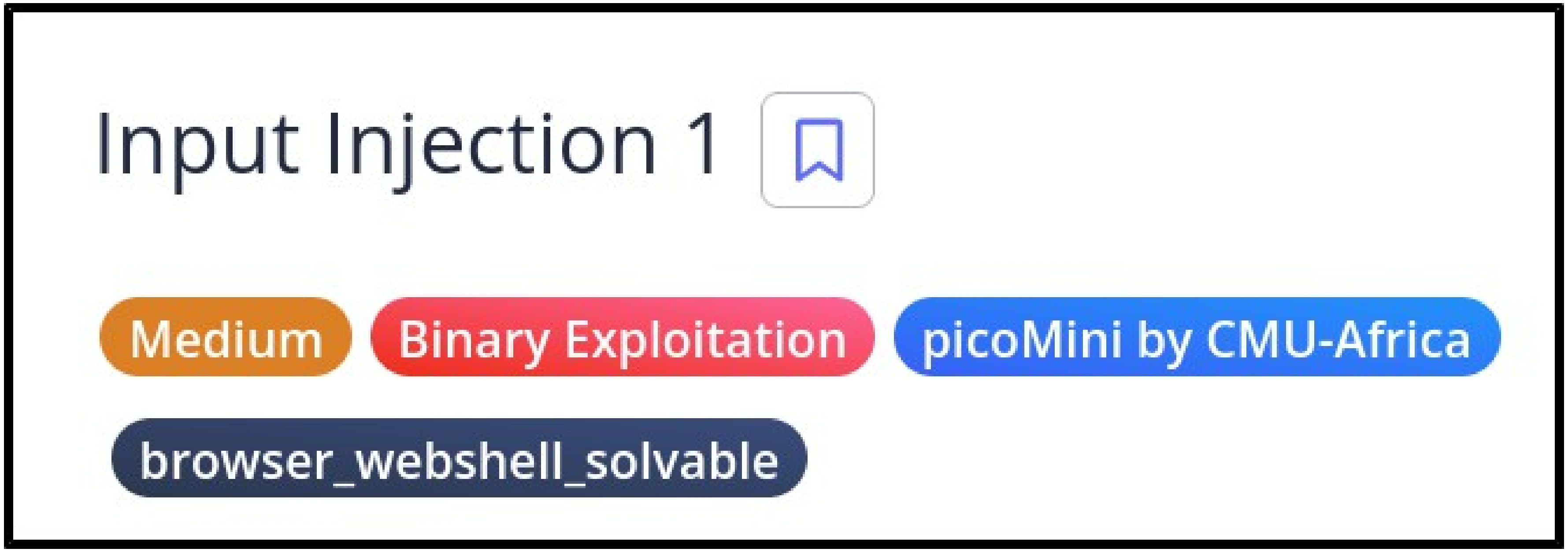
Pico Mini CMU Africa
Input Injection 1



Input Injection 1

Medium

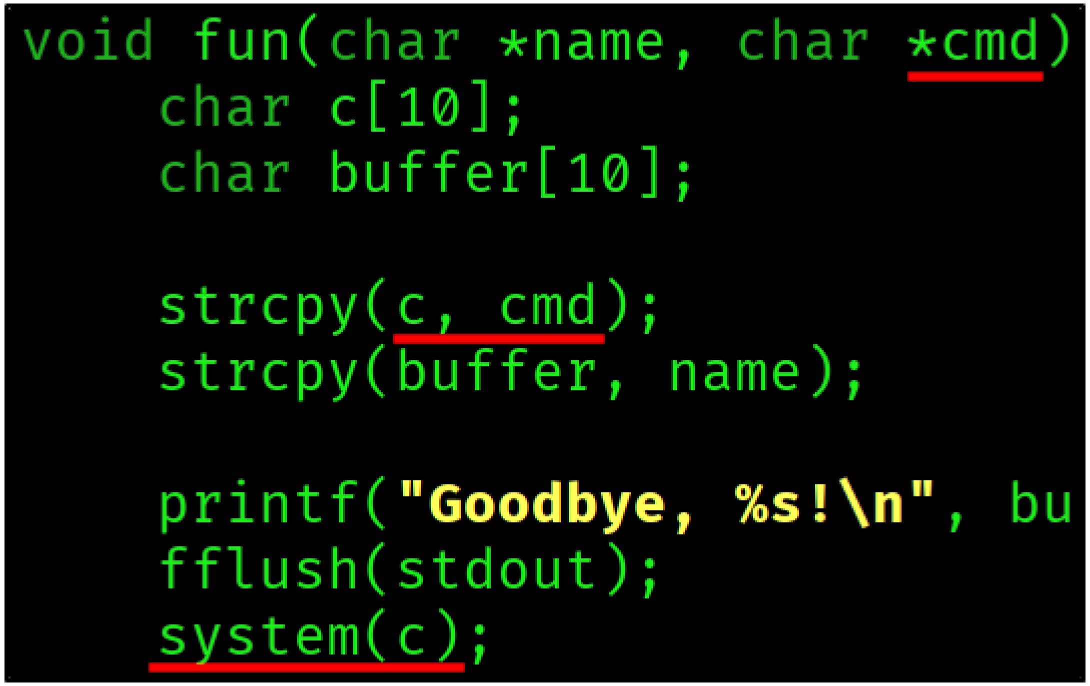
Binary Exploitation

picoMini by CMU-Africa

browser_webshell_solvable

In this challenge, we're tasked with abusing a binary running on a remote server

C System Function



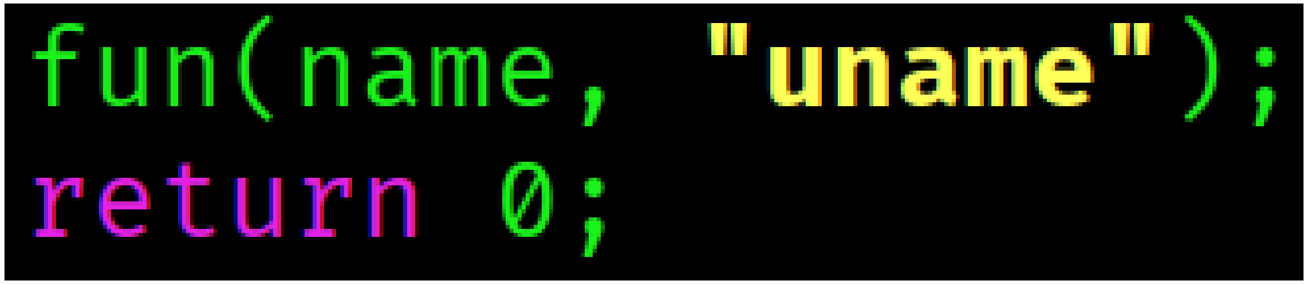
```
void fun(char *name, char *cmd)
    char c[10];
    char buffer[10];

    strcpy(c, cmd);
    strcpy(buffer, name);

    printf("Goodbye, %s!\n", bu
    fflush(stdout);
    system(c);
```

We see that the binary uses the `system` function with the `c` variable, which is copied from `cmd` variable. The C `system` function is used to run OS commands

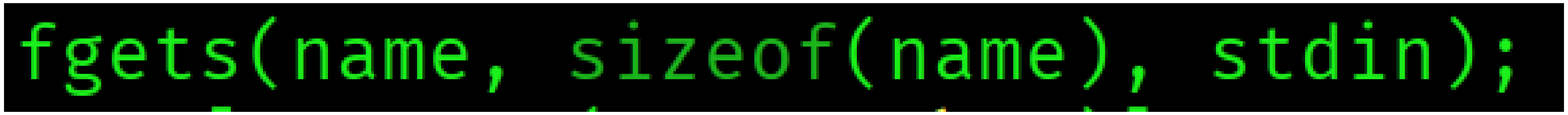
Buffer Variable Overwrite



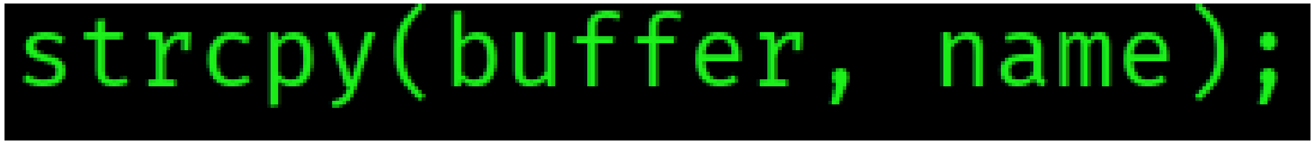
```
fun(name, "uname");
return 0;
```

In this challenge the goal is to force the binary to run arbitrary OS system commands. The source code lets us know that the binary runs the `uname` command by default

Memory Vulnerable Functions



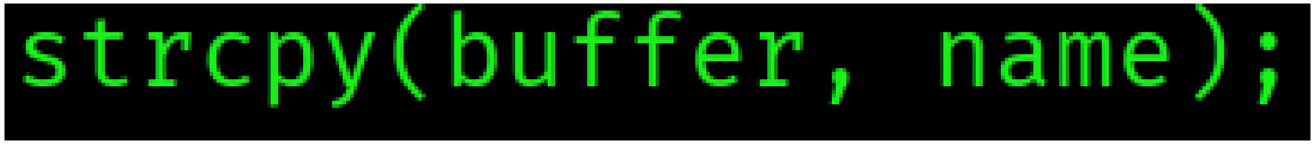
```
fgets(name, sizeof(name), stdin);
```



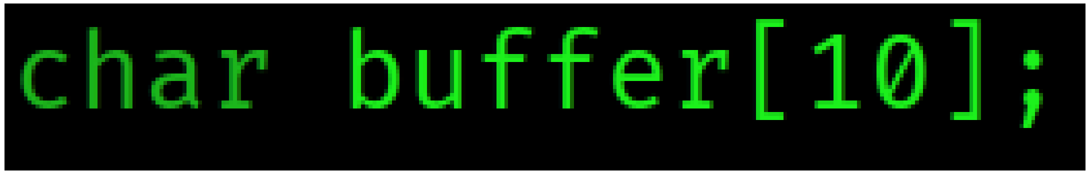
```
strcpy(buffer, name);
```

The binary takes in user input and saves it to the `name` variable, which in turn is copied to the `buffer` variable using the `strcpy` function, which is a memory-unsafe function in C

Memory Vulnerable Functions



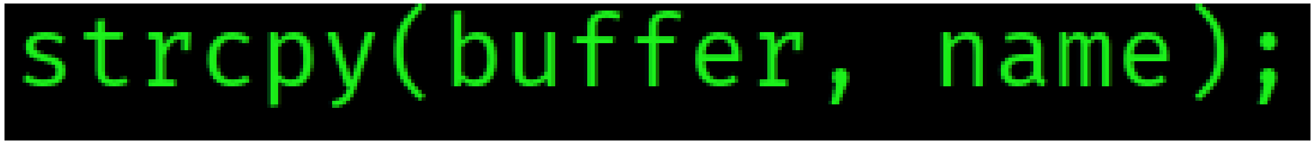
```
strcpy(buffer, name);
```



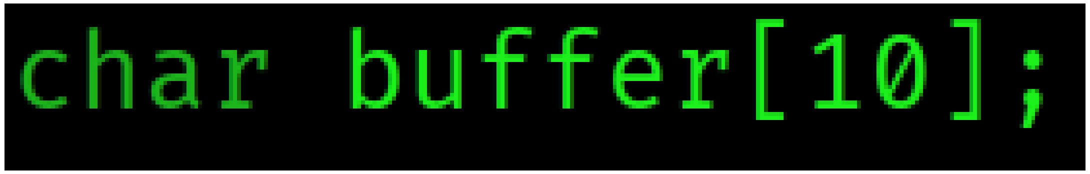
```
char buffer[10];
```

`strcpy` is memory-unsafe because it does not check the size of the memory buffer it is copying to, which can result in memory buffer overflow

Memory Vulnerable Functions



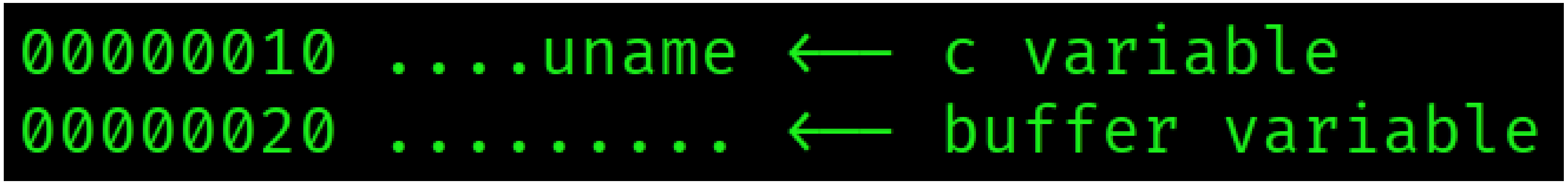
```
strcpy(buffer, name);
```



```
char buffer[10];
```

In this case, `name` is copied to the `buffer` variable, but its only been allocated ten bytes to store the `name` variable, so user input in excess of 10 characters will cause a buffer overflow when this `strcpy` function executes

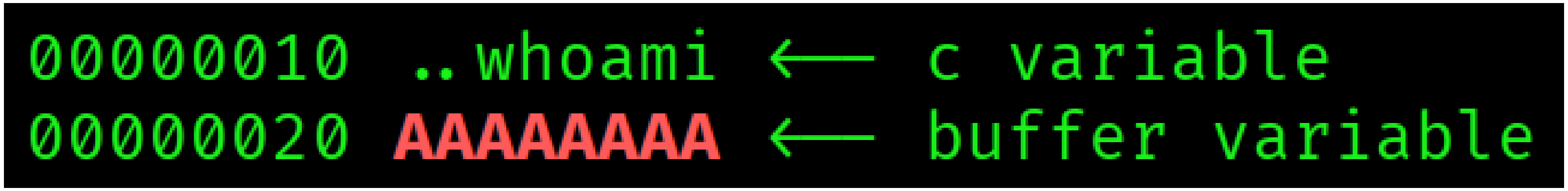
Stack Buffer Overflow



```
00000010 ....uname ← c variable
00000020 .........  ← buffer variable
```

As normal variables, both the `buffer` and the `c` variable (which is the system command) are initialized on the memory stack, and the c variable is initialized first

Stack Buffer Overflow



```
00000010 ..whoami ⟵ c variable
00000020 AAAAAAAA ⟵ buffer variable
```

So if the `buffer` variable is overflowed, it overflows into the `c` variable, we can run any system command we want through variable overwrite