Pico Gym Exclusive: Local-Target

```
int num = 64;
```

```
if( num == 65 ){
  printf("You win!\n");
  fflush(stdout);
  // Open file
  fptr = fopen("flag.txt", "r");
```

This challenge is an example of a variable overwrite binary hacking challenge

Variable Overwrite Challenges

```
int num = 64;
```

```
if( num == 65 ){
  printf("You win!\n");
  fflush(stdout);
  // Open file
  fptr = fopen("flag.txt", "r");
```

In variable overwrite challenges, the goal is to overwrite a variable the program uses, but is not modified during regular program execution

Variable Overwrite Challenges

```
int num = 64;
```

```
if( num == 65 ){
  printf("You win!\n");
  fflush(stdout);
  // Open file
  fptr = fopen("flag.txt", "r");
```

In this binary, if the `num` variable is 65, then the contents of the `flag.txt` file are revealed

Buffer Overflow via GETS Function

```
char input[16];
```

```
gets(input);
```

The program uses the C `gets` function to record user input to the `input` buffer, and the `gets` function is a memory unsafe function, which can be used for buffer overflow attacks

Buffer Overflow via GETS Function

```
00000010 ....aa ⟵ other memory
00000020 aaaaaa ⟵ input buffer
```

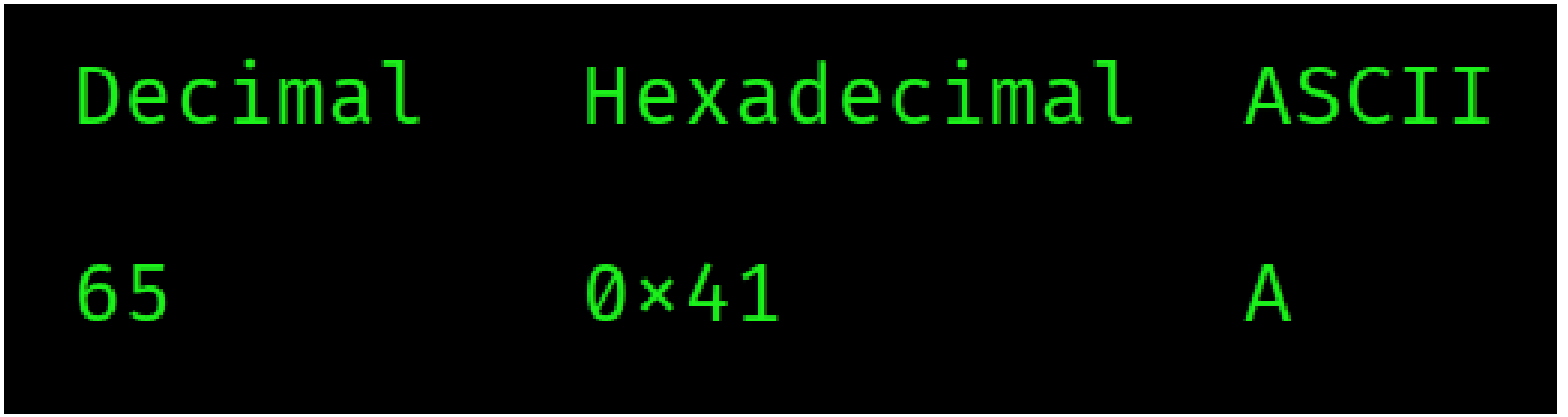
Buffer overflow causes data to overflow out of a memory buffer into other parts of the program memory, and in this case it can be used to overwrite the `num` variable, which is our goal

Buffer Overflow via GETS Function

```
char input[16];
```

The buffer for `input` is 16 bytes, so we need to send at least 17 bytes to overflow the memory buffer and overwrite the `num` variable

Decimal Number Storage in Memory



In program memory, decimal numbers are stored as bytes, which are represented by hexadecimal numbers, so if we want to put 65 into the `num` variable, we actually want to use the hex byte `0x41` or the ASCII character `A`