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

Index

1. Introduce

CVE-2021-21781 은 Userland 상의 Application 이 Return-Oriented Programing 을 통해 trigger 되는 Leak of SIGPAGE data 취약점이다. 이는 Process 가 최초 Signal Initialization 시 ARM command & Signal handler data 를 주고 받기 위한 Page 를 Random 한 Offset 을 포함해 Allocating 하는 과정 중 Uninitialized 된 Address 가 포함된 Page 를 할당 받아 해당 Page 에 존재하는 기존의 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.

이는 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.

메모 포함[RH1]: 어찌 Exploit 이 되는가에 대한 설명도 없는 문단
맥락에 맞지 않음

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 struct page *signal_page;
02    extern struct page *get_signal_page(void);
03    int arch_setup_additional_pages(struct linux_binprm *bprm,
              int uses_interp)
04    {
05            struct mm_struct *mm = current->mm;
06            struct vm_area_struct *vma;
07            unsigned long npages;
08            unsigned long addr;
09            unsigned long hint;
10            int ret = 0;
11
12            if (!signal_page)
13                    signal_page = get_signal_page();    // [1]
```

```
14              if (!signal_page)
15                      return -ENOMEM;
16
17              npages = 1; /* for sigpage */
18              npages += vdso_total_pages;
19
20              if (down_write_killable(&mm->mmap_sem))
21                      return -EINTR;
22              hint = sigpage_addr(mm, npages);
23              addr = get_unmapped_area(NULL, hint,
                        npages << PAGE_SHIFT, 0, 0);
24      if (IS_ERR_VALUE(addr)) {
25                      ret = addr;
26                      goto up_fail;
27              }
28
29              vma = _install_special_mapping(mm
                        , addr, PAGE_SIZE, VM_READ |
                        VM_EXEC | VM_MAYREAD | VM_MAYWRITE |
                        VM_MAYEXEC, &sigpage_mapping);
30
31              if (IS_ERR(vma)) {
32                      ret = PTR_ERR(vma);
33                      goto up_fail;
34              }
35
36              mm->context.sigpage = addr;
37
38              /* Unlike the sigpage, failure to install the
                        vdso is unlikely
39               * to be fatal to the process, so
                        no error check needed
40               * here.
41               */
42              arm_install_vdso(mm, addr + PAGE_SIZE);
43
44      up_fail:
45              up_write(&mm->mmap_sem);
46              return ret;
47      }
```

이 중 후자에서는 Line 13, [1]와 같이 최초 Initialization 시 데이터를 주고 받기 위한 일정 PAGE 를 프로세스에게 allocating 한다. 이 과정에서 get_signal_page() 를 통해 Page 구조체를 Return 받는다.

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. 허나 만일 sizeof(`sigreturn_codes`)가 0 인 경우에는 정상적으로 flush 가 수행되지 않으며 Abnormal 한 `page` 객체가 process.c 의 arch_setup_additional_pages()로 반환 된다.

iii. Vulnerability analysis

```
01    extern const unsigned long sigreturn_codes[17];
02
03    ptr = (unsigned long)addr + offset;
04    flush_icache_range(ptr, ptr +
            sizeof(sigreturn_codes));
```

실질적으로 Vulnerability 인 Code 의 경우 [1], [2] 부분으로 위와 같다. `sigreturn_codes` 는 unsigned long 형식으로 extern 형식으로 선언이 되어있다. sizeof() function 는 Pointer 의 경우 해당 Pointer 의 타입의 Size 가 반환 된다.

따라서

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-gerneric 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. Fuzzer Specification

Tested Kernel Ver:	Linux 5.4.66-gerneric arm
Fuzzer:	.
Enabled Sanitizer:	.

01Proof of Concept

???