

ctfd-plus

Let's look at the decompiled output.

Our input gets compared with the result of `somefunction(buf[i])`, character by character.

```
do {
    cVar1 = somefunction(buf[i]);
    if (cVar1 != userInput[i]) {
        puts("Incorrect flag.");
        return 0;
    }
    i = i + 1;
} while (i != 47);
puts("You got the flag! Unfortunately we don't exactly have a database to store the solve in...");
;

int somefunction(uint buf1char)
{
    uint uVar1;
    int i;
    byte x;

    uVar1 = 0;
    i = 0;
    do {
        x = (byte)i & 0x1f;
        i = i + 1;
        buf1char = (buf1char * buf1char >> x | buf1char * buf1char << 32 - x) * 0x1337 + 0x4201337 ^
            uVar1;
        uVar1 = uVar1 + 0x13371337;
    } while (i != 32);
    return (buf1char >> 8) + (buf1char >> 16) + buf1char + (buf1char >> 24);
}
```

We don't really have to understand the logic of the function. We can just use GDB and set a breakpoint right after the call returns and stops at the CMP operation. The result of the function call is stored in RAX, but we are only interested in the lower byte AL.



```
Breakpoint 6, 0x00005555555510b in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
[ REGISTERS / show-flags off / show-compact-regs off ]
*RAX 0xe900e26c
*RBX 0x7fffffffdbc0 ← 0x74736574 /* 'test' */;
*RCX 0x20
*RDX 0xe817
*RDI 0xe8
*RSI 0x0
*R8 0x555555558060 ← 0xc2bb0ce99c7f9274
*R9 0x0
*R10 0x7ffff7ddd68 ← 0x10001a000008b5
*R11 0x7ffff7f3d030 (___strcsn_sse42) ← cmp byte ptr [rsi], 0
R12 0x0
R13 0x7ffff7ffded8 → 0x7ffff7ffe259 ← 'COLORFGBG=15;0'
R14 0x0
R15 0x7ffff7ffd020 (_rtld_global) → 0x7ffff7ffe2e0 → 0x555555554000 ← 0x10102464c457f
RBP 0x1
RSP 0x7fffffffdbc0 ← 0x74736574 /* 'test' */
*RIP 0x5555555510b ← cmp al, byte ptr [rbx + rsi]
```

The first character returned from the function call is (0x6c) 'l'.

Use 'l' as the first character for our input in the next run.

```

[RAX] [ RBX ] [ RCX ] [ RDX ] [ RDI ] [ RSI ] [ R8 ] [ R9 ] [ R10 ] [ R11 ] [ R12 ] [ R13 ] [ R14 ] [ R15 ] [ RBP ] [ RSP ] [ RIP ]
0x955ddd61 0x7fffffffcb0 0x20 0x94c8 0x94 0x1 0x55555558060 0x0 0x7ffff7ddd68 0x7ffff7f3d030 0x0 0x7ffff7fdded8 0x0 0x7ffff7ffd020 0x1 0x7fffffffcb0 0x5555555510b
9 char userInput [256];
10 puts("Welcome to CTFd+!");
11 puts("So far, we only have one challenge, which is one more than the num
12 puts(
13     "So far, we only have one challenge, which is one more than the num
14 );
15 Pwn - 500 points, 0 solves");
16 __strcspn_sse42) <- cmp byte ptr [rsi], 0");
17 puts("#include <stdio.h>\nint main(void) {\n    puts(\"Bye!\");\n    re
18 fgets(userInput, 256, stdin);
19 stars = strcsplit(userInput, '\n');
20 stars = strcsplit(userInput, '\n');
21 stars = strcsplit(userInput, '\n');
22 stars = strcsplit(userInput, '\n');
23 stars = strcsplit(userInput, '\n');
24 stars = strcsplit(userInput, '\n');
25 stars = strcsplit(userInput, '\n');
26 stars = strcsplit(userInput, '\n');
27 stars = strcsplit(userInput, '\n');
28 stars = strcsplit(userInput, '\n');
29 stars = strcsplit(userInput, '\n');
30 stars = strcsplit(userInput, '\n');
31 stars = strcsplit(userInput, '\n');
32 stars = strcsplit(userInput, '\n');
33 stars = strcsplit(userInput, '\n');
34 stars = strcsplit(userInput, '\n');
35 stars = strcsplit(userInput, '\n');
36 stars = strcsplit(userInput, '\n');
37 stars = strcsplit(userInput, '\n');
38 stars = strcsplit(userInput, '\n');
39 stars = strcsplit(userInput, '\n');
40 stars = strcsplit(userInput, '\n');
41 stars = strcsplit(userInput, '\n');
42 stars = strcsplit(userInput, '\n');
43 stars = strcsplit(userInput, '\n');
44 stars = strcsplit(userInput, '\n');
45 stars = strcsplit(userInput, '\n');
46 stars = strcsplit(userInput, '\n');
47 stars = strcsplit(userInput, '\n');
48 stars = strcsplit(userInput, '\n');
49 stars = strcsplit(userInput, '\n');
50 stars = strcsplit(userInput, '\n');
51 stars = strcsplit(userInput, '\n');
52 stars = strcsplit(userInput, '\n');
53 stars = strcsplit(userInput, '\n');
54 stars = strcsplit(userInput, '\n');
55 stars = strcsplit(userInput, '\n');
56 stars = strcsplit(userInput, '\n');
57 stars = strcsplit(userInput, '\n');
58 stars = strcsplit(userInput, '\n');
59 stars = strcsplit(userInput, '\n');
60 stars = strcsplit(userInput, '\n');
61 stars = strcsplit(userInput, '\n');
62 stars = strcsplit(userInput, '\n');
63 stars = strcsplit(userInput, '\n');
64 stars = strcsplit(userInput, '\n');
65 stars = strcsplit(userInput, '\n');
66 stars = strcsplit(userInput, '\n');
67 stars = strcsplit(userInput, '\n');
68 stars = strcsplit(userInput, '\n');
69 stars = strcsplit(userInput, '\n');
70 stars = strcsplit(userInput, '\n');
71 stars = strcsplit(userInput, '\n');
72 stars = strcsplit(userInput, '\n');
73 stars = strcsplit(userInput, '\n');
74 stars = strcsplit(userInput, '\n');
75 stars = strcsplit(userInput, '\n');
76 stars = strcsplit(userInput, '\n');
77 stars = strcsplit(userInput, '\n');
78 stars = strcsplit(userInput, '\n');
79 stars = strcsplit(userInput, '\n');
80 stars = strcsplit(userInput, '\n');
81 stars = strcsplit(userInput, '\n');
82 stars = strcsplit(userInput, '\n');
83 stars = strcsplit(userInput, '\n');
84 stars = strcsplit(userInput, '\n');
85 stars = strcsplit(userInput, '\n');
86 stars = strcsplit(userInput, '\n');
87 stars = strcsplit(userInput, '\n');
88 stars = strcsplit(userInput, '\n');
89 stars = strcsplit(userInput, '\n');
90 stars = strcsplit(userInput, '\n');
91 stars = strcsplit(userInput, '\n');
92 stars = strcsplit(userInput, '\n');
93 stars = strcsplit(userInput, '\n');
94 stars = strcsplit(userInput, '\n');
95 stars = strcsplit(userInput, '\n');
96 stars = strcsplit(userInput, '\n');
97 stars = strcsplit(userInput, '\n');
98 stars = strcsplit(userInput, '\n');
99 stars = strcsplit(userInput, '\n');
100 stars = strcsplit(userInput, '\n');

```

The second compared byte is 0x61 (a).
 Concatenate 'a' as our second character for our input in the next run.
 Repeat the process until we get the flag.

```

L$ ./ctfd_plus
Welcome to CTFd+!
So far, we only have one challenge, which is one more than the number of databases we have

Very Doable Pwn - 500 points, 0 solves
Can you help me pwn this program?
#include <stdio.h>
int main(void) {
    puts("Bye!");
    return 0;
}

Enter the flag:
lactf{m4yb3_th3r3_1s_s0m3_m3r1t_t0_us1ng_4_db}
You got the flag! Unfortunately we don't exactly have a database to store the solve in...

```