Challenge Impossible

-- Multiple Exploit On Android

Hanxiang Wen, Xiaodong Wang

C0RE Team

About us & C0RE Team

- Hanxiang Wen, 温瀚翔
  - Security researcher @ C0RE Team
  - Focus on Android vulnerability research and exploit development

- Xiaodong Wang, 王晓东
  - Security researcher @ C0RE Team
  - Focus on Kernel vulnerability research and exploit development

- C0RE Team
  - A security-focused group started in mid-2015, with a recent focus on the Android/Linux platform
  - The team aims to discover zero-day vulnerabilities, develop proof-of-concept and exploit
  - 131 public CVEs for AOSP and Linux Kernel currently
  - Android top researcher team for submitting high quality reports to Google VRP.

Agenda

- AOSP Exploit
  - CVE-2016-6707
  - Looking Into Exploit
  - Improvement & Limitation
- Kernel Exploit
  - CVE-2017-0437
  - Vulnerability Analysis
  - How to Exploit
- Combination

CVE-2016-6707

Background:

- "BitUnmap" in system_server
- Open source exploit with some defects

Thanks to Gal Beniamini, blog link:
https://googleprojectzero.blogspot.com/2016/12/bitunmap-attacking-android-ashmem.html

Mismatch in Ashmem

Set/Get size in Ashmem:

```
static long ashmem_ioctl(struct file *file, unsigned int cmd, unsigned long arg)
{
    struct ashmem_area *asma = file->private_data;
    long ret = -ENOTTY;

    switch (cmd) {
--------------------------------[skipped]----------------------------------------
    case ASHMEM_SET_SIZE:
        ret = -EINVAL;
        if (!asma->file) {
            ret = 0;
            asma->size = (size_t) arg;
        }
        break;
    case ASHMEM_GET_SIZE:
        ret = asma->size;
        break;
--------------------------------[skipped]----------------------------------------
    }
    return ret;
}
```

Mismatch in Ashmem

Map memory with Ashmem:

```
static int ashmem_mmap(struct file *file, struct vm_area_struct *vma)
{
    struct ashmem_area *asma = file->private_data;
    int ret = 0;
--------------------------------[skipped]----------------------------------------
    if (!asma->file) {
        char *name = ASHMEM_NAME_DEF;
        struct file *vmfile;

        if (asma->name[ASHMEM_NAME_PREFIX_LEN] != '\0')
            name = asma->name;

        /* ... and allocate the backing shmem file */
        vmfile = shmem_file_setup(name, asma->size, vma->vm_flags);
        if (unlikely(IS_ERR(vmfile))) {
            ret = PTR_ERR(vmfile);
            goto out;
        }
        asma->file = vmfile;
    }
--------------------------------[skipped]----------------------------------------
    return ret;
}
```

Region size in ashmem may not equal to its mmaped size !!!

False assumption in Bitmap

Create Bitmap

```
status_t Parcel::readBlob(size_t len, ReadableBlob* outBlob) const {
--------------------------------[skipped]--------------------------------
    void* ptr = ::mmap(NULL, len, isMutable ? PROT_READ | PROT_WRITE : PROT_READ,
            MAP_SHARED, fd, 0);
--------------------------------[skipped]--------------------------------
    return NO_ERROR;
}
```

```
static jobject Bitmap_createFromParcel(JNIEnv* env, jobject, jobject par
--------------------------------[skipped]--------------------------------
    std::unique_ptr<SkBitmap> bitmap(new SkBitmap);

    if (!bitmap->setInfo(SkImageInfo::Make(width, height, colorType, alphaType), rowBytes)) {
        return NULL;
    }
--------------------------------[skipped]--------------------------------
    size_t size = bitmap->getSize();
    android::Parcel::ReadableBlob blob;
    android::status_t status = p->readBlob(size, &blob);
--------------------------------[skipped]--------------------------------
    Bitmap* nativeBitmap;
--------------------------------[skipped]--------------------------------
    int dupFd = dup(blob.fd());
--------------------------------[skipped]--------------------------------
    nativeBitmap = GraphicsJNI::mapAshmemPixelRef(env, bitmap.get(),
                ctable, dupFd, const_cast<void*>(blob.data()), !isMutable);
--------------------------------[skipped]--------------------------------
    return GraphicsJNI::createBitmap(env, nativeBitmap,
            getPremulBitmapCreateFlags(isMutable), NULL, NULL, density);
}
```

```
android::Bitmap* GraphicsJNI::mapAshmemPixelRef(JNIEnv* env, SkBitmap* bitmap,
        SkColorTable* ctable, int fd, void* addr, bool readOnly) {
    const SkImageInfo& info = bitmap->info();
--------------------------------[skipped]--------------------------------
    android::Bitmap* wrapper = new android::Bitmap(addr, fd, info, rowBytes, ctable);
--------------------------------[skipped]--------------------------------
    bitmap->lockPixels();
    return wrapper;
}
```

```
Bitmap::Bitmap(void* address, int fd,
            const SkImageInfo& info, size_t rowBytes, SkColorTable* ctable)
        : mPixelStorageType(PixelStorageType::Ashmem) {
    mPixelStorage.ashmem.address = address;
    mPixelStorage.ashmem.fd = fd;
    mPixelStorage.ashmem.size = ashmem_get_size_region(fd);
--------------------------------[skipped]--------------------------------
}
```

False assumption in Bitmap

Free Bitmap

```
Bitmap::~Bitmap() {
    doFreePixels();
}

void Bitmap::doFreePixels() {
    switch (mPixelStorageType) {
--------------------------------[skipped]-------------------------------
    case PixelStorageType::Ashmem:
        munmap(mPixelStorage.ashmem.address, mPixelStorage.ashmem.size);
        close(mPixelStorage.ashmem.fd);
        break;
--------------------------------[skipped]-------------------------------
}
```

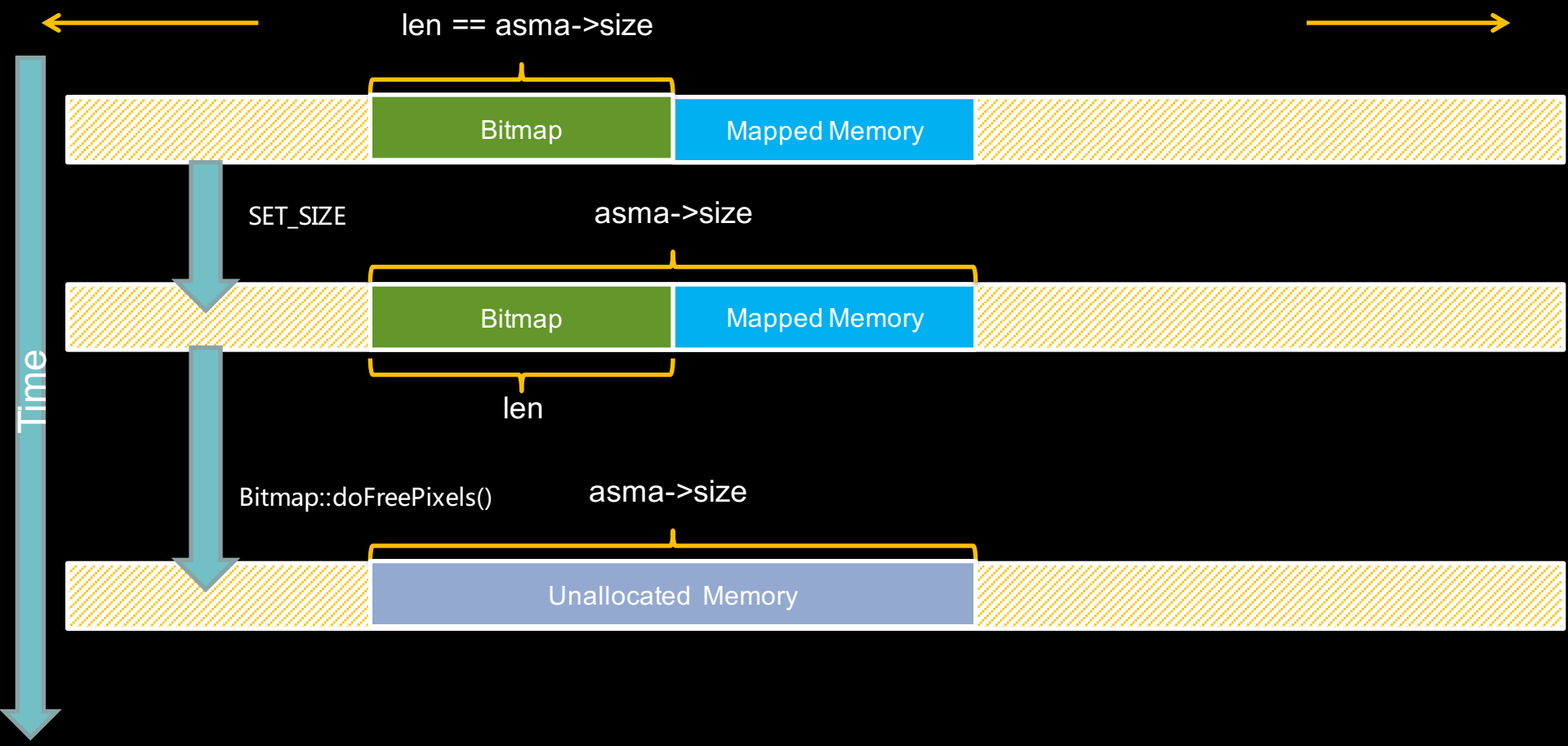
```
Bitmap::Bitmap(void* address, int fd,
            const SkImageInfo& info, size_t rowBytes, SkColorTable* ctable)
        : mPixelStorageType(PixelStorageType::Ashmem) {
    mPixelStorage.ashmem.address = address;
    mPixelStorage.ashmem.fd = fd;
    mPixelStorage.ashmem.size = ashmem_get_size_region(fd);
--------------------------------[skipped]-------------------------------
}
```

size (using in mmap)

$\neq$

mPixelStorage.ashmem.size(using in munmap)

Bitmap OOB unmap



Preparation

Target structure --- Thread

```
static int __allocate_thread(pthread_attr_t* attr, pthread_internal_t** threadp, void** child_stack) {
  size_t mmap_size;
  uint8_t* stack_top;
--------------------------------[skipped]-----------------------------------------
    mmap_size = BIONIC_ALIGN(attr->stack_size + sizeof(pthread_internal_t), PAGE_SIZE);
    attr->guard_size = BIONIC_ALIGN(attr->guard_size, PAGE_SIZE);
    attr->stack_base = __create_thread_mapped_space(mmap_size, attr->guard_size);
--------------------------------[skipped]-----------------------------------------
    stack_top = reinterpret_cast<uint8_t*>(attr->stack_base) + mmap_size;
--------------------------------[skipped]-----------------------------------------
  pthread_internal_t* thread = reinterpret_cast<pthread_internal_t*>(stack_top);
--------------------------------[skipped]-----------------------------------------
}
```

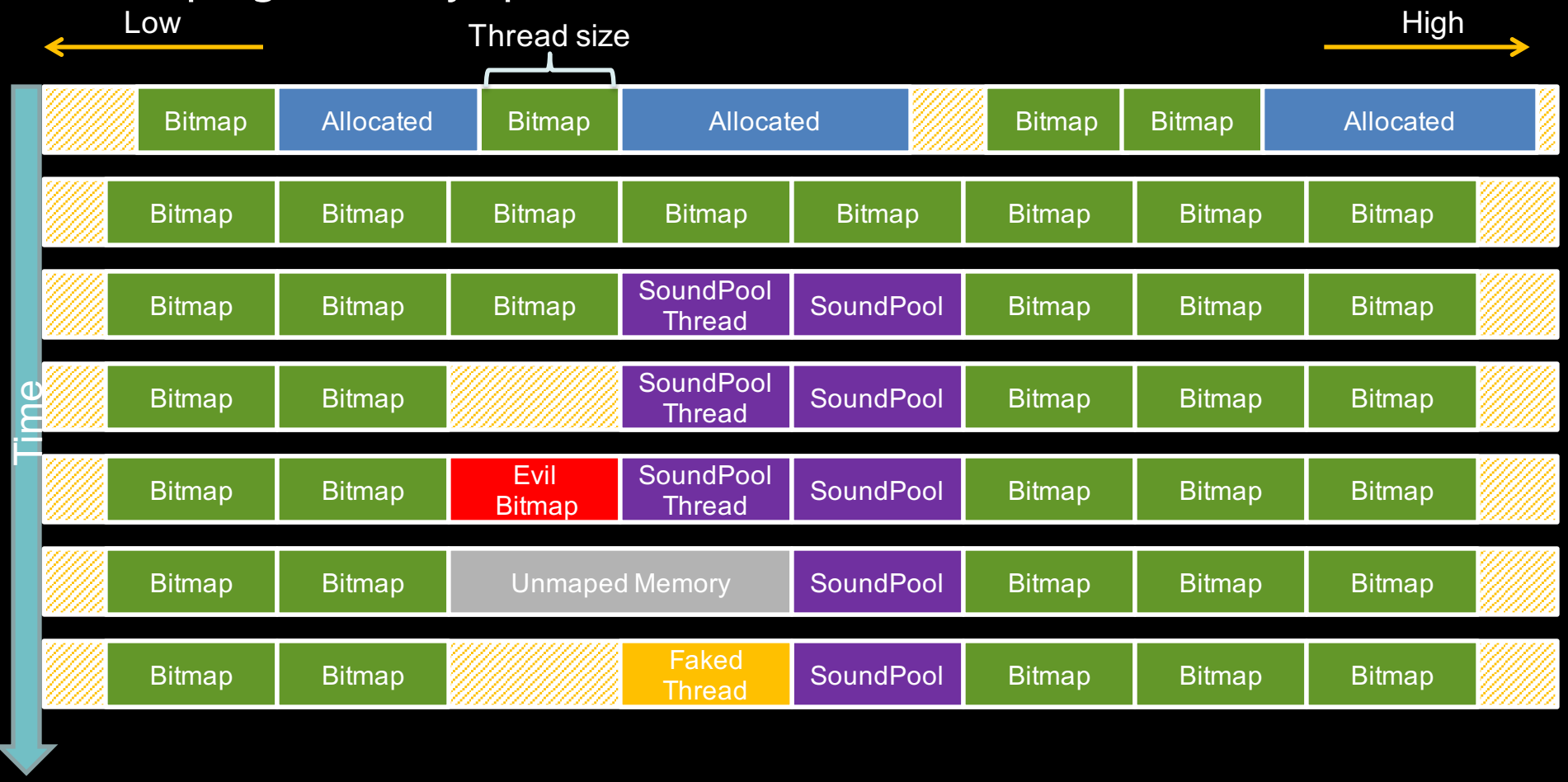
Bypass SELinux rules (Embedded Shellcode in APK):

```
// system_server.te, updated in Android-N

# system_server should never execute anything from /data except for /data/dalvik-cache files.
neverallow system_server {
   data_file_type
   -dalvikcache_data_file #mapping with PROT_EXEC
}:file no_x_file_perms;
```

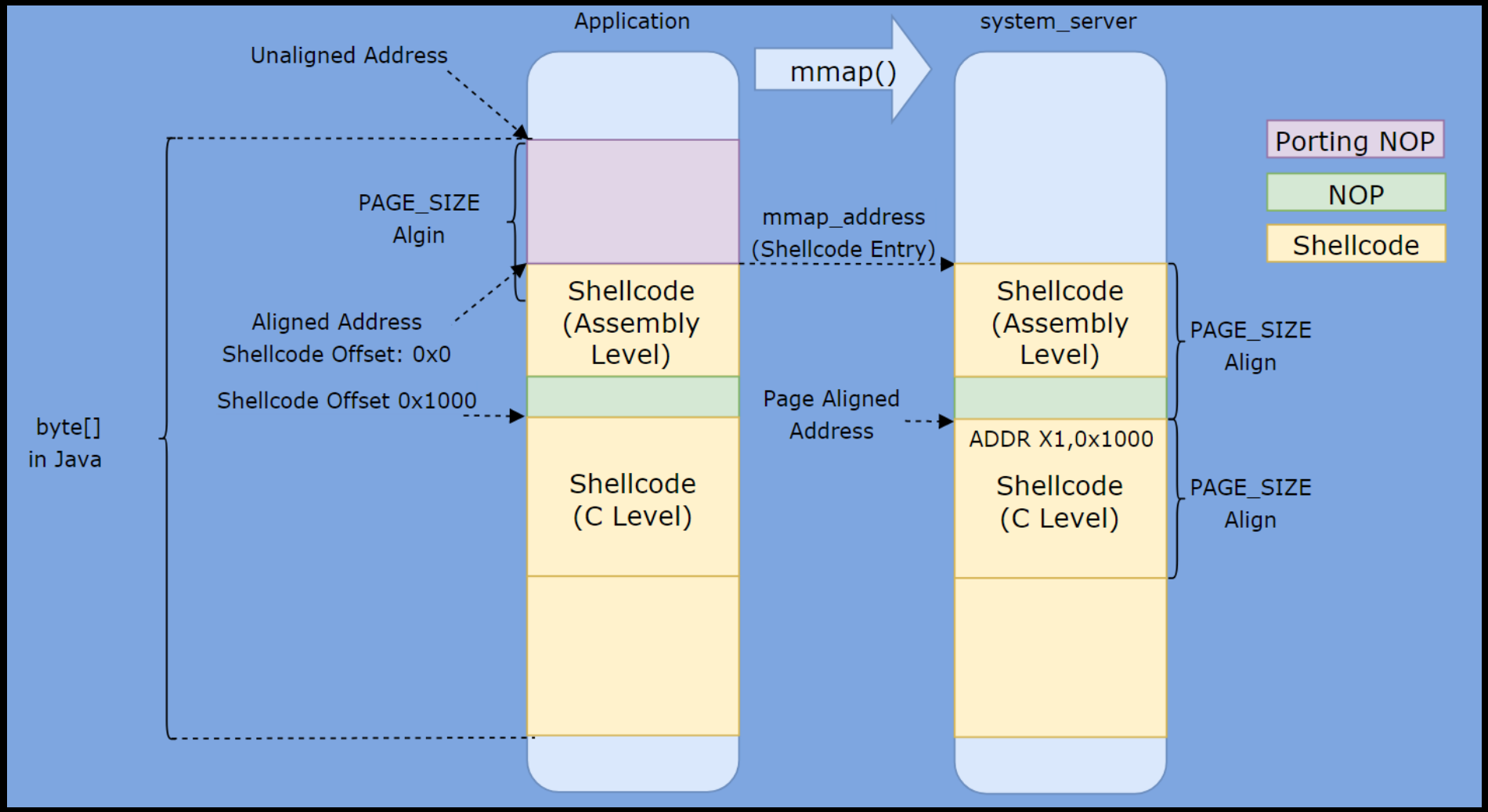
Exploit

Shaping memory space (IPC with Notification Service):



Improvement

Accuray "NOP" before "ADRP" in embedded shellcode



Patch for fake thread attributes assembly

- Enable dlopen() dlsym().

Limitation

SELinux mitigation

```
// system_server.te, updated in Android-N

# Do not allow opening files from external storage as unsafe ejection
# could cause the kernel to kill the system_server.
neverallow system_server sdcard_type:dir { open read write };
neverallow system_server sdcard_type:file rw_file_perms;

# system server should never be opening zygote spawned app data
# files directly. Rather, they should always be passed via a
# file descriptor.
# Types extracted from seapp_contexts type= fields, excluding
# those types that system_server needs to open directly.
neverallow system_server { bluetooth_data_file nfc_data_file shell_data_file app_data_file }:file open;
```

CVE-2017-0437 Introduction

- Qualcomm driver vulnerability of wlan_hdd_cfg80211.c
- Chen Hao of Qihoo 360 Technology Co. Ltd. reported to Google in February this year
- Impacted Phones: Nexus 5X/Pixel…
- Vulnerability Type: stack buffer overflow
- Exploitation： Using the stack overflow, we could rewrite the return address of the call function, then we could control the PC register to the gadget, and then remove the process’s address_limit
- We have been validated on the MTC19V version of the Nexus 5X

CVE-2017-0437Analysis

- Qualcomm Wi-Fi driver’s vulnerability

```
1462 static int
1463 __wlan_hdd_cfg80211_set_ext_roam_params(struct wiphy *wiphy,
1464                                         struct wireless_dev *wdev,
1465                                         const void *data,
1466                                         int data_len)
1467 {
[...]
1472         struct roam_ext_params roam_params;
1473         uint32_t cmd_type, req_id;
1474         struct nlattr *curr_attr;
1475         struct nlattr *tb[QCA_WLAN_VENDOR_ATTR_ROAMING_PARAM_MAX + 1];
1476         struct nlattr *tb2[QCA_WLAN_VENDOR_ATTR_ROAMING_PARAM_MAX + 1];
[...]
1509         switch(cmd_type) {
[...]
1645         case QCA_WLAN_VENDOR_ATTR_ROAM_SUBCMD_SET_BSSID_PREFS:
[...]
1651                 roam_params.num_bssid_favored = nla_get_u32(
1652                         tb[QCA_WLAN_VENDOR_ATTR_ROAMING_PA
[...]
1656                 i = 0;
1657                 nla_for_each_nested(curr_attr,
1658                         tb[QCA_WLAN_VENDOR_ATTR_ROAMING_PARAM_SET_BSSID_PREFS],
1659                         rem) {
[...]
1672                         nla_memcpy(roam_params.bssid_favored[i],
1673                                 tb2[QCA_WLAN_VENDOR_ATTR_ROAMING_PARAM_SET_LAZY_ROAM_BSSID],
1674                                 sizeof(tSirMacAddr));
[...]
1687                         i++;
1688                 }
[...]
1691                 break;
```

1. The times of loop could be set by the PoC/PWN

2. The contents of memcpy could also be set by the PoC/PWN , and then we could control the contents of the stack

3. Stack Overflow

CVE-2017-0437Analysis

- The definition of the struct roam_ext_params

```
#define MAX_SSID_ALLOWED_LIST 4
#define MAX_BSSID_AVOID_LIST  16
#define MAX_BSSID_FAVORED     16
struct roam_ext_params {
        uint8_t num_bssid_avoid_list;
        uint8_t num_ssid_allowed_list;
        uint8_t num_bssid_favored;
        tSirMacSSid ssid_allowed_list[MAX_SSID_ALLOWED_LIST];
        tSirMacAddr bssid_avoid_list[MAX_BSSID_AVOID_LIST];
        tSirMacAddr bssid_favored[MAX_BSSID_FAVORED];
        uint8_t bssid_favored_factor[MAX_BSSID_FAVORED];
};
```

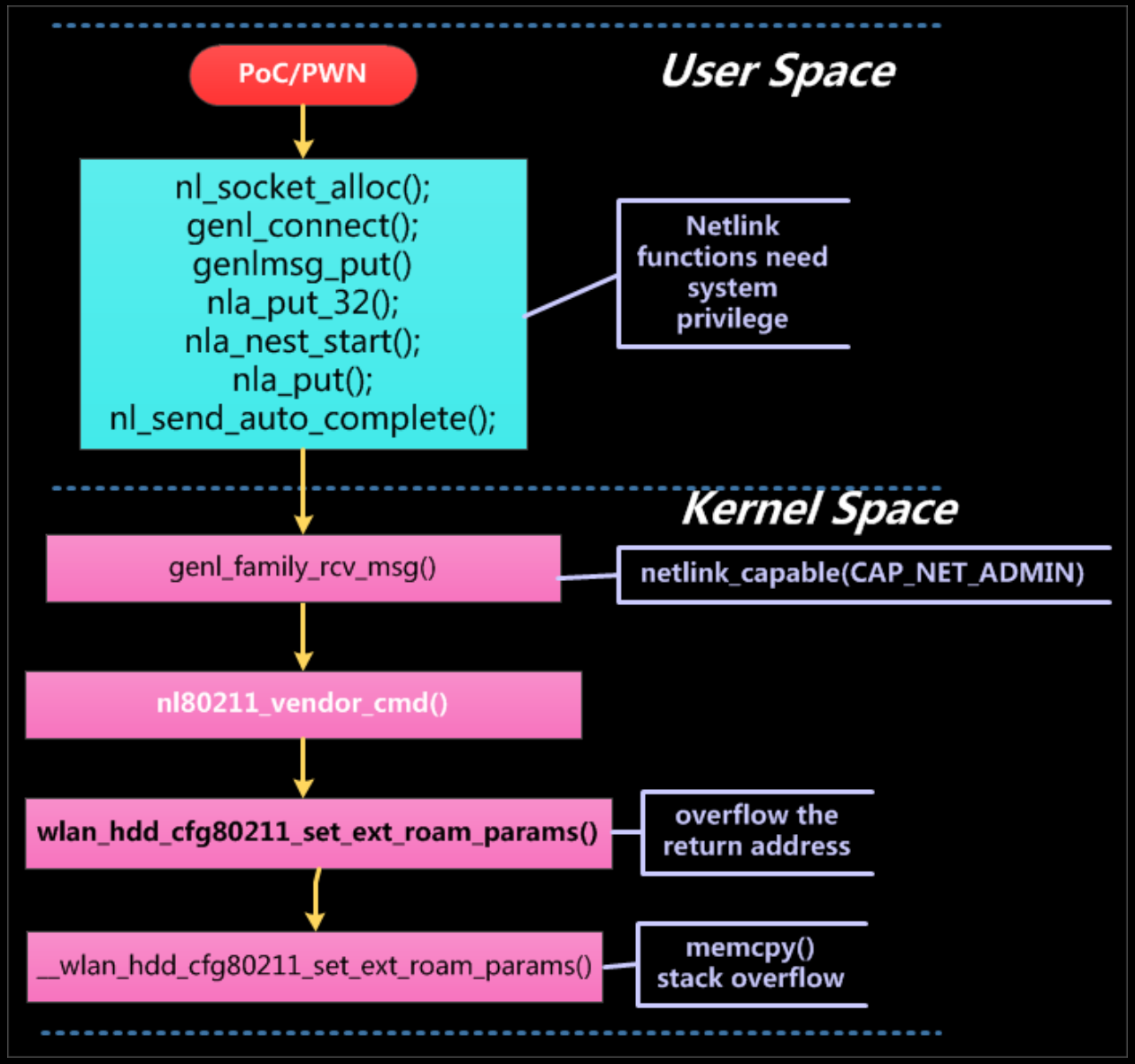
- ARMv8’s LR(X30) on the stack, Push down stack pointer and store FP and LR

```
< __wlan_hdd_cfg80211_set_ext_roam_params.isra.24 >
ffffffc000976a80:       d111c3ff        sub     sp, sp, #0x470
ffffffc000976a84:       a9ba7bfd        stp     x29, x30, [sp,#-96]!
ffffffc000976a88:       910003fd        mov     x29, sp
ffffffc000976a8c:       a90153f3        stp     x19, x20, [sp,#16]
ffffffc000976a90:       a9025bf5        stp     x21, x22, [sp,#32]
ffffffc000976a94:       a90363f7        stp     x23, x24, [sp,#48]
ffffffc000976a98:       a9046bf9        stp     x25, x26, [sp,#64]
ffffffc000976a9c:       a90573fb        stp     x27, x28, [sp,#80]
ffffffc000976aa0:       d10043ff        sub     sp, sp, #0x10
```

- The netlink commands of PoC/PWN in the user space

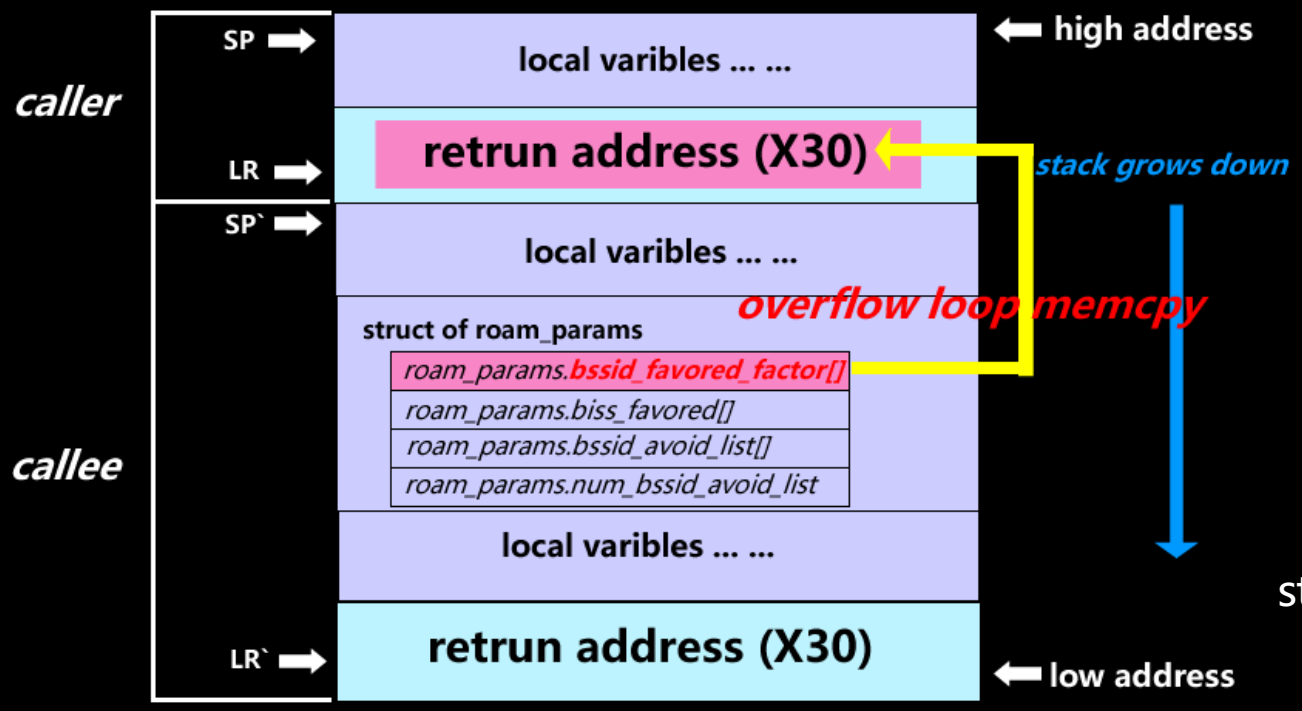
```
nla_nest_start(msg, QCA_WLAN_VENDOR_ATTR_ROAMING_PARAM_SET_BSSID_PREFS);
nla_put(msg, QCA_WLAN_VENDOR_ATTR_ROAMING_PARAM_SET_LAZY_ROAM_BSSID, SEND_DATA_LEN, &expoit_data);
nla_put_u32(msg, QCA_WLAN_VENDOR_ATTR_ROAMING_PARAM_SET_LAZY_ROAM_RSSI_MODIFIER, 1);
```

CVE-2017-0437Analysis



PoC/PWN workflow

CVE-2017-0437 Exploit



stack frame layout

caller: wlan_hdd_cfg80211_set_ext_roam_params()
callee: __wlan_hdd_cfg80211_set_ext_roam_params()

1. The roam_params.bssid_favored_fator[] overflow the caller function's return address when the loop more then 16 times.

2. using some gadgets, we could remove the address_limit of the process.

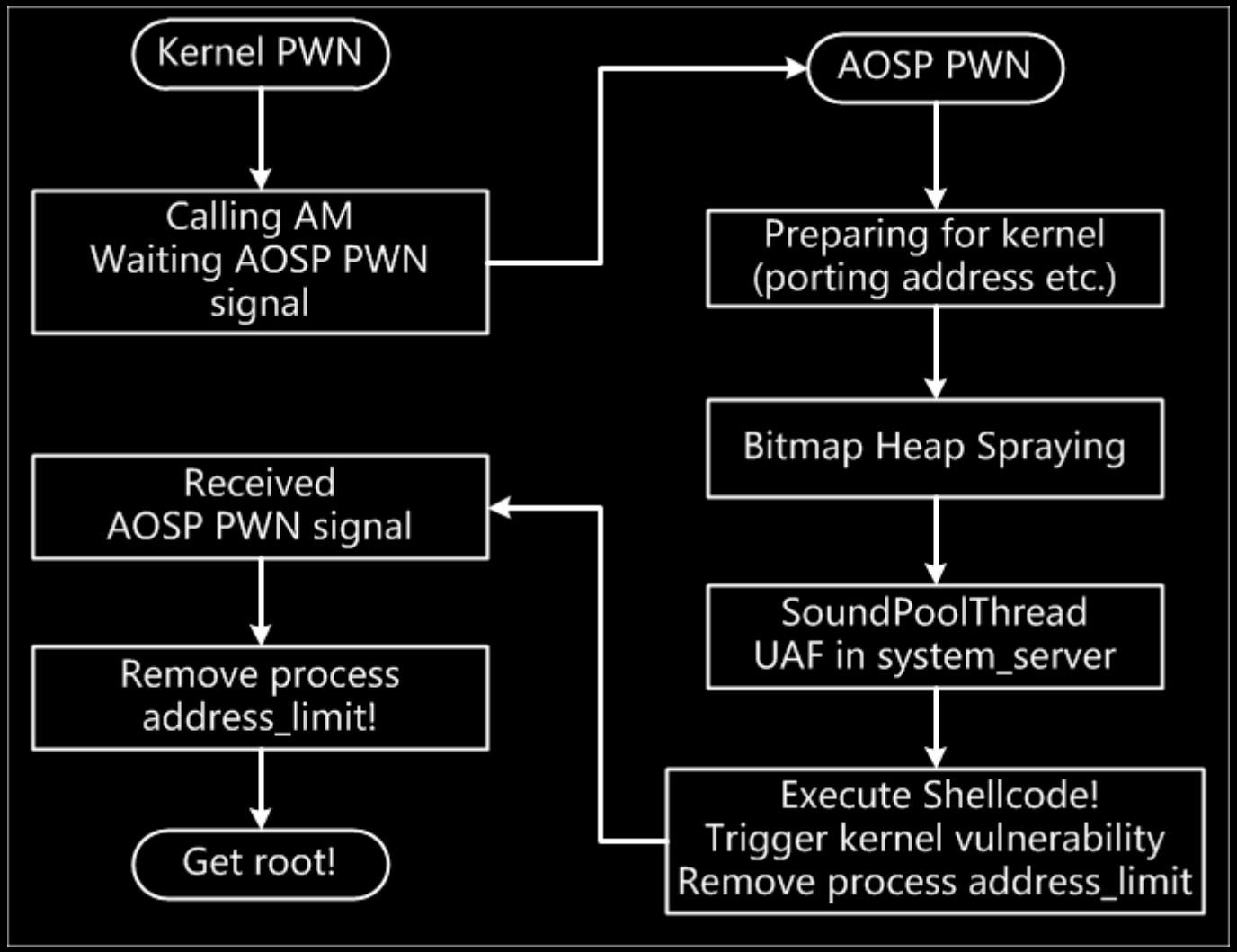
CVE-2017-0437’s patch

- The patch

```
--- a/drivers/staging/qcacld-2.0/CORE/HDD/src/wlan_hdd_cfg80211.c
+++ b/drivers/staging/qcacld-2.0/CORE/HDD/src/wlan_hdd_cfg80211.c
@@ -1799,6 +1799,7 @@ __wlan_hdd_cfg80211_set_ext_roam_params(struct wiphy *wiphy,
        struct nlattr *tb2[QCA_WLAN_VENDOR_ATTR_ROAMING_PARAM_MAX + 1];
        int rem, i;
        uint32_t buf_len = 0;
+       uint32_t count;
        int ret;

        if (VOS_FTM_MODE == hdd_get_conparam()) {
@@ -1974,15 +1975,25 @@ __wlan_hdd_cfg80211_set_ext_roam_params(struct wiphy *wiphy,
                        hddLog(LOGE, FL("attr num of preferred bssid failed"));
                        goto fail;
                }
-               roam_params.num_bssid_favored = nla_get_u32(
+               count = nla_get_u32(
                        tb[QCA_WLAN_VENDOR_ATTR_ROAMING_PARAM_SET_LAZY_ROAM_NUM_BSSID]);
+               if (count > MAX_BSSID_FAVORED) {
+                       hddLog(LOGE, FL("Preferred BSSID count %u exceeds max %u"),
+                              count, MAX_BSSID_FAVORED);
+                       goto fail;
+               }
                hddLog(VOS_TRACE_LEVEL_DEBUG,
-                       FL("Num of Preferred BSSID (%d)"),
-                       roam_params.num_bssid_favored);
+                       FL("Num of Preferred BSSID: %d"), count);
                i = 0;
                nla_for_each_nested(curr_attr,
                        tb[QCA_WLAN_VENDOR_ATTR_ROAMING_PARAM_SET_BSSID_PREFS],
                        rem) {
+
+                       if (i == count) {
+                               hddLog(LOGW, FL("Ignoring excess Preferred BSSID"));
+                               break;
```

AOSP and Kernel Combination Exploits



AOSP and Kernel Exploit Workflow

Demonstration

phybio@phybio-OptiPlex-7010: ~

phybio@phybio-OptiPlex-7010:~$

phybio@phybio-OptiPlex-7010: ~

phybio@phybio-OptiPlex-7010:~$

phybio@phybio-OptiPlex-7010:~/projects/CVE/CVE-2017-0439_wifi_vendor_commands/wifi_0437_roam_params.bssid_favored/jni$

Q & A

Thank You

C0RE Team