Software vulnerabilities I v2

Note Title

(1) Background

- Most interesting computer programs receive input

  e.g. - from a user (keyboard or mouse clicks)
  - from a file, such as reading "my-homework.docx"
  - from the network, such as from a web server like www.dickinson.edu

- If the input contains unexpected content, a program may behave strangely.

- Usually, unexpected input causes the program to crash. This means that it stops executing, usually displaying some kind of error message

  [demo: examples in Python and C.]

- If, however, the unexpected input is specially crafted, it may cause the program to continue executing, but do something different to its original purpose

(2) Example: Injecting python code into python.

The python programming language has a function called 'eval'. When given a string as input, this function evaluates the string as if it were part of the current program.

e.g. eval("5+7") gives 12

The calculator.py program on the resources page accepts input from the user. It expects a mathematical expression to be input. It evaluates the expression using 'eval', and prints out the result for the user

[demo: - read through program
- show simple arithmetic
- show modulo arithmetic with large integers — good for crypto calculations!]

Note that it is easy to crash this program by entering anything that isn't legal Python code

[demo: show crash]

But, this program is too powerful!!
It will evaluate any legal python expression.
In particular, we could do something evil,

such as delete a file using the python 'remove' function.

[Demo: - remove a file
- remove the program itself (!!)]

Note that if you are running calculator.py in your own account, you would only be able to delete your own files (there are exceptions, but this gives the basic idea).

The real danger comes if calculator.py is running as part of a web service. Then a remote user could delete files on the web server that they do not own !!

And, by the way, calculator.py is vulnerable to DoS attacks too

[Demo: - use exponentiation that takes a long time to compute, e.g. 3**3**3**3]

↑
in python, exponentiation is '**' not '^'.

(3) Buffer overflow in a C or C++ program

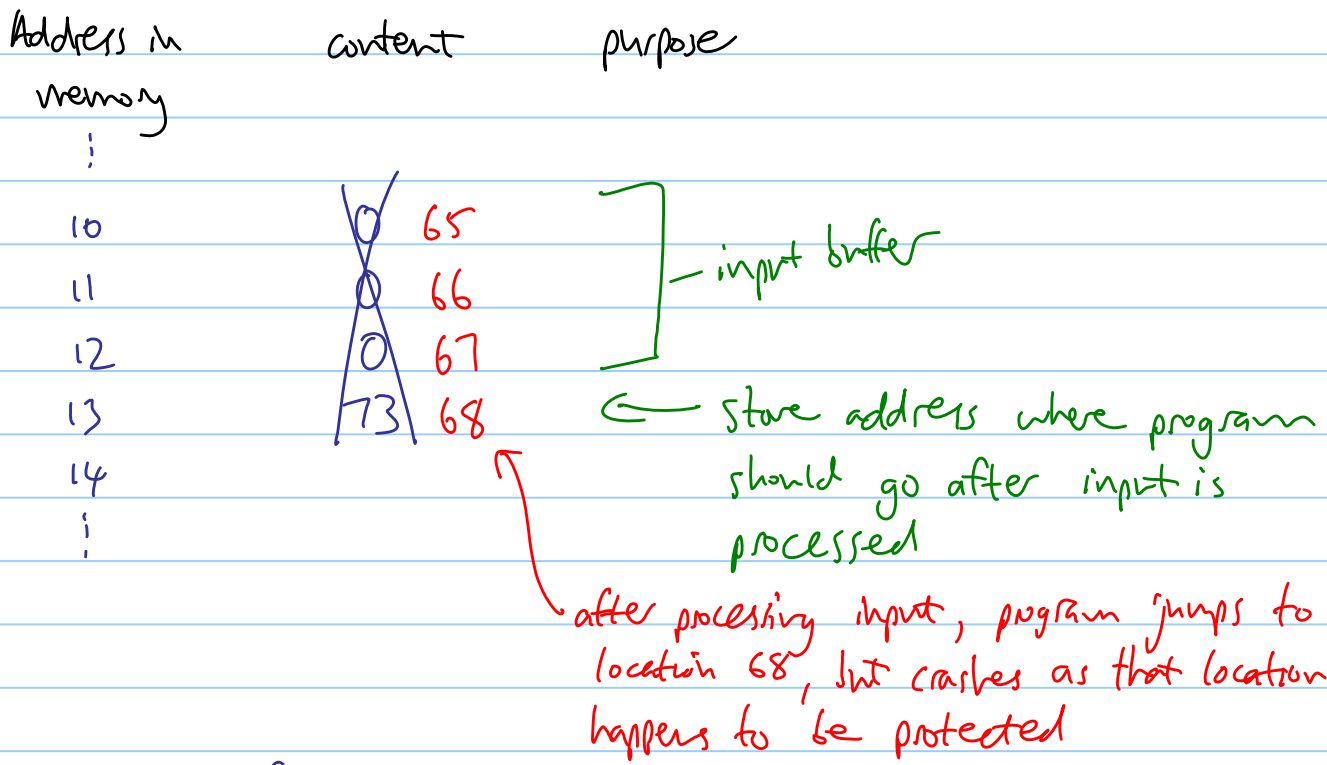
Sometimes, a program pre-allocates a fixed amount of space (called a buffer) to store input.

If the input is longer than the buffer, the input will overflow the buffer, replacing some of the program's own information. Usually, this will just cause a crash, but a carefully-crafted input can cause the program to execute the user's own code.

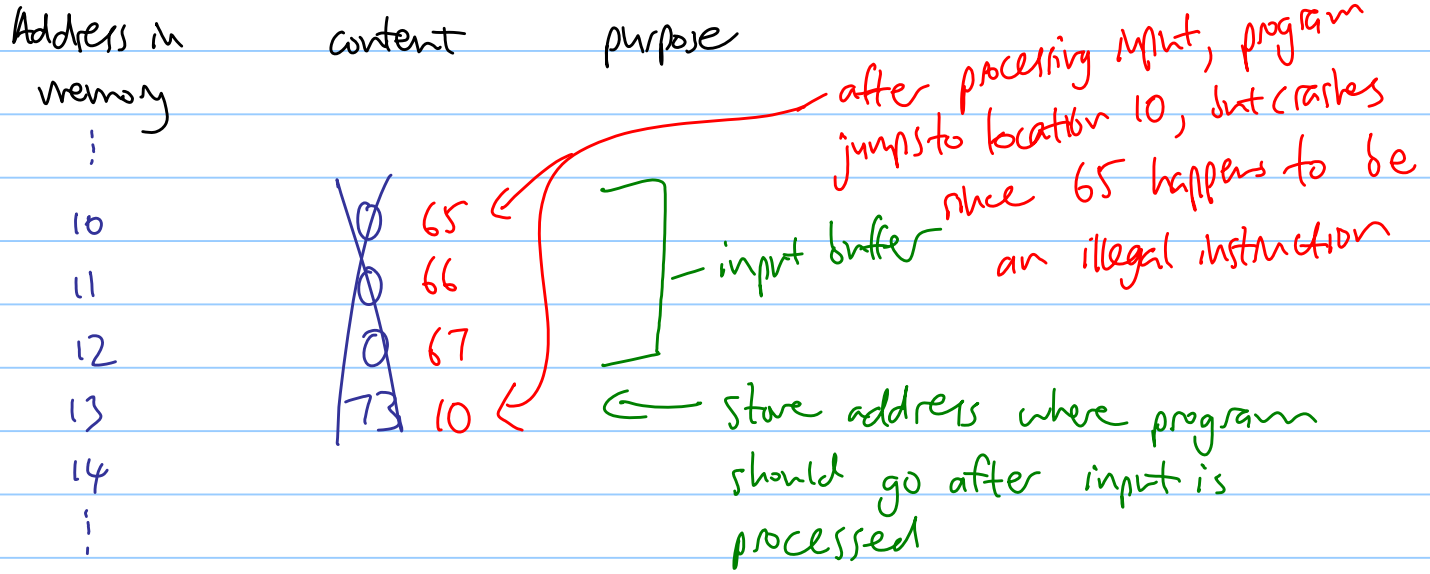
Simplified example:

Address in memory	content	purpose
⋮		
10	0	input buffer
11	0	
12	0	
13	73	← store address where program should go after input is processed
14		
⋮		

(a) After overflowing input, jumping to illegal location:
[Input is 'ABCD' or in numbers, 65 66 67 68]

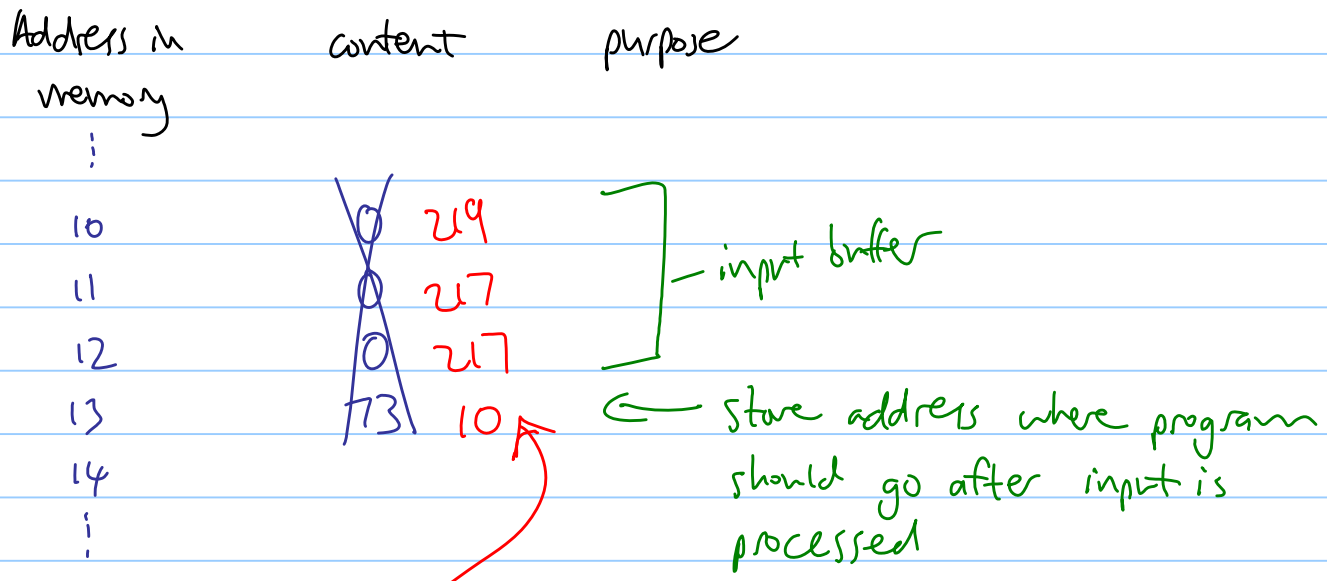


(b) After overflowing input, jumping to start of input!!:
(but still crashing)
[Input, in numbers, is 65 66 67 10]



(c) After overflowing input, jump to start of buffer, and do something evil like delete a file:

(input, in numbers, is 249 217 217 10)



Note: This is a completely fictitious example. In reality, a sequence of numbers much more complex than 249 217 217 would be required.

[Demo: Show some of this in action, if time].
Use greeting.c with input aaaabbbbcccc, and longer.