./level09

In the level09 directory, reminiscent of the previous level, we encountered an executable named *“level09”* and a file labeled *“token”*. Unlike before, however, we were able to read the token this time, but its contents appeared obfuscated, potentially due to an overflow.

```
level09@SnowCrash:~$ cat token
f4kmm6p|=□p□n□□DB□Du{□□

level09@SnowCrash:~$ ./level09 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~□□□□□□□□□□□□□□□□□□
□□□□□□
```

Upon further inspection of the executable, we found that it processes a string argument in a specific way: it takes each character and adds its index position to its ASCII value. This transformation sometimes leads to an overflow, causing certain characters to be unreadable.

Given the behavior of the executable we hypothesized that the *“token”* file might have been obfuscated using the same mechanism. To decrypt its contents, we crafted a C program to reverse this transformation:

```c
#include <stdio.h>

int main() {
    FILE *file = fopen(“token”, “r”);
    if (file == NULL) {
        fprintf(stderr, “Could not open file\n”);
        return 1;
    }

    int ch;
    int i = 0;

    while ((ch = fgetc(file)) != EOF) {
        printf(“%c”, ch - i);
        i++;
    }

    fclose(file);
    return 0;
}
```

By running this program on the *“token”* file, we successfully unmasked its contents, unveiling the flag09 user password.

```
level09@SnowCrash:/var/tmp$ gcc exploit.c -o exploit

level09@SnowCrash:/var/tmp$ ./exploit ~/token
f3iji1ju5yuevaus41q1afiuq

level09@SnowCrash:/var/tmp$ su flag09
Password: f3iji1ju5yuevaus41q1afiuq
Don’t forget to launch getflag !

flag09@SnowCrash:~$ getflag
Check flag.Here is your token : s5cAJpM8ev6XHw998pRWG728z

flag09@SnowCrash:~$ su level10
Password: s5cAJpM8ev6XHw998pRWG728z

level10@SnowCrash:~$
```