

CVE-2021-21781

Analysis Report

Analysis report on ARM SIGPAGE data disclosure

Caused by use of uninitialized page

In get_signal_page(void) of Kernel Signal Management

First reporter: `Lilith >_>` of "Cisco Talos Intelligence Group"

Author: iCAROS7

Data Created: 2022.06.21 Tue

Data Version: 1.0.1.1

Index

1. Introduce
2. Analysis of crash occurrence function
 - i. Basic Knowledge
 - ii. Code audit
 - iii. Vulnerability analysis
3. Crash inducement
 - i. Host PC specification
 - ii. Fuzzer specification
 - iii. Proof of Exploit
4. Patch for the vulnerability
5. Conclusion
6. Reference

1. Introduce

CVE-2021-21781 은 Userland 상의 Application 이 Return-Oriented Programing (이하 ROP)을 통해 Information leak 이 trigger 가능한 취약점이다. 이는 Process 가 최초 Signal Initialization 시 ARM command & Signal handler data 를 주고받기 위한 Page 를 Random 한 Offset 을 포함해 Allocating 하는 과정 중 Uninitialized 된 Address 가 포함된 Page 를 allocating 받아 해당 Page 에 존재하는 기존의 타 SIGPAGE Data 에 Access 가능하게 되며 이루어진다.

이는 Linux Kernel version 4.0 ~ 4.14.221, ~ 4.19.176, 5.0 ~ 5.4.98, ~ 5.10.16 그리고 ~ 5.11-rc4 까지의 범위를 지닌다.

2021 년 1 월 28 일 `Cisco Talos` 팀의 `Lilith >_>` 에 의해 발견된 후 각 Vendor 에게 First report 되었으며 2022 년 2 월 5 일 Linux main branch 에 Patch 가 commit & merge 되었다. 이후 동년 6 월 25 일 Public Release 되며 CVSS 3.0 기준 `3.3 Low`로 Scoring 되었다.

2. Analysis of crash occurrence function

```
# /Linux/arch/arm/kernel/signal.c

01 struct page *get_signal_page(void)
02 {
03     unsigned long ptr;
04     unsigned offset;
05     struct page *page;
06     void *addr;
07
08     page = alloc_pages(GFP_KERNEL, 0);
09
10     if (!page)
11         return NULL;
12
13     addr = page_address(page);
14
15     /* Give the signal return code some randomness */
16     offset = 0x200 + (get_random_int() & 0x7fc);
17     signal_return_offset = offset;
18
19     /* Copy signal return handlers into the page */
20     memcpy(addr + offset, sigreturn_codes, sizeof(sigreturn_codes));
21
22     /* Flush out all instructions in this page */
23     ptr = (unsigned long)addr + offset; // [1]
24     flush_icache_range(ptr, ptr + sizeof(sigreturn_codes)); // [2]
25
26     return page;
27 }
```

i. Basic Knowledge

Linux kernel 에서의 Signal 은 Process - Process 혹은 Kernel - Process 간 주고 받는 정보 이다.

```
# /Linux/arch/arm/kernel/process.c

01 static const struct vm_special_mapping sigpage_mapping = {
02     .name = "[sigpage]",
03     .pages = &signal_page, // [1]
04     .mremap = sigpage_mremap,
05 };
06
07
08 static struct page *signal_page;
09 extern struct page *get_signal_page(void);
10 int arch_setup_additional_pages(struct linux_binprm *bprm,
11     int uses_interp)
12 {
13     struct mm_struct *mm = current->mm;
14     struct vm_area_struct *vma;
```

```

14         unsigned long npages;
15         unsigned long addr;
16         unsigned long hint;
17         int ret = 0;
18
19         if (!signal_page)
20             signal_page = get_signal_page();    // [2]
21         if (!signal_page)
22             return -ENOMEM;
23
24         npages = 1; /* for sigpage */
25         npages += vdso_total_pages;
26
27         if (down_write_killable(&mm->mmap_sem))
28             return -EINTR;
29         hint = sigpage_addr(mm, npages);
30         addr = get_unmapped_area(NULL, hint,
31                                   npages << PAGE_SHIFT, 0, 0);
32         if (IS_ERR_VALUE(addr)) {
33             ret = addr;
34             goto up_fail;
35         }
36         vma = _install_special_mapping(mm
37                                         , addr, PAGE_SIZE, VM_READ |
38                                         VM_EXEC | VM_MAYREAD | VM_MAYWRITE |
39                                         VM_MAYEXEC, &sigpage_mapping);    // [3]

```

/Linux/mm/mmap.c

```

01         struct vm_area_struct * _install_special_mapping(
02             struct mm_struct *mm,
03             unsigned long addr, unsigned long len,
04             unsigned long vm_flags,
05             const struct vm_special_mapping *spec)    // [4]
06         {
07             return __install_special_mapping(mm,
08                                             addr, len, vm_flags, (void *)spec,
09                                             &special_mapping_vmops);

```

이 중 후자에서는 `process.c` 상의 Line 20, [2]와 같이 최초 Initialization 시 데이터를 주고 받기 위한 일정 PAGE 를 프로세스에게 allocating 한다. 이 과정에서 get_signal_page() function 을 통해 Page structor 를 Return 받는다.

이를 기반으로 Line 36, [3]에서 `sigpage_mapping` struct 가 `mm/mmap.c` 상 `_install_special_mapping()`의 arguments 로 이용되는 과정 중 Line 3, [1]와 같이 `signal_page`가 `pages`에 reference 된다.

이후 `mmap.c` 상 Line 04, [4]에서 `_install_special_mapping()` arguments 로 전달되어 Memory 의 Available address 에 allocate 된다.

ii. Code Audit

Line. 7 에서 alloc_pages() function 을 통해 Logical Page 를 return 받는다. 이후 Line. 13 과 같이 page_address() function 을 통해 첫번째 Page 의 Logical Address 를 return 받아 `addr` pointer 에 저장한다.

Line. 16 에서 기존에 받은 `SIGRET_CODE` 의 내용을 복사하기 위한 새로운 영역을 만들기 위해 특정 Offset 을 계산하여 unsigned int 형으로 `offset` 및 `signal_return_offset`에 저장 한다. 이는 int 형이 4byte 씩 메모리를 사용하고, Linux Kernel 에서 ARM Architecture 의 Unit of Page Allocating 인 4KB 로 잘라 생성하기 위함이다.

Line. 20 에서 `addr` 로부터 4byte * `offset` 의 합 Address 에 `sigreturn_codes`의 명령어들을 크기만큼 복사한다.

Line. 23 에서는 Line. 20 에서 복사된 Address 를 `ptr` 변수에 저장한다.

Line. 24 에서 flush_icache_range() function 을 통해 `ptr` ~ (`ptr` + sizeof(`sigreturn_codes`)) 까지의 Kernel 이 명령어 처리를 위한 Instruction cache 임을 명시함과 동시에 해당 영역을 Flushing 한다.

iii. Vulnerability analysis

```
ptr = (unsigned long)addr + offset;  
flush_icache_range(ptr, ptr + sizeof(sigreturn_codes));
```

실질적으로 Vulnerability 인 Code 의 경우 위와 같다. `sigreturn_codes` 는 unsigned long 형식으로 extern 형식으로 선언이 되어있다. sizeof() function 는 Pointer 의 경우 해당 Pointer 의 타입의 Size 가 반환된다.

허나 만일 sizeof(`sigreturn_codes`)가 0 인 경우에는 정상적으로 flush 가 수행되지 않으며 Abnormal 한 `page` object 가 `process.c` 의 arch_setup_additional_pages()로 return 되며 Available Memory Address 로 Mapping 되는 과정에서 flush 가 되지 않아 Uninitialized 된 메모리가 allocating 될 수 있다.

단, 이는 상기한 allocating 과정 상에서 Virtual Memory Table 상에 Available 하다 판단되는 Address + Offset 에 이전의 Flushing 되지 않은 Data 가 남아 있어야만 Read 가 가능하다.

상기한 Case 의 경우 제 3 자가 ROP Attack 을 통해 특정 Offset 상의 Kernel Memory Data 를 Read 가능한 상황이 된다.

3. Crash inducement

i. Host PC Specification

Host OS Version : Ubuntu 20.04.04 LTS (Focal Fossa)
Host Kernel Version : Linux 5.4.0-137-generic x86_64
Host CPU : Intel(R) Core(TM) i7-2620M CPU @ 2.70GHz
Host RAM : Samsung PC3L-12800 12GB
Swap File Size : 16,384MB

ii. Target Specification

Tested Architecture : arm (ARMv7, Cotrex-a15)
Tested Virtual Env : vexpress-a15
Tested Kernel Ver : Linux 5.4.66-generic
Tested Qemu Ver : 4.2.1
Enabled Sanitizer :

- KGDB (Kernel Gnu Debugger)
- KASAN (Kernel Address Sanitizer)
- UBSAN (Undefined Behavior Sanitizer)
- KCOV (Kernel Coverage)
- Kmemleak (Kernel Memory Leak Detector)

iii. Proof of Vulnerability

Linux 상에서 현재 실행 중인 process 의 정보는 `/proc/<pid>` 에서 확인이 가능하다.

```
# ps -e | grep sh
164 root      sshd: /usr/sbin/sshd [listener] 0 of 10-100 startups
167 root      -sh
202 root      grep sh

# cat /proc/167/status | grep Name
Name: sh
```

이중 `maps` 파일을 확인 시 현재 process 에 대한 Memory Map Information 을 얻을 수 있다.

```
# cat /proc/167/maps
0046d000-00533000 r-xp 00000000 b3:00 17      /bin/busybox
00542000-00544000 r--p 000c5000 b3:00 17      /bin/busybox
00544000-00545000 rw-p 000c7000 b3:00 17      /bin/busybox
00545000-0054a000 rw-p 00000000 00:00 0      [heap]
76e9b000-76f03000 r-xp 00000000 b3:00 173     /lib/libuClibc-1.0.41.so
76f03000-76f12000 ---p 00000000 00:00 0
76f12000-76f13000 r--p 00067000 b3:00 173     /lib/libuClibc-1.0.41.so
76f13000-76f14000 rw-p 00068000 b3:00 173     /lib/libuClibc-1.0.41.so
76f14000-76f2a000 rw-p 00000000 00:00 0
76f2a000-76f30000 r-xp 00000000 b3:00 163     /lib/ld-uClibc-1.0.41.so
76f3e000-76f40000 rw-p 00000000 00:00 0
76f40000-76f41000 r--p 00006000 b3:00 163     /lib/ld-uClibc-1.0.41.so
76f41000-76f42000 rw-p 00007000 b3:00 163     /lib/ld-uClibc-1.0.41.so
7ec5e000-7ec7f000 rw-p 00000000 00:00 0      [stack]
7ed2c000-7ed2d000 r-xp 00000000 00:00 0      [sigpage]
7ed2d000-7ed2e000 r--p 00000000 00:00 0      [vvar]
7ed2e000-7ed2f000 r-xp 00000000 00:00 0      [vdso]
fffff000-fffff1000 r-xp 00000000 00:00 0      [vectors]
```

예시로 pid 167 로 실행 중인 `/bin/sh`에 대한 Memory Map Information 을 출력 시 위와 같다. 여기서 `[sigpage]` Data 가 담긴 0x7ed2c000 - 0x7ed2d000 의 Virtual Memory 가 allocating 된 것을 확인 가능 하다.

.

KGDB 를 통해 해당 Memory Address 의 Data 를 확인 시 위와 같다.

4. Patch for the vulnerability

```
# /Linux/arch/arm/kernel/signal.c

struct page *get_signal_page(void)
{
    unsigned long ptr;
    unsigned offset;
    struct page *page;
    void *addr;

    addr = page_address(page);

    // memset32 를 통하여 uint32_t 로 Memory area Filling
    /* Poison the entire page */
    + memset32(addr, __opcode_to_mem_arm(0xe7fddef1),
    +          PAGE_SIZE / sizeof(u32)); // [1]
    +
    /* Give the signal return code some randomness */
    offset = 0x200 + (get_random_int() & 0x7fc);
    signal_return_offset = offset;

    - /*
    -  * Copy signal return handlers into the vector page, and
    -  * set sigreturn to be a pointer to these.
    -  */
    + /* Copy signal return handlers into the page */
    memcpy(addr + offset, sigreturn_codes, sizeof(sigreturn_codes));

    - ptr = (unsigned long)addr + offset;
    - flush_icache_range(ptr, ptr + sizeof(sigreturn_codes));
    + /* Flush out all instructions in this page */
    // offset 값을 통한 ROP Attack Prevent
    + ptr = (unsigned long)addr; // [2]
    // `
    + flush_icache_range(ptr, ptr + PAGE_SIZE); // [3]

    return page;
}
```

[1]과 같이 사전에 `memset32()` Function 을 통해 uint32_t 로 Memory area 를 4 byte 단위로 Filling 한다.

[2]에서 `offset`를 지우는 것으로 인해 Trigger 될 수 있는 ROP Attack 을 prevent 가능하다. 이후 [3]에서 기존 `sizeof`가 아닌 `PAGE_SIZE` 단위로 Flushing 을 진행하여 Uninitialization 을 prevent 한다.

5. Conclusion

위 Exploit 은 실질적으로 Exploitability 한 상황이 Local 상에서 쉬운 난이도로 별도의 Privileges 이나 User Interaction 없이 가능하지만 System 의 Confidentiality / Integrity / Availability 에 영향을 주지 않아 낮게 Scoring 되었다. USD \$0 ~ \$5K 로 추정 Bounty 만 존재 할 뿐 정확한 Bounty 정보는 공개되지 않았다.

6. Reference

- <https://github.com/torvalds/linux/commits/master/arch/arm/kernel/signal.c>
- <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-0185>
- <https://ubuntu.com/security/CVE-2021-21781>
- <https://elixir.bootlin.com/linux>
- <https://www.cve.org/CVERecord?id=CVE-2021-21781>
- <https://nvd.nist.gov/vuln/detail/CVE-2021-21781>
- <https://cve.report/CVE-2021-21781>
- <https://access.redhat.com/security/cve/cve-2021-21781>
- https://bugzilla.redhat.com/show_bug.cgi?id=1981950
- <https://vulners.com/cve/CVE-2021-21781>
- https://books.google.co.kr/books/about/코드로_알아보는_ARM_리눅스_커.html
- <https://cpuu.postype.com/post/9075747>
- <https://blog.daum.net/tlos6733/188>
- <https://wogh8732.tistory.com/395>
- https://github.com/google/syzkaller/blob/master/docs/linux/setup_linux-

[host_gemu-vm_arm-kernel.md](#)