“Baron Samedit”

CVE-2021-3156

#Floodnut

https://github.com/Floodnut/papers_documents_Analysis/

1. 취약점 분석

CVE – 2021 - 3156

취약점 발생 순서 및 조건

1. SUDO의 플래그 검증을 우회해야 함.

2. HEAP 버퍼 변수인 user_args에서 HEAP overflow를 발생 시켜야 함.

3. overflow가 ASLR 및 메모리 보호를 우회하여 원하는 HEAP 청크를 덮을 수 있어야 함.

1. 취약점 분석

CVE – 2021 - 3156

SUDO의 경우

(1.)MODE_SHELL (-s 옵션) 과 MODE_EDIT (-e 옵션) 이 같이 설정될 경우

-> MODE_SHELL 플래그 삭제 및 에러 처리

입력한 인자 : "AAAA₩"

parse_args()를 거쳐 "₩"가 이스케이프 처리됨.

set_cmnd로 넘어가는 인자 배열 : "AAAA₩₩"

```
if (ISSET(mode, MODE_RUN) && ISSET(flags, MODE_SHELL)) {
char **av, *cmnd = NULL;
int ac = 1;

if (argc != 0) {
    /* shell -c "command" */
    char *src, *dst;
    size_t cmnd_size = (size_t) (argv[argc - 1] - argv[0]) +
    strlen(argv[argc - 1]) + 1;

    cmnd = dst = reallocarray(NULL, cmnd_size, 2);
    if (cmnd == NULL)
    sudo_fatalx(U_("%s: %s"), __func__, U_("unable to allocate memory"));
    if (!gc_add(GC_PTR, cmnd))
    exit(1);

    for (av = argv; *av != NULL; av++) {
        for (src = *av; *src != '\0'; src++) {
        /* quote potential meta characters */
        if (!isalnum((unsigned char)*src) && *src != '_' && *src != '-' && *src != '$')
        *dst++ = '\\';
    *dst++ = *src;
    }
```

parse_args.c – parse_args()

1. 취약점 분석

CVE – 2021 - 3156

SUDOEDIT의 경우

(1.)SUDOEDIT이 MODE_SHELL이 설정된 상태라도 MODE_EDIT을 설정함.

1번 조건 우회(sudoedit –s)

```
root          22 Jun 18  2020  strip -> x86_64-linux-gnu
root      149080 Feb  1  2020  sudo*
root           4 Feb  1  2020  sudoedit -> sudo*
root       56128 Feb  1  2020  sudoreplay*
```

/usr/bin/

```
/* First, check to see if we were invoked as "sudoedit". */
proglen = strlen(progname);
if (proglen > 4 && strcmp(progname + proglen - 4, "edit") == 0) {
progname = "sudoedit";
mode = MODE_EDIT;
sudo_settings[ARG_SUDOEDIT].value = "true";
}
```

parse_args.c – parse_args( )

1. 취약점 분석

CVE – 2021 - 3156

취약점 발생

(2.) user_args(to)에 저장될 인자 배열(from)의 마지막 문자가 '₩' 일 경우?

배열 종료를 위한 NULL 바이트가 배열을 종료 시키지 못하고 user_args에 복사 됨.

```
/* set user_args */
if (NewArgc > 1) {
    char *to, *from, **av;
    size_t size, n;

    /* Alloc and build up user_args. */
    for (size = 0, av = NewArgv + 1; *av; av++)
        size += strlen(*av) + 1;
    if (size == 0 || (user_args = malloc(size)) == NULL) {
        sudo_warnx(U_("%s: %s"), __func__, U_("unable to allocate memory"));
        debug_return_int(-1);
    }
    if (ISSET(sudo_mode, MODE_SHELL|MODE_LOGIN_SHELL)) {
    /*
     * When running a command via a shell, the sudo front-end
     * escapes potential meta chars.  We unescape non-spaces
     * for sudoers matching and logging purposes.
     */
        for (to = user_args, av = NewArgv + 1; (from = *av); av++) {
            while (*from) {
                if (from[0] == '\\' && !isspace((unsigned char)from[1]))
                    from++;
                *to++ = *from++;
            }
            *to++ = ' ';
        }
            *--to = '\0';
```

sudoers.c – set_cmnd()

1. 취약점 분석

CVE – 2021 - 3156

```
for (to = user_args, av = NewArgv + 1; (from = *av); av++) {
    while (*from) {
        if (from[0] == '\\' && !isspace((unsigned char)from[1]))
            from++;
        *to++ = *from++;
    }
    *to++ = ' ';
}
```

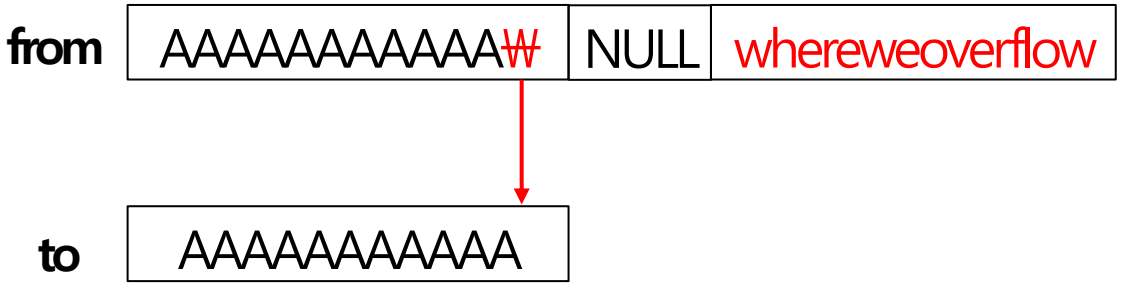
sudoers.c – set_cmnd()

1. 취약점 분석

CVE – 2021 - 3156

```
for (to = user_args, av = NewArgv + 1; (from = *av); av++) {
    while (*from) {
        if (from[0] == '\\' && !isspace((unsigned char)from[1]))
            from++;
        *to++ = *from++;
    }
    *to++ = ' ';
}
```

sudoers.c – set_cmnd()

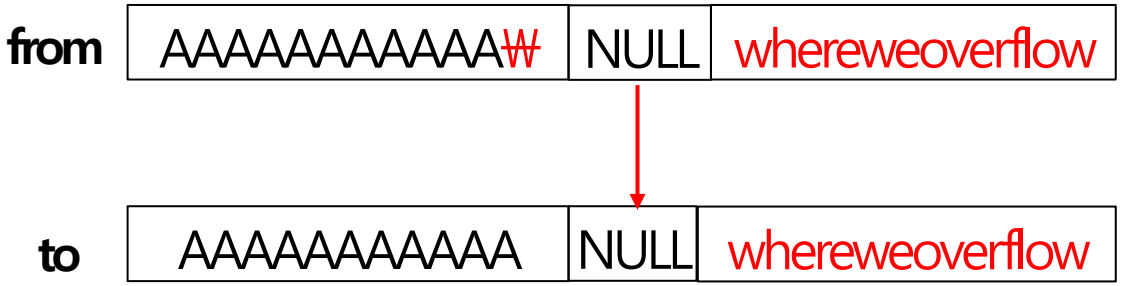


1. 취약점 분석

CVE – 2021 - 3156

```
for (to = user_args, av = NewArgv + 1; (from = *av); av++) {
    while (*from) {
        if (from[0] == '\\' && !isspace((unsigned char)from[1]))
            from++;
        *to++ = *from++;
    }
    *to++ = ' ';
}
```

sudoers.c – set_cmnd()



1. 취약점 분석

CVE – 2021 - 3156

```
for (to = user_args, av = NewArgv + 1; (from = *av); av++) {
    while (*from) {
        if (from[0] == '\\' && !isspace((unsigned char)from[1]))
            from++;
        *to++ = *from++;
    }
    *to++ = ' ';
}
```

sudoers.c – set_cmnd()

1. 취약점 분석

CVE – 2021 - 3156

HEAP fengshui

setlocale( ) 로 인해 HEAP 영역에
LC_CTYPE, LC_MESSAGES, LC_TIME 등의 환경 변수가 할당 및 해제

```
setlocale(LC_ALL, "");
bindtextdomain(PACKAGE_NAME, LOCALEDIR);
textdomain(PACKAGE_NAME);
```

sudo.c – int main()

1. 취약점 분석

CVE – 2021 - 3156

HEAP fengshui

setlocale( ) 로 인해 HEAP 영역에
LC_CTYPE, LC_MESSAGES, LC_TIME 등의 환경 변수가 할당 및 해제

```
char messages[0xe0] = {"LC_MESSAGES=en_GB.UTF-8@"};
memset(messages + strlen(messages), 'A', 0xb8);

char telephone[0x50] = {"LC_TELEPHONE=C.UTF-8@"};
memset(telephone + strlen(telephone), 'A', 0x28);

char measurement[0x50] = {"LC_MEASUREMENT=C.UTF-8@"};
memset(measurement + strlen(measurement), 'A', 0x28);
```

1. 취약점 분석

CVE – 2021 - 3156

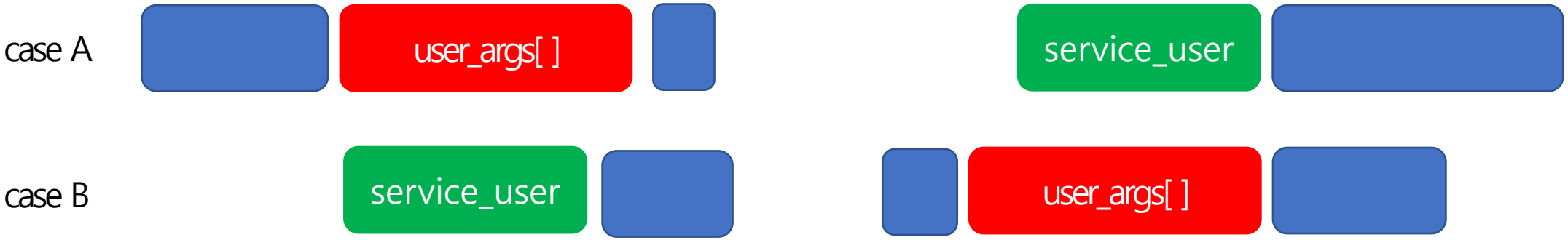
HEAP fengshui

SUDO는 NSS를 활용하여 HEAP 영역에 service_user를 배치시킴

->NSS = NameServiceSwitch

```
typedef struct service_user
{
  struct service_user *next;
  lookup_actions actions[5];
  service_library *library;
  void *known;
  char name[0];
} service_user;
```

service_user



1. 취약점 분석

CVE – 2021 - 3156

HEAP fengshui

NSS의 파일 변조를 통해 공격자의 권한을 ROOT로 격상시킴.

user_args[ ]

service_user

```
static int
nss_load_library (service_user *ni)
{
  if (ni->library == NULL)
    {
      /* This service has not yet been used.  Fetch the service
         library for it, creating a new one if need be.  If there
         is no service table from the file, this static variable
         holds the head of the service_library list made from the
         default configuration.  */
      static name_database default_table;
      ni->library = nss_new_service (service_table ?: &default_table,
                                     ni->name);
      if (ni->library == NULL)
        return -1;
    }

  if (ni->library->lib_handle == NULL)
    {
      /* Load the shared library.  */
      size_t shlen = (7 + strlen (ni->name) + 3
                      + strlen (__nss_shlib_revision) + 1);
      int saved_errno = errno;
      char shlib_name[shlen];

      /* Construct shared object name.  */
      __stpcpy (__stpcpy (__stpcpy (__stpcpy (shlib_name,
                                              "libnss_"),
                                    ni->name),
                          ".so"),
                __nss_shlib_revision);

      ni->library->lib_handle = __libc_dlopen (shlib_name);
      if (ni->library->lib_handle == NULL)
```