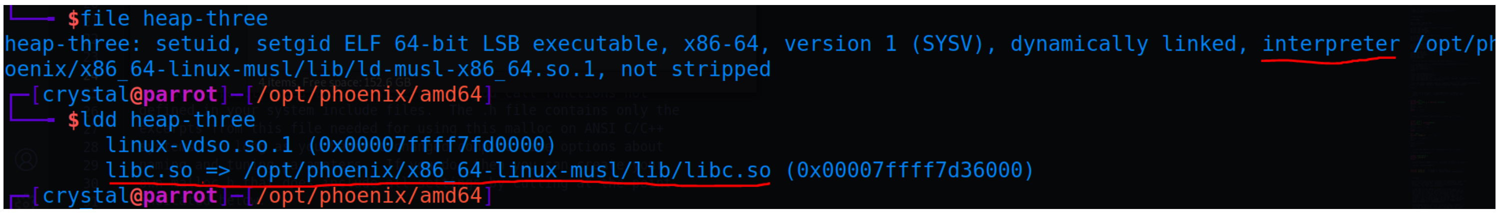
setup

https://github.com/ExploitEducation/Phoenix/releases

LiveOverflow - https://www.youtube.com/watch?v=gL45bjQvZSU
https://blog.lamarranet.com/index.php/exploit-education-phoenix-heap-three-solution
https://n1ght-w0lf.github.io/binary%20exploitation/heap-three
https://www.lucas-bader.com/ctf/2019/05/02/heap3
https://airman604.medium.com/protostar-heap-3-walkthrough-56d9334bcd13

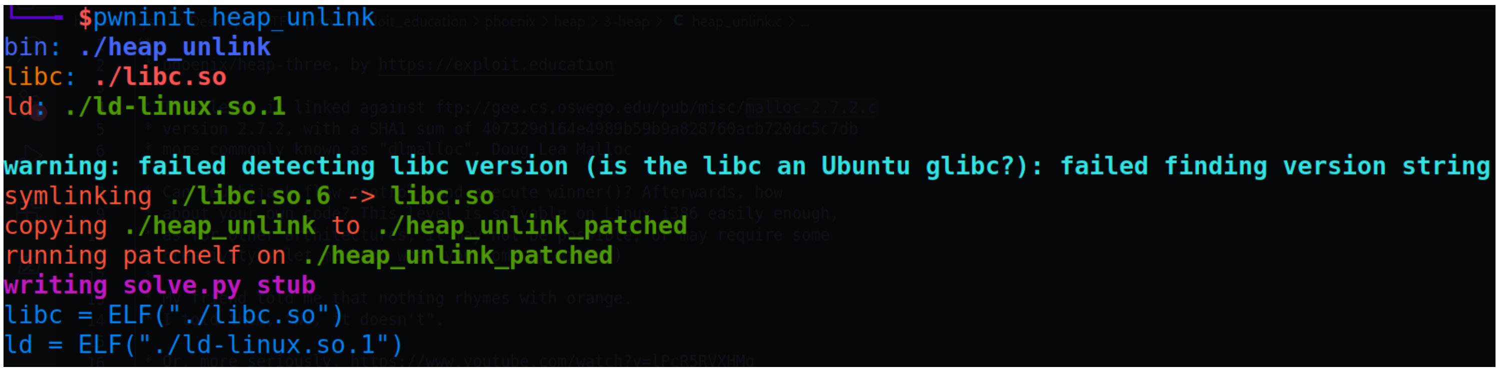
NOTE: As mentioned in the C file, this challenge uses an old version of malloc (2.7.2), known as dmalloc(). I tried to copy the following two files from the official challenge VM (I installed the .deb file from GitHub):

interpreter /opt/phoenix/x86_64-linux-musl/lib/ld-musl-x86_64.so.1
libc.so => /opt/phoenix/x86_64-linux-musl/lib/libc.so



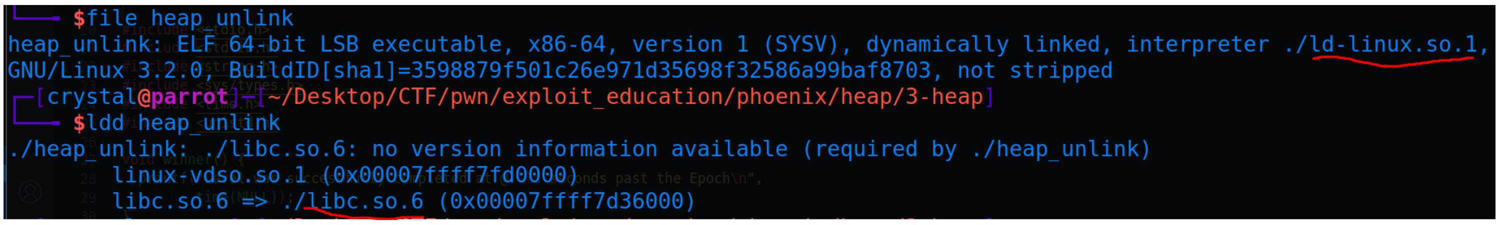
```
└──╼ $file heap-three
heap-three: setuid, setgid ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /opt/ph
oenix/x86_64-linux-musl/lib/ld-musl-x86_64.so.1, not stripped
┌─[crystal@parrot]─[/opt/phoenix/amd64]
└──╼ $ldd heap-three
        linux-vdso.so.1 (0x00007ffff7fd0000)
        libc.so => /opt/phoenix/x86_64-linux-musl/lib/libc.so (0x00007ffff7d36000)
┌─[crystal@parrot]─[/opt/phoenix/amd64]
```

I renamed the first file to "ld-linux.so.1" then used pwninit to patch the binary:

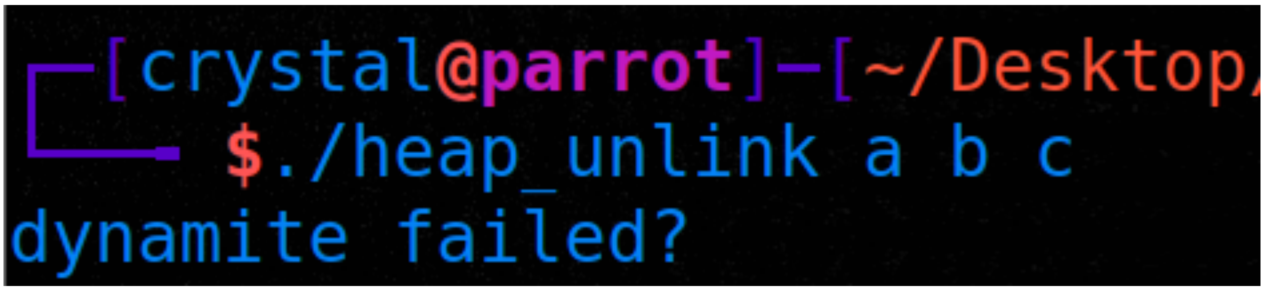


```
└──╼ $pwninit heap_unlink
bin: ./heap_unlink
libc: ./libc.so
ld: ./ld-linux.so.1

warning: failed detecting libc version (is the libc an Ubuntu glibc?): failed finding version string
symlinking ./libc.so.6 -> libc.so
copying ./heap_unlink to ./heap_unlink_patched
running patchelf on ./heap_unlink_patched
writing solve.py stub
libc = ELF("./libc.so")
ld = ELF("./ld-linux.so.1")
```



```
└──╼ $file heap_unlink
heap_unlink: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter ./ld-linux.so.1,
GNU/Linux 3.2.0, BuildID[sha1]=3598879f501c26e971d35698f32586a99baf8703, not stripped
┌─[crystal@parrot]─[~/Desktop/CTF/pwn/exploit_education/phoenix/heap/3-heap]
└──╼ $ldd heap_unlink
./heap_unlink: ./libc.so.6: no version information available (required by ./heap_unlink)
        linux-vdso.so.1 (0x00007ffff7fd0000)
        libc.so.6 => ./libc.so.6 (0x00007ffff7d36000)
```



```
┌─[crystal@parrot]─[~/Desktop
└──╼ $./heap_unlink a b c
dynamite failed?
```

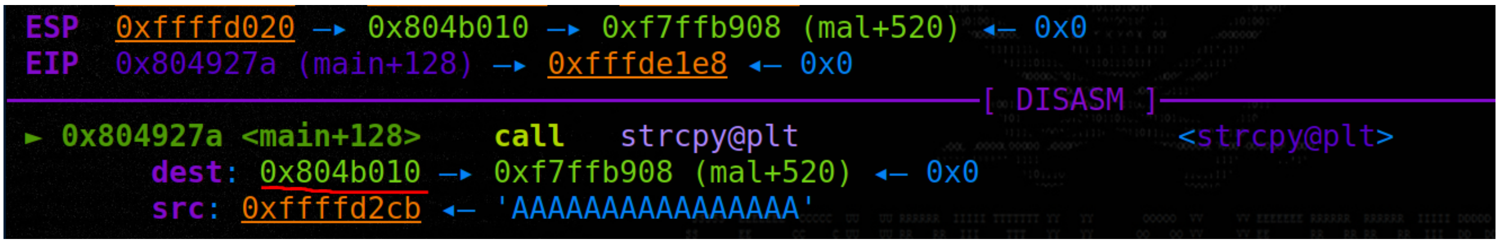
Appears to be working, should be able to use the binary without VM, providing the lib-c and ld are both in the same directory and the binary is patched to use them :)

UPDATE: I initially tried to recompile the C file (with some extra comments and parameter validation) but it did not work. Presumably need to compile with different GCC?? Anyway, here's what happened when I tried that:

Set a breakpoint at each of the 3 strcpy:

break *0x0804927a
break *0x08049291
break *0x080492a8

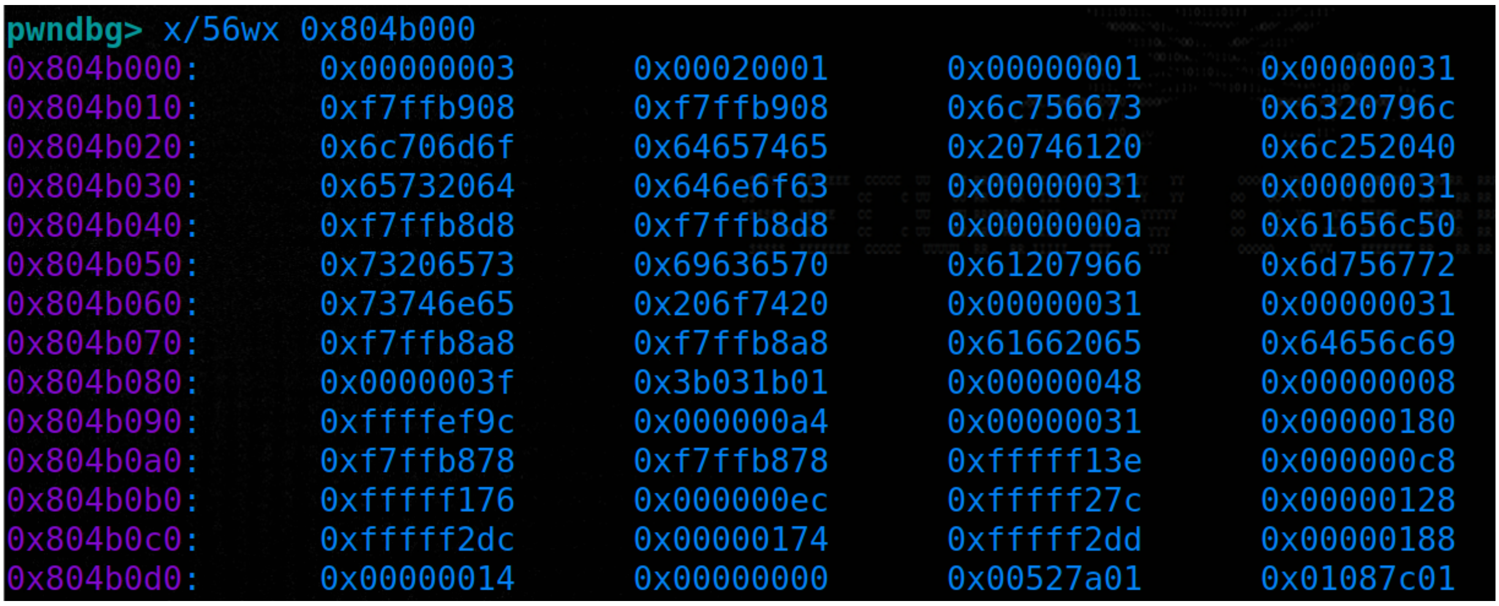
run AAAAAAAAAAAAAAAA BBBBBBBBBBBBBBBB CCCCCCCCCCCCCCCC



```
 ESP  0xffffd020 —▸ 0x804b010 —▸ 0xf7ffb908 (mal+520) ◂— 0x0
 EIP  0x804927a (main+128) —▸ 0xfffde1e8 ◂— 0x0
─────────────────────────────────────────────────[ DISASM ]──────────────
 ► 0x804927a <main+128>     call   strcpy@plt                  <strcpy@plt>
        dest: 0x804b010 —▸ 0xf7ffb908 (mal+520) ◂— 0x0
        src: 0xffffd2cb ◂— 'AAAAAAAAAAAAAAAA'
```

In the screenshot we can see the first parameter being copied to the first chunk (a)

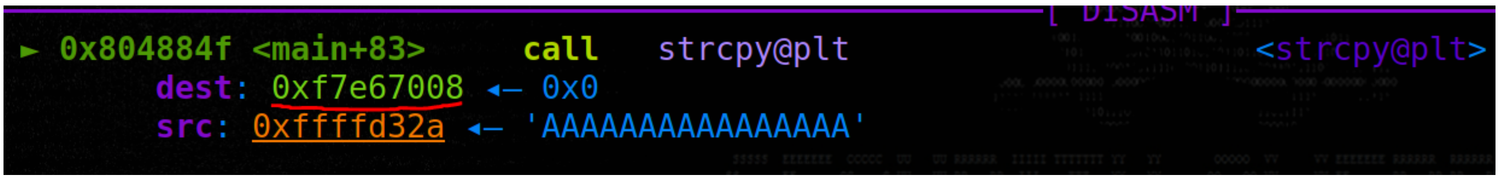
x/56wx 0x804b000



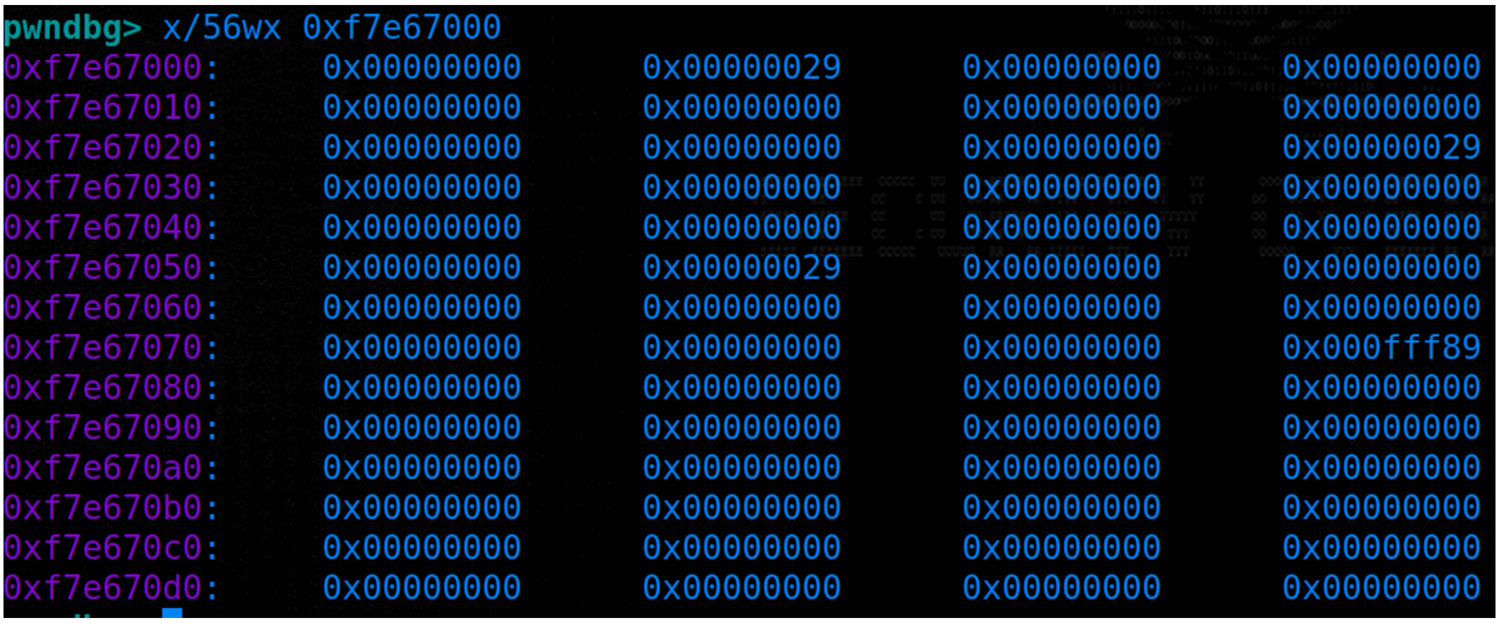
```
pwndbg> x/56wx 0x804b000
0x804b000:      0x00000003      0x00020001      0x00000001      0x00000031
0x804b010:      0xf7ffb908      0xf7ffb908      0x6c756673      0x6320796c
0x804b020:      0x6c706d6f      0x64657465      0x20746120      0x6c252040
0x804b030:      0x65732064      0x646e6f63      0x00000031      0x00000031
0x804b040:      0xf7ffb8d8      0xf7ffb8d8      0x0000000a      0x61656c50
0x804b050:      0x73206573      0x69636570      0x61207966      0x6d756772
0x804b060:      0x73746e65      0x206f7420      0x00000031      0x00000031
0x804b070:      0xf7ffb8a8      0xf7ffb8a8      0x61662065      0x64656c69
0x804b080:      0x0000003f      0x3b031b01      0x00000048      0x00000008
0x804b090:      0xfffffef9c     0x000000a4      0x00000031      0x00000180
0x804b0a0:      0xf7ffb878      0xf7ffb878      0xfffff13e      0x000000c8
0x804b0b0:      0xfffff176      0x000000ec      0xfffff27c      0x00000128
0x804b0c0:      0xfffff2dc      0x00000174      0xfffff2dd      0x00000188
0x804b0d0:      0x00000014      0x00000000      0x00527a01      0x01087c01
```

Doesn't look right... LiveOverflows is all zeroes at this stage.

Tried with the official VM binary and get a very different result:



```
──────────────────────────────────────────────────────[ DISASM ]──────────
 ► 0x804884f <main+83>      call   strcpy@plt                    <strcpy@plt>
        dest: 0xf7e67008 ◂— 0x0
        src: 0xffffd32a ◂— 'AAAAAAAAAAAAAAAA'
```



```
pwndbg> x/56wx 0xf7e67000
0xf7e67000:     0x00000000      0x00000029      0x00000000      0x00000000
0xf7e67010:     0x00000000      0x00000000      0x00000000      0x00000000
0xf7e67020:     0x00000000      0x00000000      0x00000000      0x00000029
0xf7e67030:     0x00000000      0x00000000      0x00000000      0x00000000
0xf7e67040:     0x00000000      0x00000000      0x00000000      0x00000000
0xf7e67050:     0x00000000      0x00000029      0x00000000      0x00000000
0xf7e67060:     0x00000000      0x00000000      0x00000000      0x00000000
0xf7e67070:     0x00000000      0x00000000      0x00000000      0x000fff89
0xf7e67080:     0x00000000      0x00000000      0x00000000      0x00000000
0xf7e67090:     0x00000000      0x00000000      0x00000000      0x00000000
0xf7e670a0:     0x00000000      0x00000000      0x00000000      0x00000000
0xf7e670b0:     0x00000000      0x00000000      0x00000000      0x00000000
0xf7e670c0:     0x00000000      0x00000000      0x00000000      0x00000000
0xf7e670d0:     0x00000000      0x00000000      0x00000000      0x00000000
```

I guess the binary itself is the issue, I can't compiled it with my lib-c library and have it produce the same result :(

Not sure how to get around this, so I'll just use the official binaries (still linked with the appropriate LD and Lib-C)

ANOTHER UPDATE: I spent a full day trying to get exploits to work with the official, everything looked good right up to executing the shellcode then segfaulted with no explanation. Eventually I gave up and used the QEMU VM from https://exploit.education/downloads which immediately worked fine with my exploit script.