Fried Apples:
Jailbreak DIY

Max Bazaliy

Alex Hude

Vlad Putin

March 28-31, 2017

black hat

Who we are ?

- Security research group
- Focused on hardware and software exploitation
- Made a various jailbreaks for iOS, tvOS, watchOS
- Contributors to jailbreak community

iOS Security Overview

- Secure Boot Chain
- Mandatory Code Signing
- Sandbox
- Exploit Mitigations
- Data Protection
- Secure Enclave Processor

What is jailbreak ?

- Disable OS restrictions
- Gain full access to device
- Install 3-rd party tools and apps
- Exploit chain required

Jailbreak types

- Tethered
  - Re-exploit device on each boot manually
- Untethered
  - Re-exploit device on each boot automatically

Initial attack vector strategies

- Application archive (IPA) based
- USB payload based
- WebKit\SMS\baseband based

Making jailbreak if you have bugs

- Write an exploit chain
- Patch OS security restrictions
- Install persistent binary
- Add Cydia\ssh\remote shell

Making jailbreak if you don't have bugs

- ~~Write an exploit chain~~ Use public write-ups
- Patch OS security restrictions
- Install persistent binary
- Add Cydia\ssh\remote shell

Implementation

Arbitrary code execution strategies

- ROP
- Binary with Mach-O bug
- JavaScriptCore JIT region
- Sign with dev\ent certificate

Bypassing sandbox strategies

- TOCTOU \ Symlinks
- XPC
- Kernel patch

Escalating privileges strategies

- Code injection in system service
- Kernel patch

Bypassing KASLR strategies

- Information leak
- Brute force

Bypassing DEP strategies

- JavaScriptCore JIT
- Userland mmap\mprotect bug
- Kernel patch
- ROP chain

Seeking for patches in kernel

- Static patchfinder (memmem)
  - memmem string\pattern, xref + instruction analysis
- Dynamic patchfinder
  - syscall, sysctl, mach location, known structs + emulation

Kernel patches in detail

- root
- task_for_pid(0)
- amfi
- sandbox
- __mac_mount
- _mapFor10

Escalate privileges

- Interesting APIs are restricted
- task_for_pid, mount etc

Escalate privileges patch

- Find setreuid
- Find ruid/euid checks
- Patch to skip reuid checks condition

Escalate privileges patch detailed

kern_prot.c

```
int setreuid(...)
{
    ...
    if (((ruid != KAUTH_UID_NONE &&        /* allow no change of ruid */
          ruid != my_pcred->cr_ruid &&     /* allow ruid = ruid */
          ruid != my_pcred->cr_uid &&      /* allow ruid = euid */
          ruid != my_pcred->cr_svuid) ||   /* allow ruid = svuid */
         (euid != KAUTH_UID_NONE &&        /* allow no change of euid */
          euid != my_pcred->cr_uid &&      /* allow euid = euid */
          euid != my_pcred->cr_ruid &&     /* allow euid = ruid */
          euid != my_pcred->cr_svuid)) &&  /* allow euid = svuid */
         (error = suser(my_cred, &p->p_acflag))) { /* allow root user any */
            kauth_cred_unref(&my_cred);
            return (error);
        }
    ...
}
```

```
int setreuid(...)
{
    ...
    if (0) {    /* allow no change of ruid */
                /* allow ruid = ruid */
                /* allow ruid = euid */
                /* allow ruid = svuid */
                /* allow no change of euid */
                /* allow euid = euid */
                /* allow euid = ruid */
                /* allow euid = svuid */
                /* allow root user any */
            kauth_cred_unref(&my_cred);
            return (error);
    }
    ...
}
```

Kernel task

- Easy access to kernel memory
- Required for some kern utilities

Kernel task patch

- Patch task_for_pid
- Re-implement task_for_pid in ROP
- Find kernel task in memory

Kernel task patch detailed

vm_unix.c

```
kern_return_t task_for_pid(...)
{
    ...
    if (pid == 0) {
        ...
        return(KERN_FAILURE);
    }

    ...









}
```

User is already ‘root’

```
if (0) {
    ...
    return(KERN_FAILURE);
}
```

Kernel task patch detailed

vm_unix.c

```
kern_return_t task_for_pid(...)
{
    ...
    if (pid == 0) {
        ...
        return(KERN_FAILURE);
    }

    ...

    if (!(task_for_pid_posix_check(p))) {
        error = KERN_FAILURE;
        goto tfpout;
    }

    ...

    if (p->task != TASK_NULL) {
        if (!kauth_cred_issuser(kauth_cred_get()) && ... ) {
            ...

            error = KERN_FAILURE;
            goto tfpout;
        }
    }
}
```

User is not ‘root’

```
if (0) {
    ...
    return(KERN_FAILURE);
}
```

```
if (0) {
    error = KERN_FAILURE;
    goto tfpout;
}
```

```
if (p->task != TASK_NULL) {
    if (0) {
        ...

        error = KERN_FAILURE;
        goto tfpout;
    }
}
```

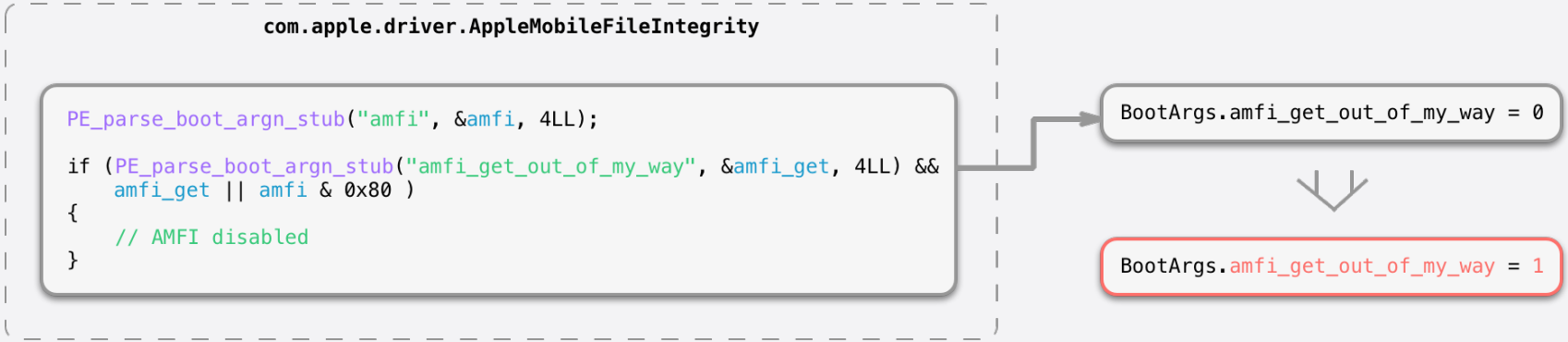
Apple Mobile File Integrity (AMFI)

- Run unsigned code
- Fake entitlements
- Get other process tasks
- Restrictions on mmap, mprotect etc

AMFI patch

- Patch amfi_get_out_of_my_way
- Patch PE_i_can_has_debugger
- Patch amfi mac policies

AMFI patch detailed



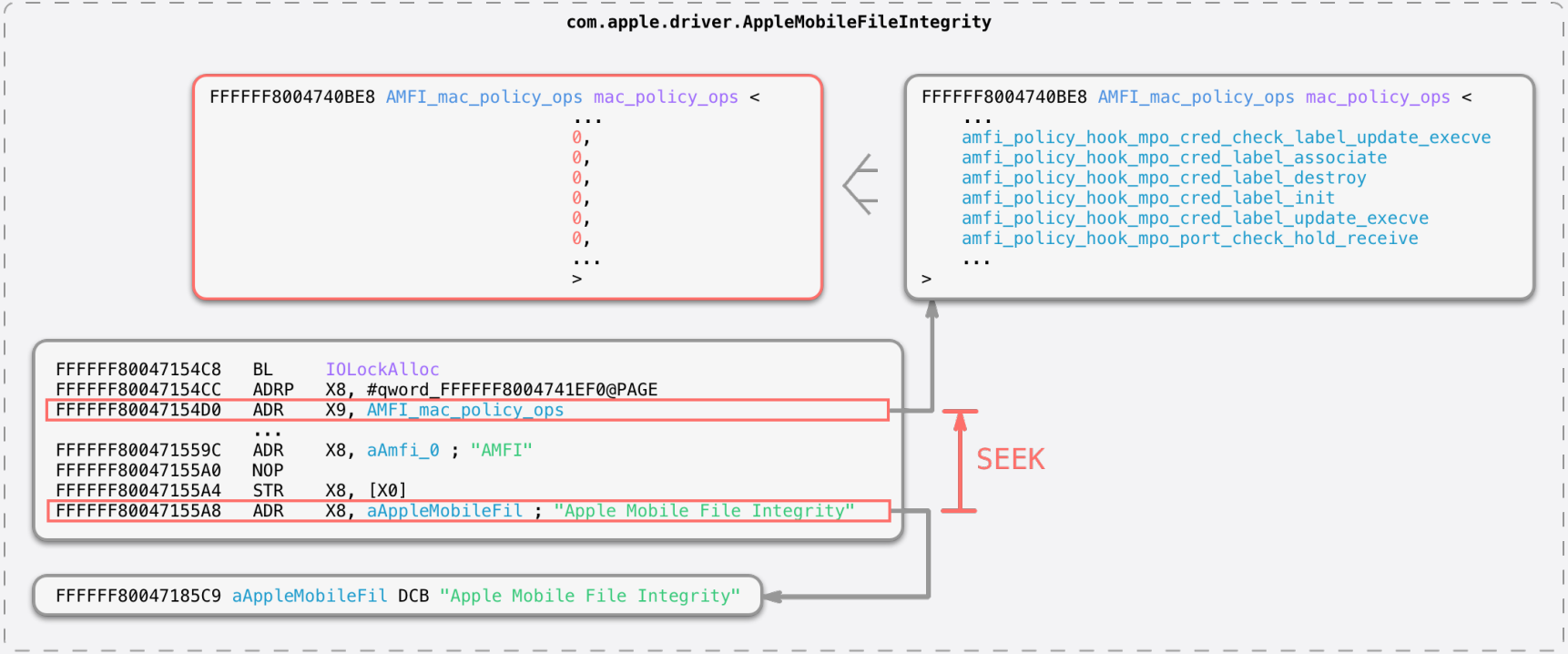
AMFI policy patch detailed

```
FFFFFF8004741660 mac_policy_conf AMFI_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                        */
};
```

```
struct mac_policy_ops {
    ...
    mpo_cred_check_label_update_execve*  mpo_cred_check_label_update_execve_t;
    mpo_cred_label_associate*            mpo_cred_label_associate_t;
    mpo_cred_label_destroy*              mpo_cred_label_destroy_t;
    mpo_cred_label_init*                 mpo_cred_label_init_t;
    mpo_cred_label_update_execve*        mpo_cred_label_update_execve_t;
    mpo_port_check_hold_receive*         mpo_port_check_hold_receive_t;
    ...
};
```

AMFI policy patch detailed



AMFI policies to patch

```
mpo_cred_check_label_update_execve
mpo_cred_label_associate
mpo_cred_label_destroy
mpo_cred_label_init
mpo_cred_label_update_execve
mpo_proc_check_inherit_ipc_ports
mpo_vnode_check_signature
mpo_file_check_library_validation
mpo_policy_initbsd
mpo_proc_check_mprotect
mpo_proc_check_map_anon
mpo_vnode_check_exec
mpo_proc_check_get_task
mpo_proc_check_get_task
mpo_proc_check_run_cs_invalid
mpo_proc_check_cpumon
mpo_file_check_mmap
```

Sandbox

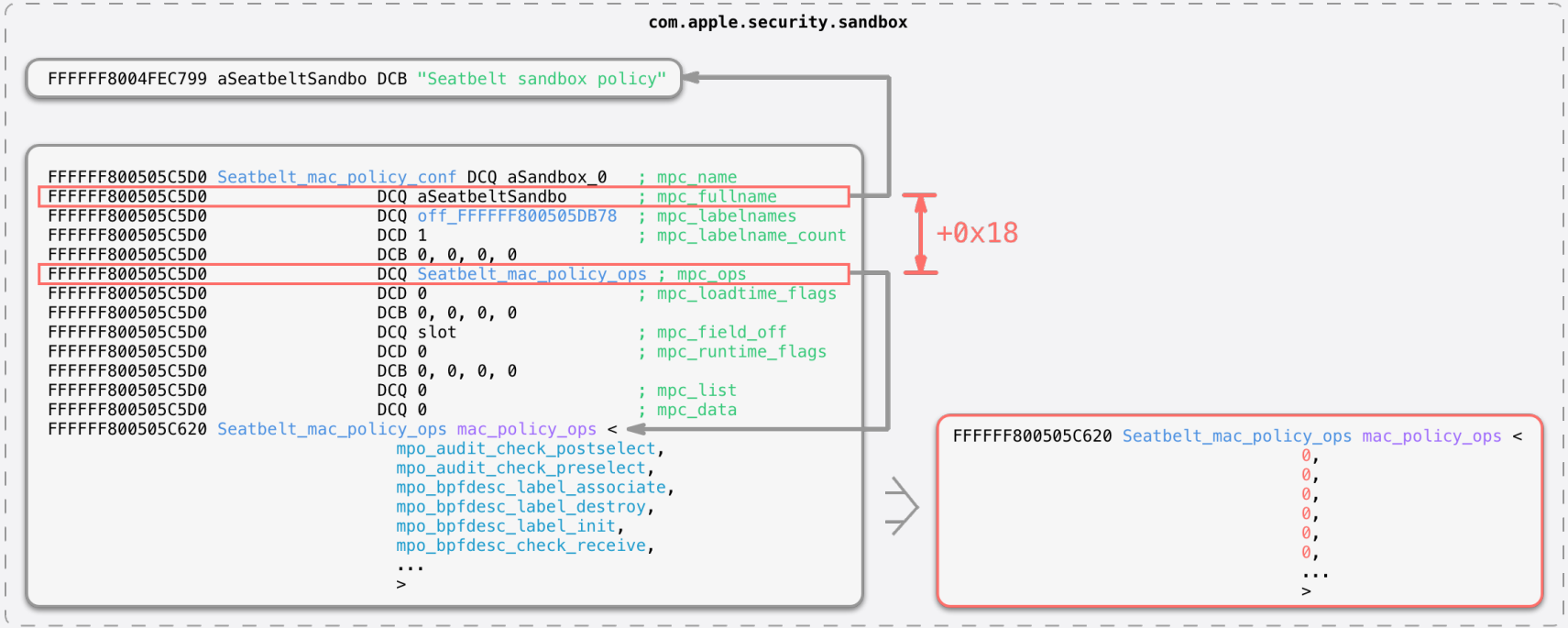
- Access files out of mobile container
- Unrestrict usage of system APIs

Sandbox patch

- Patch sb_evaluate (allow all)
- Hook sb_evaluate
- Patch sandbox mac policies

Sandbox patch detailed

Sandbox patch detailed



Sandbox policies

```
mpo_socket_check_accept
mpo_socket_check_accepted
mpo_socket_check_bind
mpo_socket_check_connect
mpo_socket_check_create
mpo_socket_check_deliver
mpo_socket_check_kqfilter
mpo_socket_check_select
mpo_socket_check_listen
mpo_socket_check_received
mpo_socket_check_setsockopt
mpo_socket_check_getsockopt
mpo_vnode_check_link
mpo_vnode_check_fsgetpath
mpo_mount_check_label_update
mpo_vnode_check_ioctl
mpo_socket_label_associate_accept
mpo_socket_label_associate
mpo_socket_check_label_update
mpo_mount_check_remount
mpo_mount_check_fsctl
mpo_mount_check_mount
mpo_vnode_check_rename
mpo_vnode_check_access
mpo_vnode_check_chroot
mpo_proc_check_get_task
mpo_vnode_check_create
mpo_vnode_check_deleteextattr
mpo_vnode_check_exchangedata
mpo_vnode_check_exec
mpo_vnode_check_getattrlist
mpo_vnode_check_getextattr
mpo_vnode_check_listextattr
mpo_vnode_check_open
mpo_vnode_check_readlink
mpo_vnode_check_revoke
mpo_vnode_check_setattrlist
mpo_vnode_check_setextattr
mpo_vnode_check_setflags
mpo_vnode_check_setmode
mpo_vnode_check_setowner
mpo_vnode_check_setutimes
mpo_vnode_check_stat
mpo_vnode_check_truncate
mpo_vnode_check_unlink
mpo_vnode_notify_create
mpo_vnode_check_uipc_bind
mpo_vnode_check_uipc_connect
```

__mac_mount

- Remount system partition
- Get write access to system partition

__mac_mount patch

- Patch __mac_mount
- Call mount_common from kernel

__mac_mount patch detailed

vfs_syscalls.c

```
int mount(...)
{
    ...
    return (__mac_mount(p, &muap, retval));
}
```

```
int __mac_mount(...)
{
    ...
#if SECURE_KERNEL
    if ((flags & MNT_RDONLY) == 0) {
        /* Release kernels are not allowed to mount "/" as rw */
        error = EPERM;
        goto out;
    }
#endif
    ...
}
```

```
#define MNT_RDONLY 0x00000001
```

```
int __mac_mount(...)
{
    ...
#if SECURE_KERNEL
    if (0) {
        /* Release kernels are not allowed to mount "/" as rw */
        error = EPERM;
        goto out;
    }
#endif
    ...
}
```

_mapForIO lock

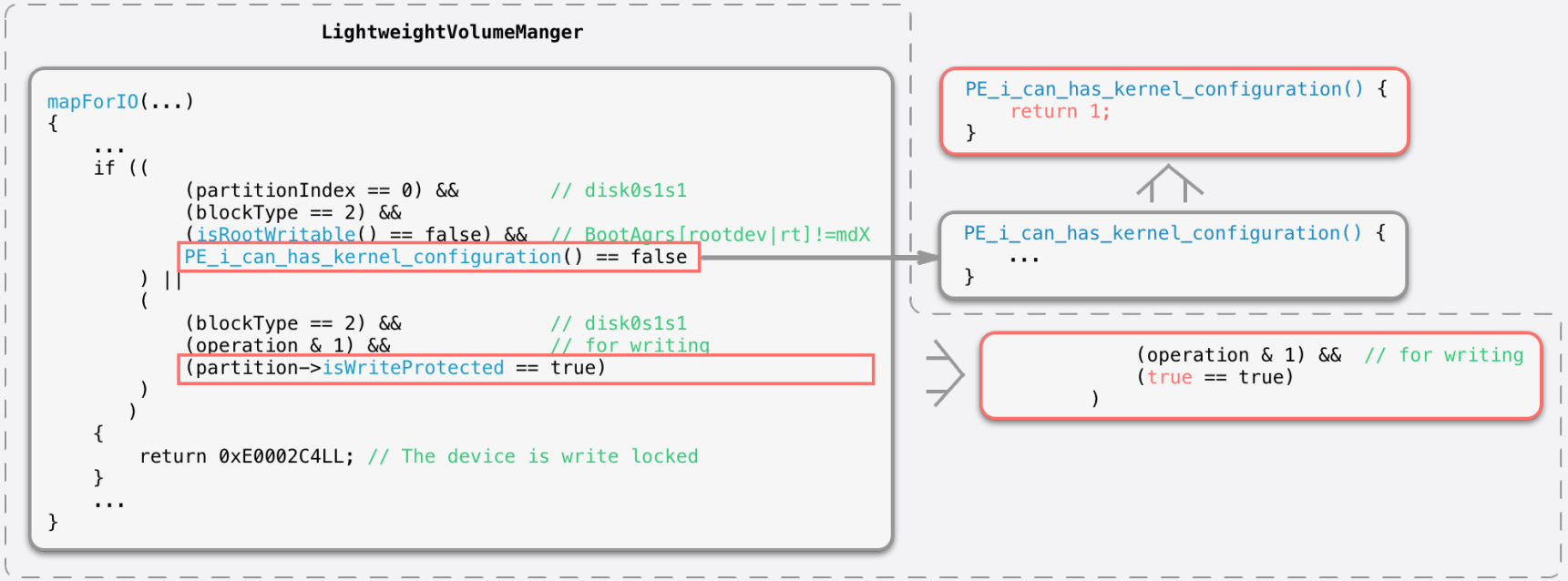
- "/" is mounted as read only
- only "/private/var" can be written

```
iPhone:~ root# df -h
Filesystem       Size   Used  Avail Capacity iused    ifree %iused  Mounted on
/dev/disk0s1s1  3.6Gi  3.0Gi  583Mi    85%  397159    74591   84%   /
devfs            57Ki   57Ki    0Bi   100%     196        0  100%   /dev
/dev/disk0s1s2   55Gi  2.3Gi   53Gi     5%  616136 13792689    4%   /private/var
/dev/disk0s1s3   10Mi  2.3Mi  7.8Mi    23%     574     1984   22%   /private/var/wireless/baseband_data
/dev/disk0s1s4  500Mi  500Mi    0Bi   100%  127987        0  100%   /private/var/logs
iPhone:~ root#
```

_mapForIO lock patch

- Patch _mapForIO
- Patch PE_i_can_has_kernel_configuartion

_mapForIO lock patch detailed

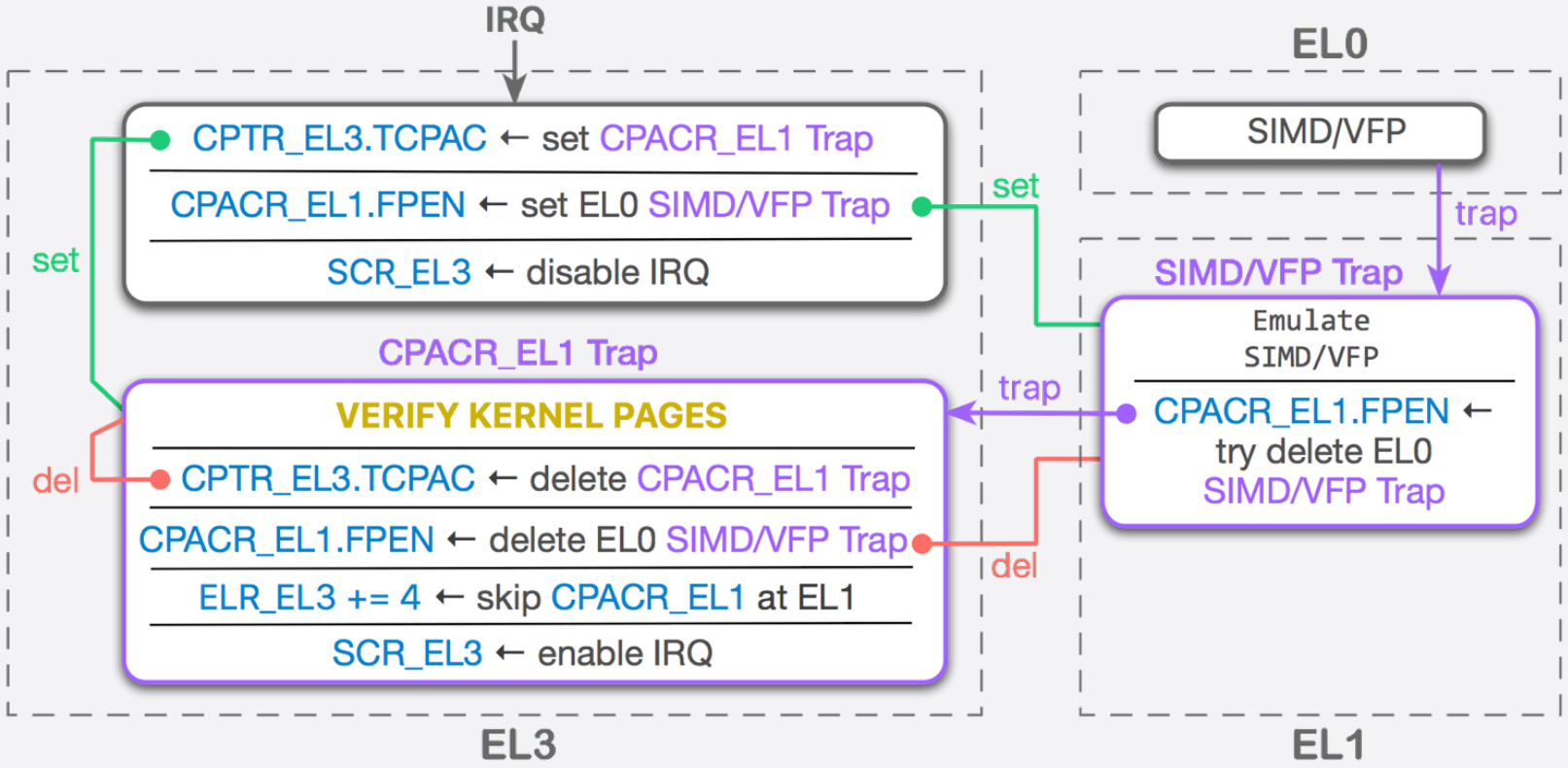


Kernel Patch Protection

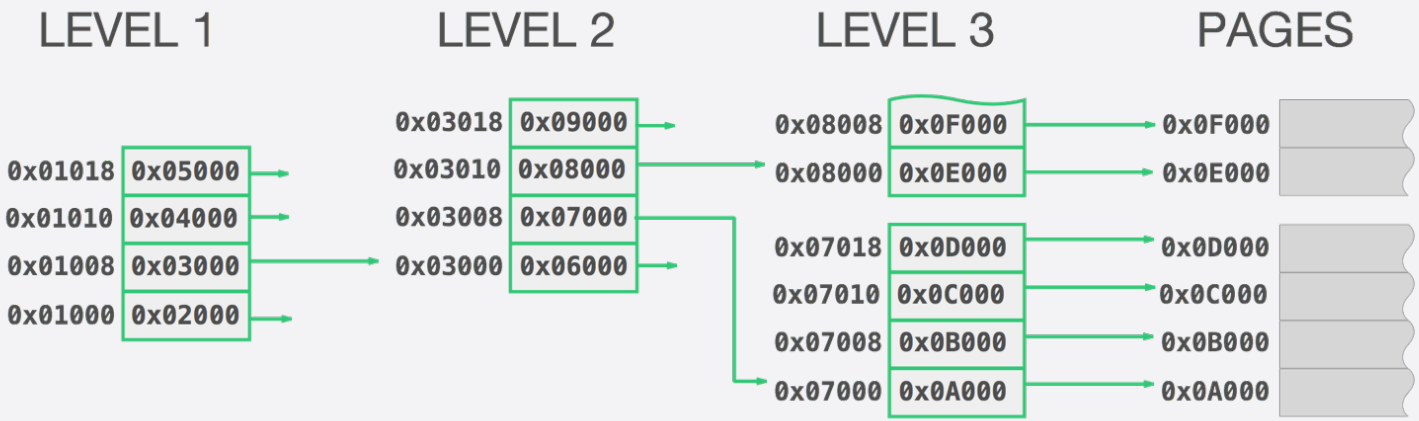
Bypassing KPP strategies

- Checks for kernel pages, MMU, sysregs
- Execution on EL3
- Can't disable, can race or …

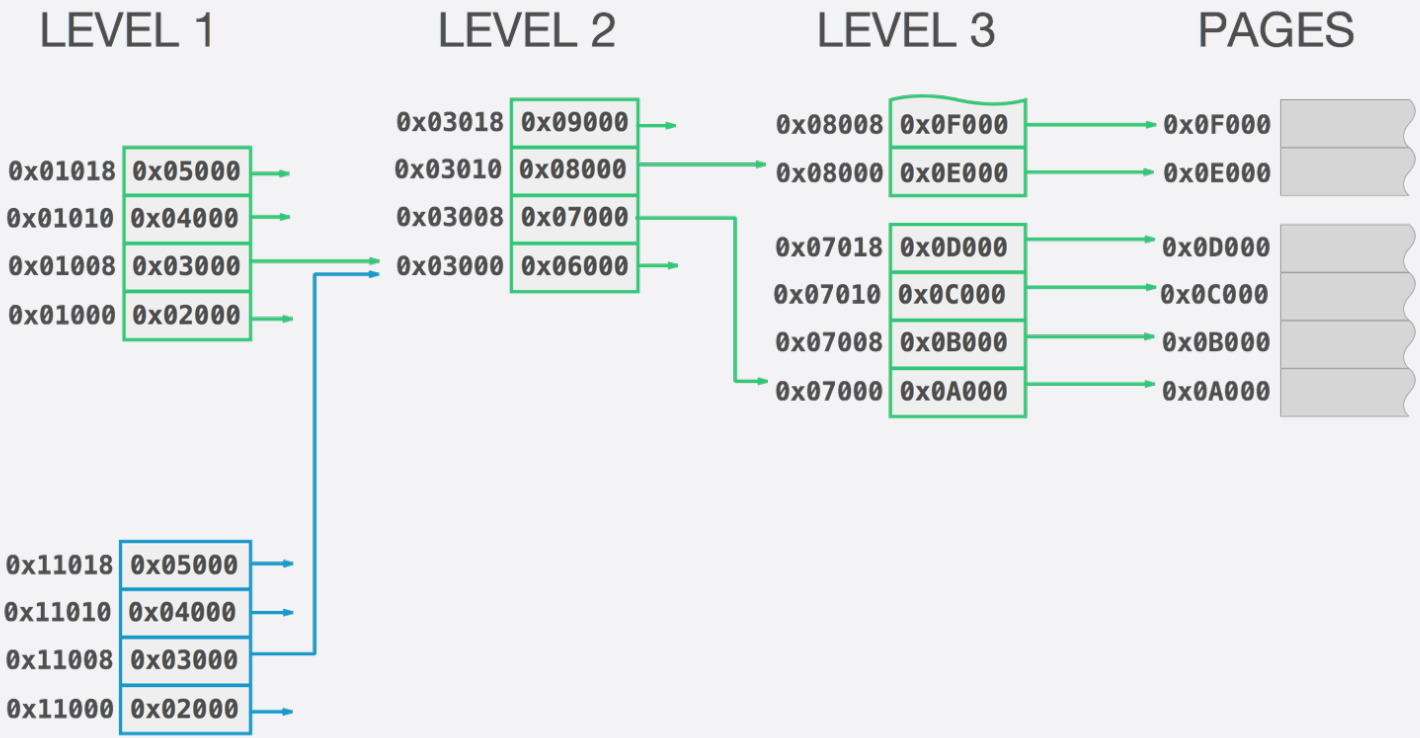
How KPP works?



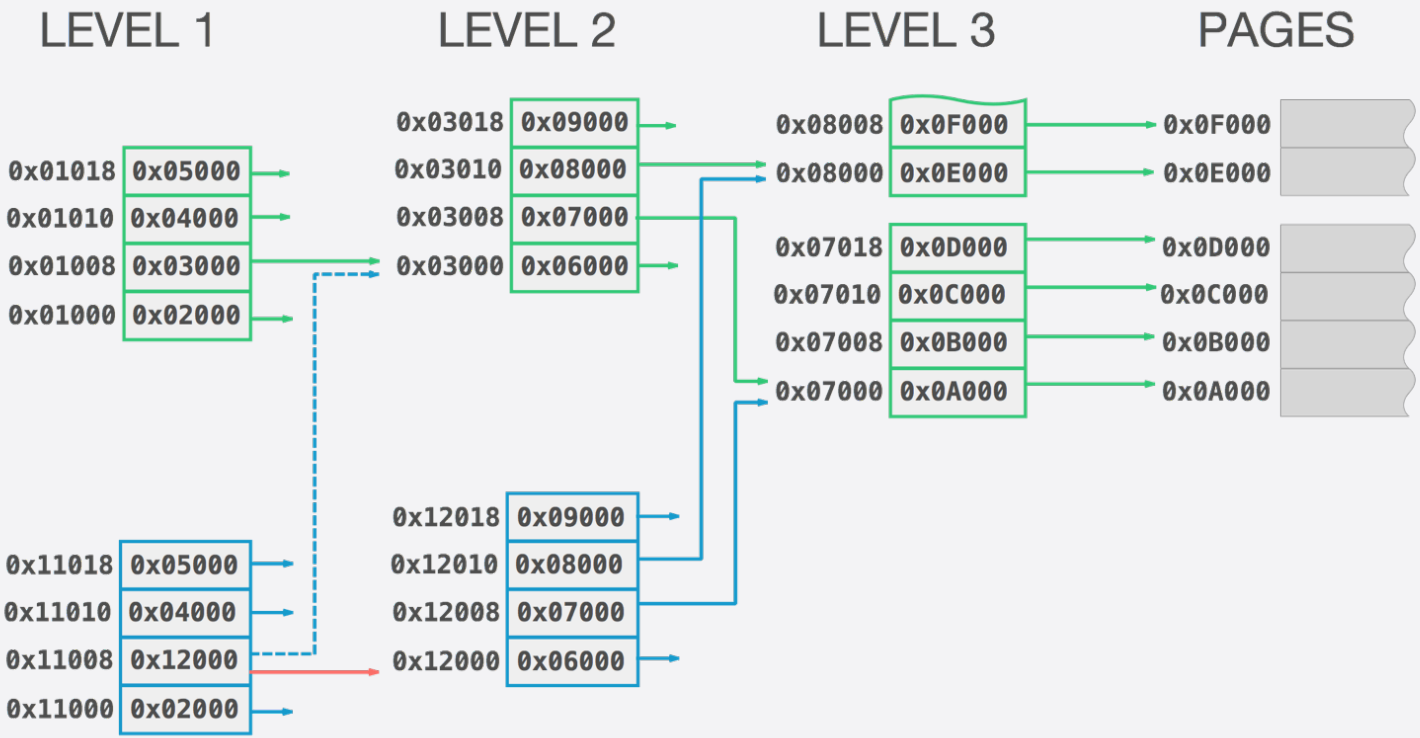
Original translation table



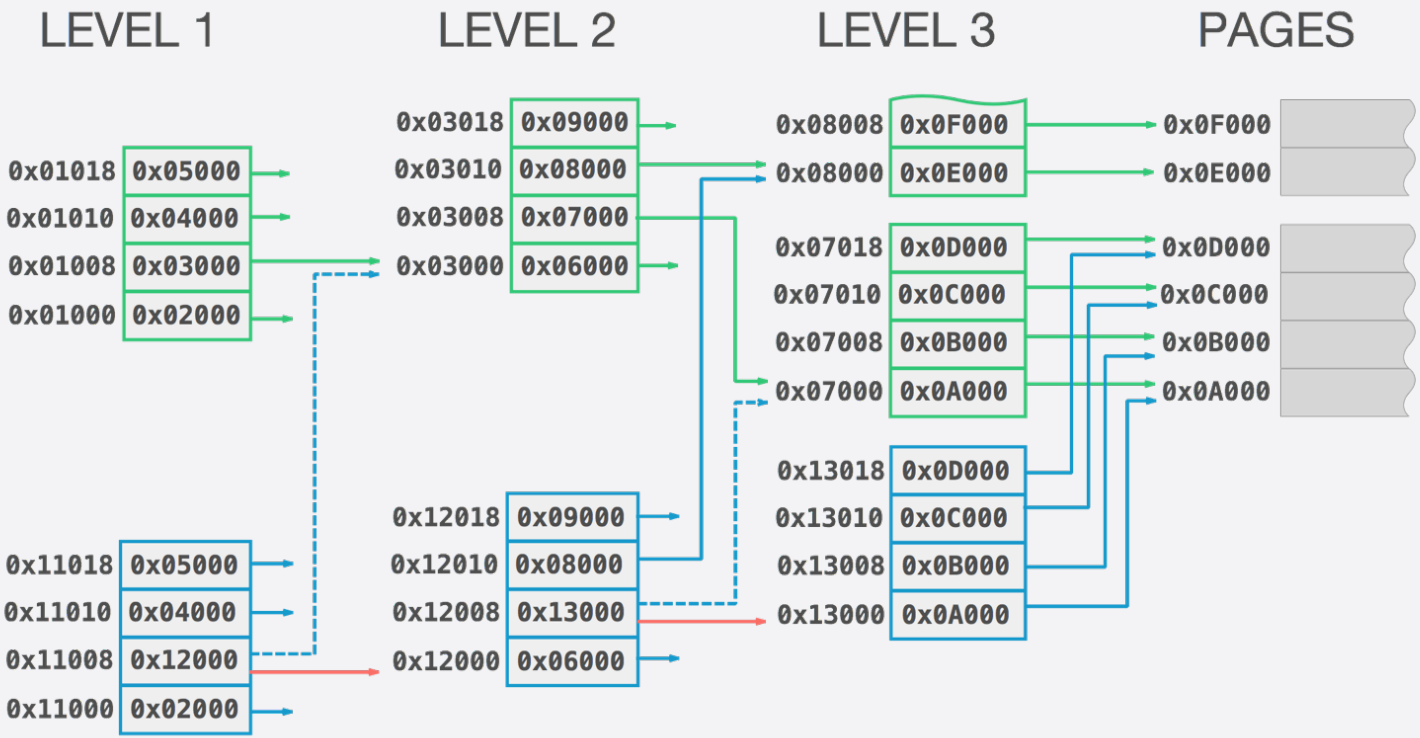
Create fake Level 1 table



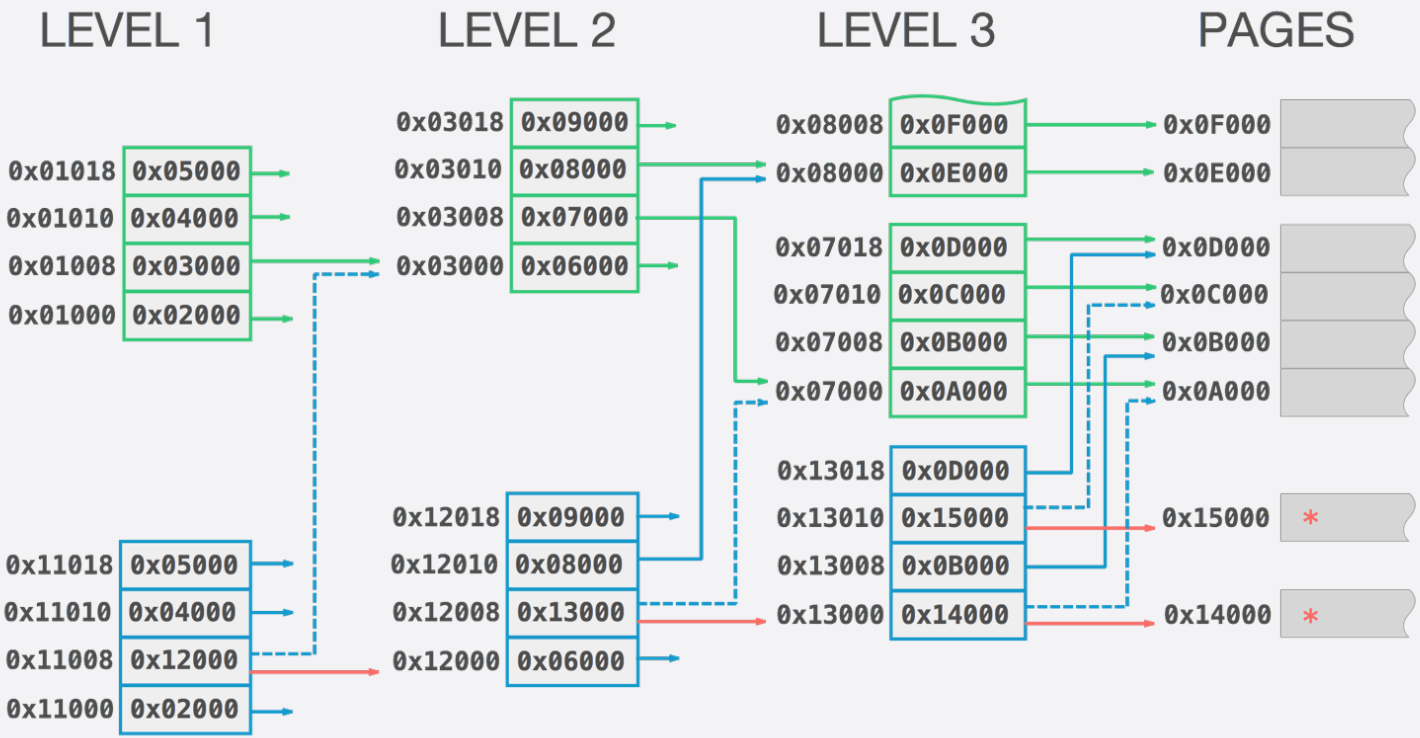
Create fake Level 2 table



Create fake Level 3 table



Create fake pages



BBQit Framework

```
// *** WALKER
TTWalker<TTGranule::Granule4K, JBPrimitives> walker(TTBR1_EL1, TTLevel::Level1);

// Get physical address
phys_addr_t paddr = walker.findPhysicalAddress(vaddr);

// Perform translation table walk
WalkResult walkResult = walker->walkTo(vaddr, [] (WalkPosition* position, TTGenericEntry* entry) -> WalkOperation {
    printf(" Level%d: %c%c%c\n", position->level,
           (entry->isValid())? 'v' : '-',
           (entry->isTableDescriptor())? 't' : '-',
           (entry->isPageDescriptor())? 'p' : '-');
    return WalkOperation::Continue;
});

// *** RELOCATOR
PageRelocator<TTGranule::Granule4K, JBPrimitives> relocator(TTBR1_EL1, TTLevel::Level1);

// Relocate page (single step)
relocator.relocatePageFor(vaddr, [&relocator] (TTLevel level, TTGenericEntry* oldEntry, TTGenericEntry* newEntry) -> ttentry_t {
    printf("    MOVE: 0x%.16llX -> 0x%.16llX\n",
           oldEntry->getOutputAddress(),
           newEntry->getOutputAddress());
    return newEntry->getDescriptor();
});

// Relocate page (two steps - prepare/complete)
virt_addr_t newPageVA = relocator.preparePageRelocationFor(vaddr);

// Modify new page
relocator.writeAddress(newPageVA, 0xDEADBEEFDEADBEEF);

// Finish relocation
relocator.completeRelocation();
```

KPP bypass technique

KERNEL

```
...
MOV     X0, #0x300000
MSR     CPACR_EL1, X0   ; Trigger KPP checks
...
```

```
...
MOV     X0, #0x300000
BL      cpacr_trampoline ; Call trampoline
...
```

```
NOP
NOP
NOP
```

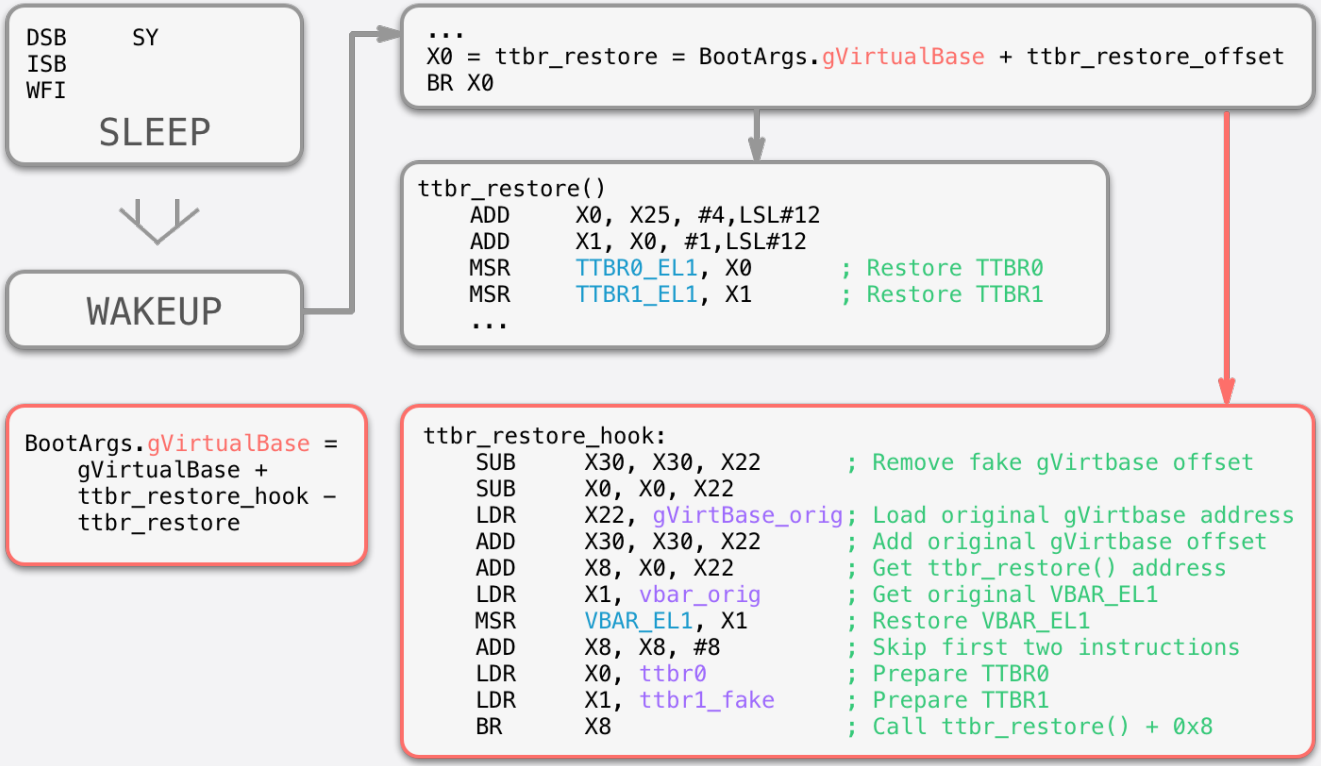
```
cpacr_trampoline:
    LDR   X1, $PC + 8   ; Get hook address
    BR    X1            ; Call hook function
    .quad cpacr_hook
```

SHELLCODE

```
cpacr_hook:
    MRS     X1, TTBR1_EL1   ; Get fake TTBR1_EL1
    LDR     X0, ttbr1_orig  ; Get original TTBR1_EL1
    MSR     TTBR1_EL1, X0   ; Restore original TTBR1_EL1
    MOV     X0, #0x300000
    MSR     CPACR_EL1, X0   ; Trigger KPP checks
    MSR     TTBR1_EL1, X1   ; Revert fake TTBR1_EL1 back
    TLBI    VMALLE1         ; Invalidate all stage 1 translations
    ISB
    DSB     SY
    DSB     ISH
    ISB
    RET
```

WELL DONE!?

KPP bypass technique (continue)



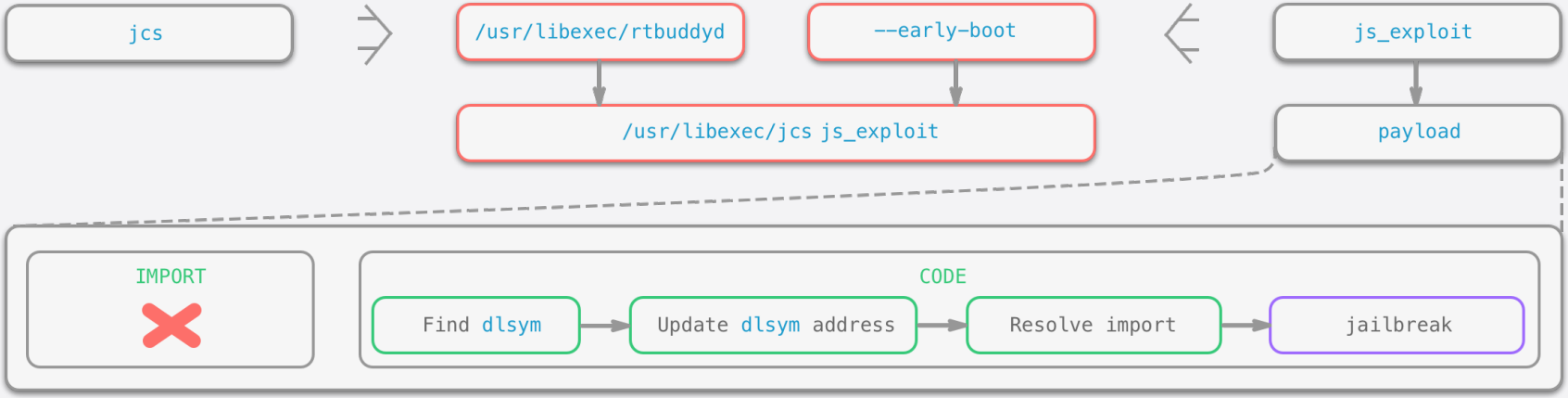
Achieving persistence strategies

- Find service that spawns on boot
- Check if it is running as root (optional)
- Find userland codesign bug
- Symlink system service to exec cs bypass

Achieving persistence example

- JavaScriptCore jsc interpreter
- Signed by Apple
- Can execute code on RWX segment
- Copy as system service to spawn on boot

Achieving persistence details



SSH

- Copy dropbear or install Cydia
- tcprelay.py -t 22:4222
- Password ‘alpine’

Cydia

- Copy tar to /bin/tar
- tar -xvfp cydia.tar
- Optional /.cydia_no_stash
- Flush uicache using /usr/bin/uicache

iOS 10 security enhancements

- New heap layout
- AMFI and Sandbox hardening
- KPP enhancements

iOS 10 amfi mitigations

- MISValidateSignatureAndCopyInfo
  - Replace with CFEqual or similar will not work
- validateCodeDirectoryHashInDaemon
  - possible race condition fixed
- Policy patches still work

iOS 10 sandbox mitigations

- New operations

  boot-arg-set, fs-snapshot*, system-package-check, …

- New hooks

  _hook_iokit_check_nvram_get,
  _hook_proc_check_set_host_special_port,
  _hook_proc_check_get_cs_info …

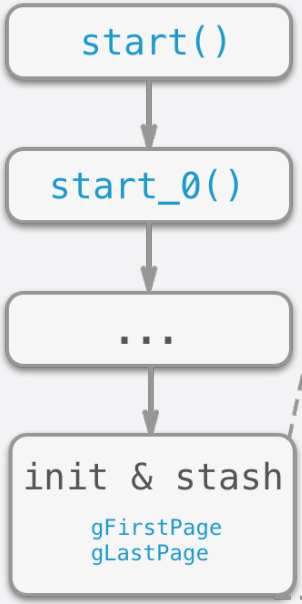
iOS 10 KPP enhancements

- New kernelcache layout
- Now _got segments are protected
- New hardware migrations on iPhone 7/Plus

KPP hardware mitigations

- AMCC
- Watch memory region for any access
- Prevents writing inside region
- Prevents exec outside region

KPP hardware mitigations




```
uint64_t dummy;
uint64_t enable = 1;
uint64_t kernelBase = 0xFFFFFFF007004000; // includes slide in runtime
uint64_t pageMask = 0x0 - (1 << PAGE_SHIFT_CONST);

gMemProt_State = enable;

// Get address of the first kernel section to protect
uint64_t firstPageVA = GetSectionVA(kernelBase, "__PRELINK_TEXT", &dummy);
uint64_t firstPagePA = VirtToPhys(firstPageVA);

// Get address of the kernel section after the last one to protect
uint64_t lastPageVA = GetSectionVA(kernelBase, "__LAST", &dummy);
uint64_t lastPagePA = VirtToPhys(lastPageVA);

lastPagePA -= 1;        // Get last byte of the last section to protect
lastPagePA &= pageMask;// Align to the page boundary

RegisterWrite(S3_4_C15_C2_3, firstPagePA); // Set protected region first page address
RegisterWrite(S3_4_C15_C2_4, lastPagePA);  // Set protected region last page address
RegisterWrite(S3_4_C15_C2_2, enable);      // Set 1 to enable/fetch settings to hardware
```


```
RegisterWrite(S3_4_C15_C2_3, gFirstPage);
RegisterWrite(S3_4_C15_C2_4, gLastPage);
RegisterWrite(S3_4_C15_C2_2, 1);
```

Future of jailbreaks

- iOS is more secure on each release
- More security on hardware side
- Exploits will be more valuable
- But there will be bugs and write-ups

Black Hat Sound Bytes

- Jailbreak is doable with public bug info
- Patches and KPP bypass from this talk
- May the XNU source be with you

@FriedAppleTeam

@mbazaliy @getorix @in7egral