Experiment no.: 10

Title: Demonstrate buffer overflow attack.

Course Outcome: Build systems that are more secure against attacks.

Theory:

A buffer overflow happens when more data is written into a fixed-size buffer than it can store. Excess bytes can overwrite adjacent memory (other variables or control data). This experiment demonstrates the concept by safely simulating how extra input bytes could overwrite a nearby 32-bit variable (secret) without invoking undefined behavior — so it runs reliably on any compiler or online IDE.

Algorithm / Methodology:

1. Declare a small buffer (char buffer[8]) and an adjacent 32-bit variable secret initialized to 0xDEADBEEF.

2. Read user input safely into a large temporary array (using fgets) so the program never actually crashes.

3. Copy up to the buffer capacity into buffer with strncpy and NUL-terminate. This keeps the program safe and reproducible.

4. If the input length is greater than the buffer size, take the next up to 4 bytes from input and build a 32-bit little-endian value — this simulates the value that would overwrite secret in a real overflow.

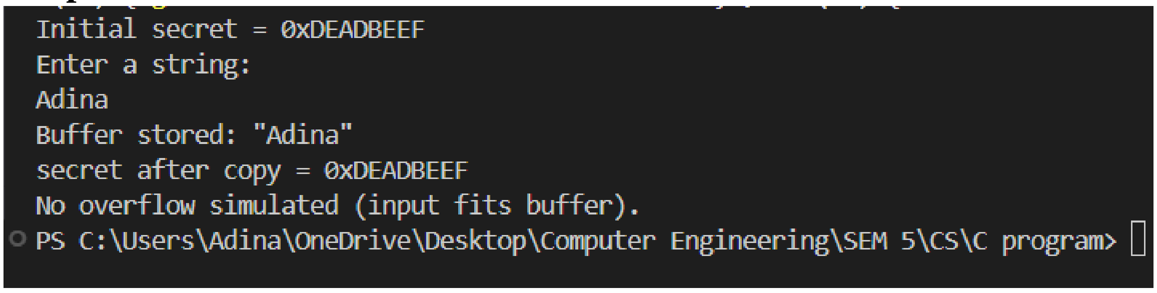
5. Print the stored buffer, the actual (unchanged) secret, and the simulated overwritten value (if any).

6. Explain the result and note that actual overflow behavior is platform-dependent and may be prevented by system protections.

Program:

```
#include <stdio.h>
#include <string.h>
#include <stdint.h>

int main(void) {
    char buffer[8];
    uint32_t secret = 0xDEADBEEF;
    char input[200];

    printf("Initial secret = 0x%08X\n", secret);
    printf("Enter a string:\n");

    if (fgets(input, sizeof input, stdin) == NULL) return 0;
    input[strcspn(input, "\n")] = '\0';

    strncpy(buffer, input, sizeof buffer - 1);
    buffer[sizeof buffer - 1] = '\0';

    printf("Buffer stored: \"%s\"\n", buffer);
    printf("secret after copy = 0x%08X\n", secret);
    size_t inlen = strlen(input);
    if (inlen > sizeof buffer) {
        unsigned char *b = (unsigned char *)(input + sizeof buffer);
        uint32_t simulated = 0;
        for (size_t i = 0; i < 4 && i < inlen - sizeof buffer; ++i)
            simulated |= ((uint32_t)b[i]) << (8 * i);
        printf("Simulated overwritten secret = 0x%08X\n", simulated);
    } else {
        printf("No overflow simulated (input fits buffer).\n");
    }
    return 0;
}
```

Output:



```
Initial secret = 0xDEADBEEF
Enter a string:
Adina
Buffer stored: "Adina"
secret after copy = 0xDEADBEEF
No overflow simulated (input fits buffer).
PS C:\Users\Adina\OneDrive\Desktop\Computer Engineering\SEM 5\CS\C program>
```

Conclusion:

I have successfully demonstrated buffer overflow attack , written in C program.