

# Lab 5: Break the Barrier

## Brief

### It's Not Enough...

We've been warned that the following program is unsafe:

```
char s[16];
strcpy(src, s);
```

`strcpy` copies the string being pointed to by `src`, character by character, until it reaches an ending 0 (`\0`). This can be dangerous, as **no matter how long** `src` is, `strcpy` will try to copy all its characters into `s`. If `src` is shorter than 16 bytes, then everything can still be fine.

But what if it's not?

Well, `strcpy` knows nothing about the size of `s`. It just keeps writing data onwards. These data are then placed to places where they aren't supposed to appear — known as an **overflow**. The overflowed data can overwrite other memory places, other data, and even — code.

### Super User Do

Code runs in LC-3 has two permission levels: supervisor (aka. privileged) and user. The supervisor mode allows instructions to access any resource, including I/O devices, trap vectors, and all code running on the same machine. In contrast, user mode grants only limited access, where important things are protected from unwanted changes. In particular, user mode programs access I/O privileged resources via interrupts, or **syscalls**, which checks the data carefully and only deliver valid results from/to the user program.

Normally, an ISR will never run arbitrary code from the user program (like `JMP R0`) since it's unsafe. Most ISRs will just do simple things, like printing strings, reading keys, etc..

### The Weak Spot

We've mentioned `strcpy` before, by giving it a longer string, `strcpy` will potentially modify data which it was not supposed to touch. If the input is constructed carefully — with particularly constructed content and layout, then it can also change some code. If the code of a user program is changed, then it would function in another way, and so are ISRs. In the worst

scenerao, a user program can hijack the ISR, making it run privileged code, that were originally "locked" in the user mode.

## Intro

---

In LC3Tools, the `IN` trap prints a prompt, notifying you to input a character:

```
Input a character>
```

This prompt is stored somewhere in the priviledged memory (`x032c` to `x033f`), so the user program can't change it. I added another trap, allowing you to set this message:

- Triggered via `TRAP x30`.
- Copies the string being pointed to by the address in `R0`, writes it to the place that stores the prompt, and stops when it has copied a `\0`.

Such behavior is vulnerable, as right after `x033f` (that is, `x0340`) stays the ISR of `PUTSP`. If the provided message is too long, it will replace the original code of `PUTSP` with the content provided by the user. In this way, the user program can control what `PUTSP` does, like running another piece of the user code without leaving the supervisor mode (via `RTI`) first.

And let's see how this will happen.

## Goal

---

Launch an attack to the vulnerable "set prompt" ISR, let your code run in the supervisor mode, and run another "malicious" user program with privileged access.

- Remember, `TRAP x30` copies data from `R0` to `x032c`, until it reaches `\0`. Use such unsafe behavior wisely.
- The code of `PUTSP` starts from `x0340`.
- The "malicious" code that needs to be run in the supervisor mode starts from `x4000`. It must not be overwritten.