULAR
attributes PURE_INSTRUCTIONS SOME_INSTRUCTIONS
 reserved1 0
 reserved2 0
Section
  sectname __const
   segname __TEXT
      addr 0xffffff80024252c0
      size 0x0000000000026068
    offset 4338368 (past end of file)
     align 2^5 (32)
    reloff 0
    nreloc 0
```

- mach-o dump after base64 decode
- addresses have KASLR slide removed
- in **iOS <= 6.0** Apple forgot to unslide section headers (disclosed by **Mark Dowd** in **October 2012**)
- fixed in **iOS 6.0.1**

kext_request() - fix for Mark Dowd bug

```
lcp = (struct load_command *) (temp_kext_mach_hdr + 1);
for (i = 0; i < temp_kext_mach_hdr->ncmds; i++) {
    if (lcp->cmd == LC_SEGMENT_KERNEL) {
        kernel_segment_command_t *  segp;
        kernel_section_t *          secp;

        segp = (kernel_segment_command_t *) lcp;
        // 10543468 - if we jettisoned __LINKEDIT clear size info
        if (flags.jettisonLinkeditSeg) {
            if (strncmp(segp->segname, SEG_LINKEDIT, sizeof(segp->segname)) == 0) {
                segp->vmsize = 0;
                segp->fileoff = 0;
                segp->filesize = 0;
            }
        }

        segp->vmaddr = VM_KERNEL_UNSLIDE(segp->vmaddr);

        for (secp = firstsect(segp); secp != NULL; secp = nextsect(segp, secp)) {
            secp->addr = VM_KERNEL_UNSLIDE(secp->addr);
        }
    }
    lcp = (struct load_command *)((caddr_t)lcp + lcp->cmdsize);
}
```

fix unslides sections

kext_request() - Fixed? But wait ...

```
Load command 4
      cmd LC_SEGMENT_64
  cmdsize 152
  segname __PRELINK_TEXT
   vmaddr 0xffffff8002593000
   vmsize 0x0000000000d39000
  fileoff 5414912 (past end of file)
 filesize 13864960 (past end of file)
  maxprot rw-
 initprot rw-
   nsects 1
    flags (none)
Section
  sectname __text
   segname __PRELINK_TEXT
      addr 0xffffff8002593000
      size 0x0000000000d39000
    offset 5414912 (past end of file)
     align 2^0 (1)
    reloff 0
    nreloc 0
      type S_REGULAR
attributes (none)
 reserved1 0
 reserved2 0
Load command 5
      cmd LC_SEGMENT_64
  cmdsize 232
  segname __PRELINK_STATE
   vmaddr 0xffffff8012f3b000
   vmsize 0x0000000000000000
  fileoff 5054464 (past end of file)
 filesize 0 (past end of file)
  maxprot rw-
 initprot rw-
```

- section addresses **now protected**
- but some **segments** / **sections** are **still showing slid pointers**
- code looked fixed, so **why?**
- because there is **additional hidden bug**

kext_request() - VM_KERNEL_UNSLIDE()

```
#define VM_KERNEL_IS_SLID(_o)                                        \
            (((vm_offset_t)(_o) >= vm_kernel_base) &&    \
             ((vm_offset_t)(_o) <  vm_kernel_top))

#define VM_KERNEL_IS_KEXT(_o)                                         \
                  (((vm_offset_t)(_o) >= vm_kext_base) && \
                   ((vm_offset_t)(_o) <  vm_kext_top))

#define VM_KERNEL_UNSLIDE(_v)                                        \
            ((VM_KERNEL_IS_SLID(_v) ||                               \
              VM_KERNEL_IS_KEXT(_v)) ?                               \
                (vm_offset_t)(_v) - vm_kernel_slide :    \
                (vm_offset_t)(_v))
```

- **VM_KERNEL_UNSLIDE()** does **only unslide** main **kernel** and **pre-loaded kernel extensions**
- but there are **parts** of kernel binary **in between** that are **not unslid**
- leaks **KASLR** slide

kext_request() - VM_KERNEL_UNSLIDE() Infoleak

- **easily seen** from **mach-o header** dump
- most likely discovered by **multiple parties** after **October 2012**
- **proof** of it being **known in the wild** since **2013**
- also **part of my iOS kernel exploitation training** material
- used in **Pangu** jailbreak by my **chinese trainees**

Unfixed?

- **yes you heard right !!!**
- bug was **used in public** jailbreak in **June 2014**
- is **most probably known** to lots of parties **since 2012**
- **Apple choose not to fix it** for new major iOS release
- at the moment **KASLR mitigation in iOS 8 is worthless**

- **Addition: 29. Nov 2014 - TAIG has just released an iOS 8.1.1 jailbreak that uses this bug to break KASLR**

Questions ?

want more?

there will be a free whitepaper later this year

there are still free slots in our November iOS kernel exploitation training

http://www.sektioneins.de/