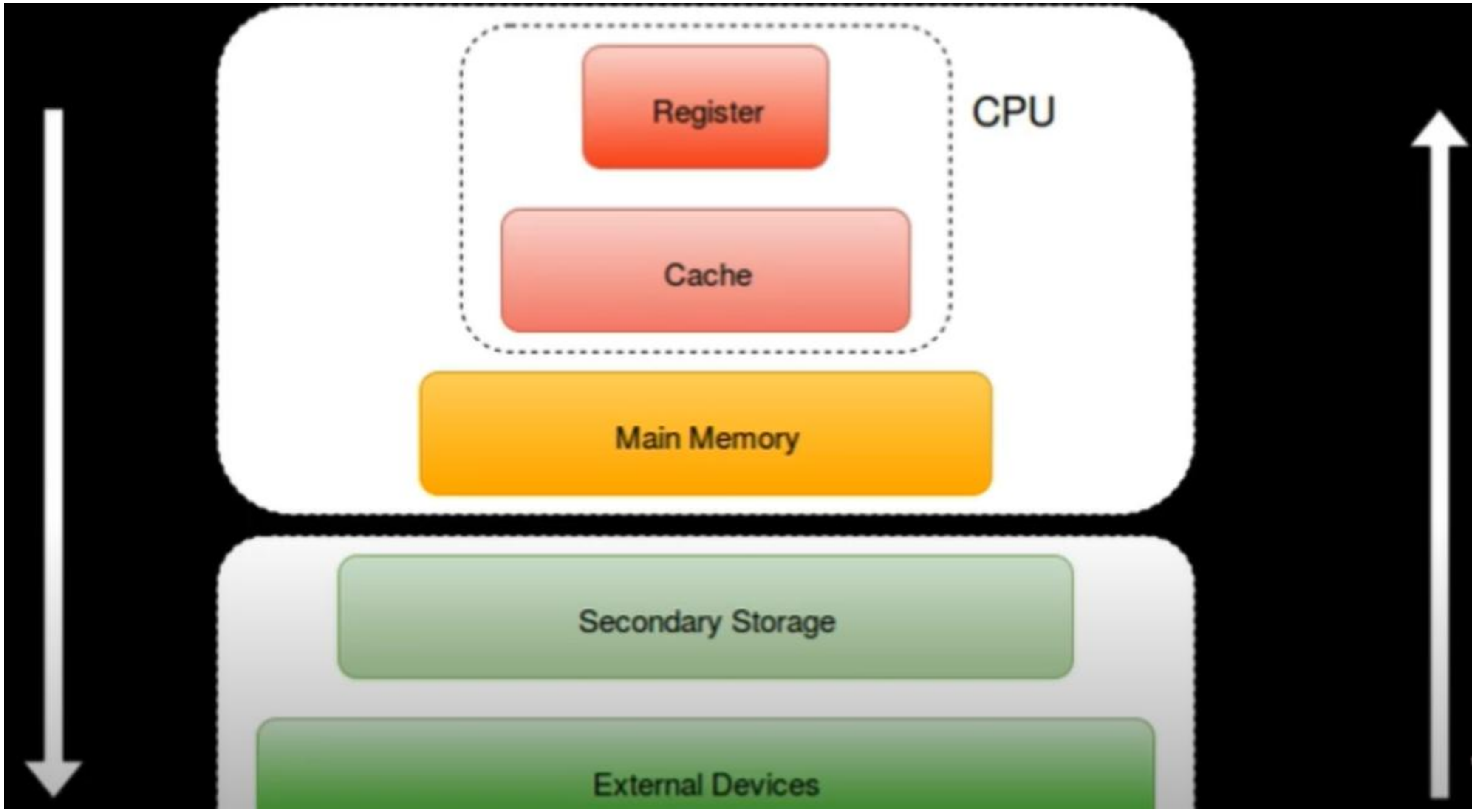
Buffer , Integer Over Flow & Sandboxing

TCS 591 : UNIT 1:Session 2

Computer Memory



Computer Memory

Memory Allocation Process

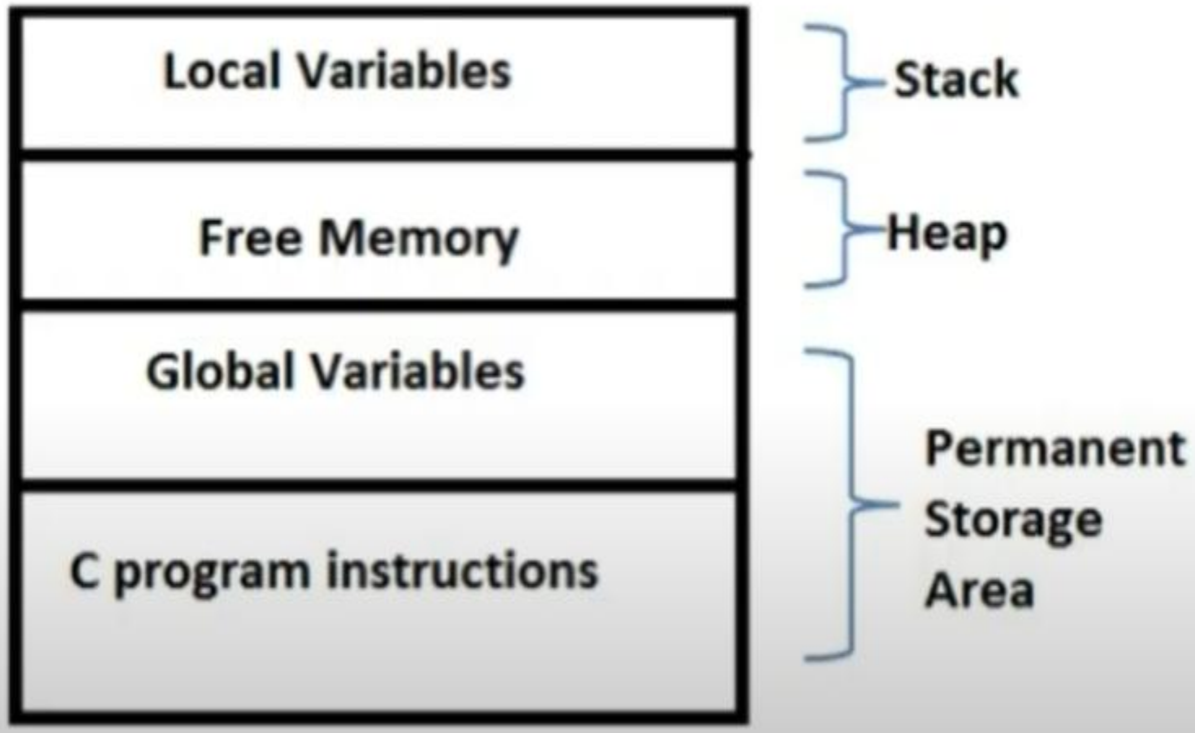
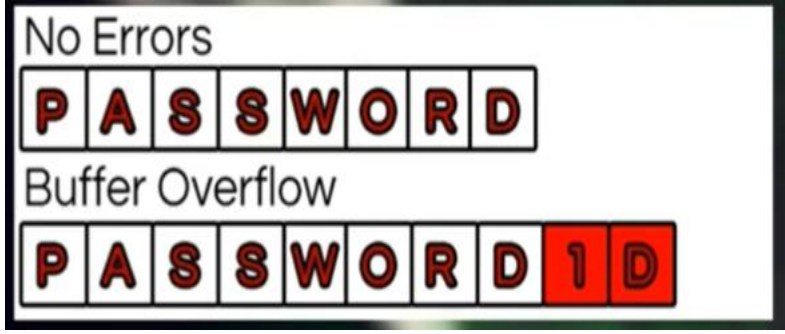
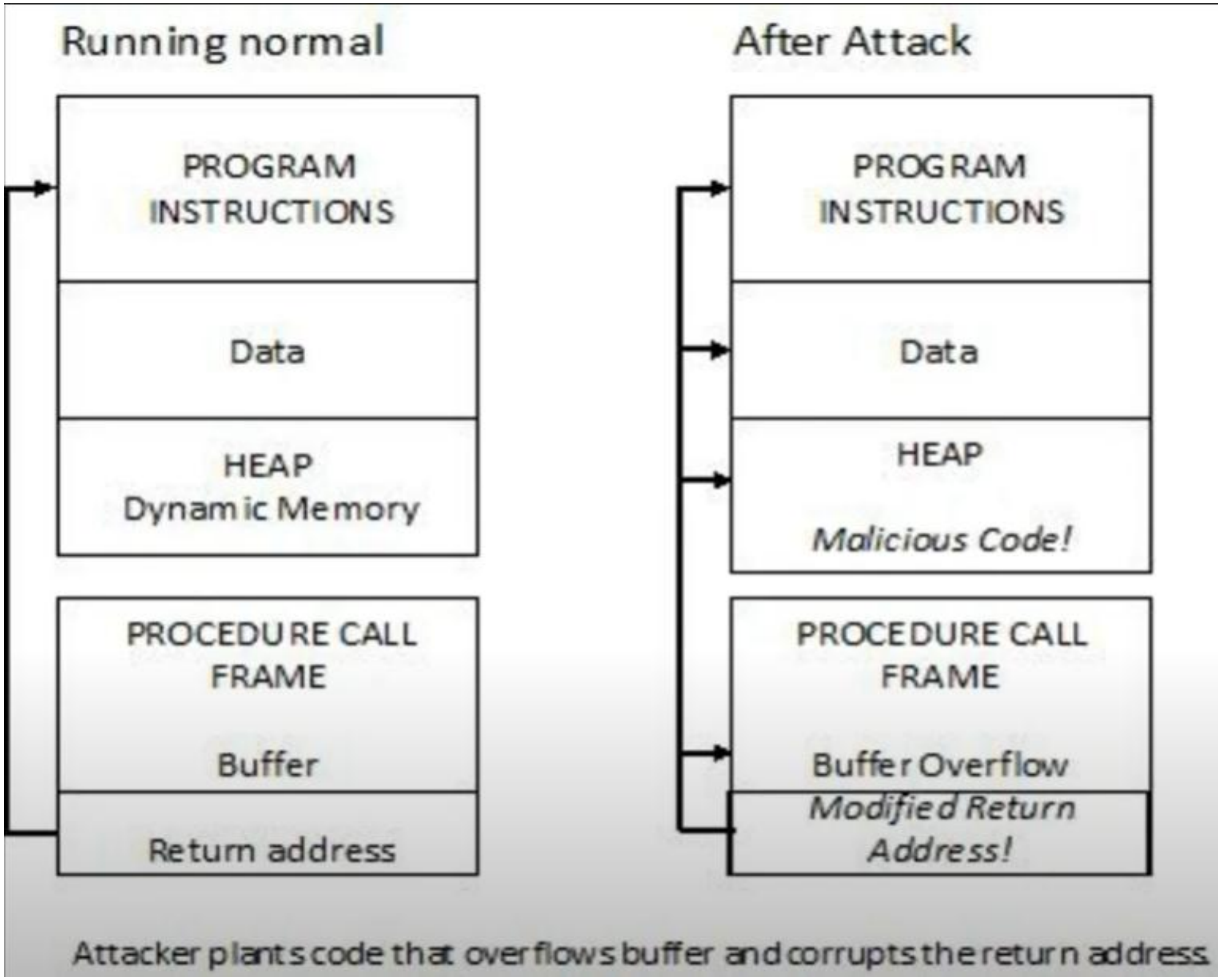


Fig: Storage of a C program

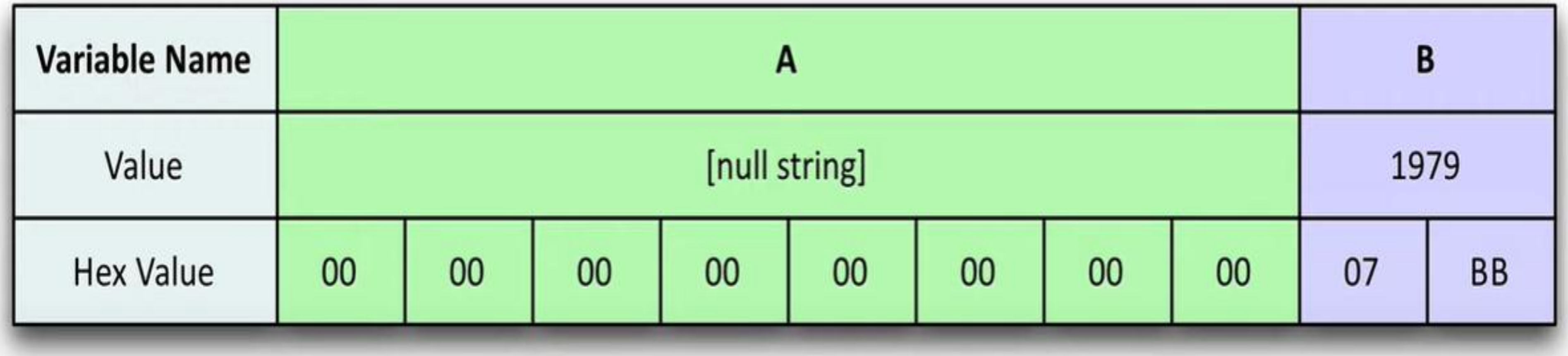
Buffer Overflows



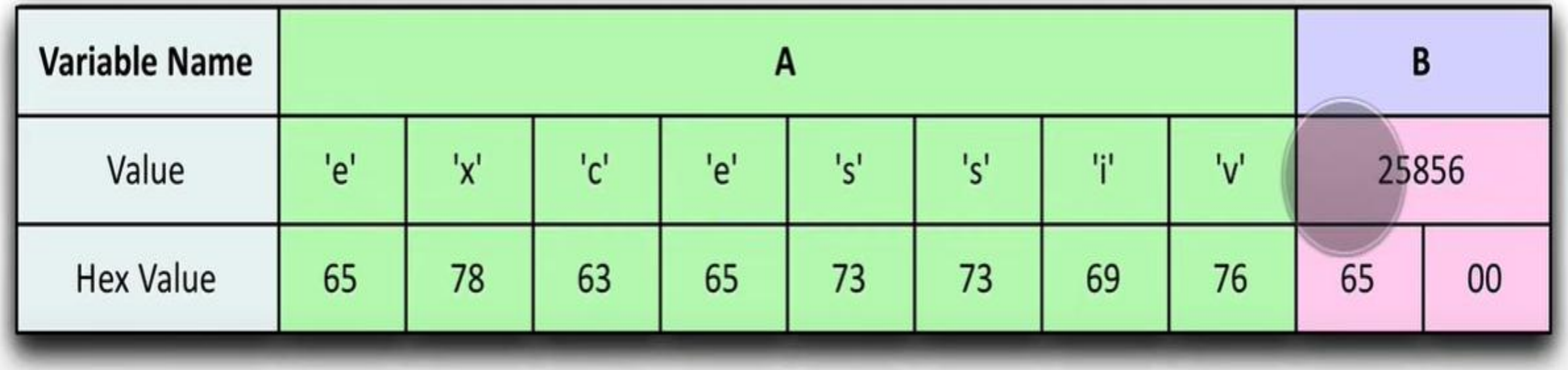
- Overwriting a buffer of memory
  - Spills over into other memory areas
- Developers need to **perform bounds checking**
  - The attacker spend a lot of time looking for openings (**error in program**)
- **Not a simple** exploit
  - Takes time to avoid crashing things(one must have administrator access for this)



Buffer Overflow

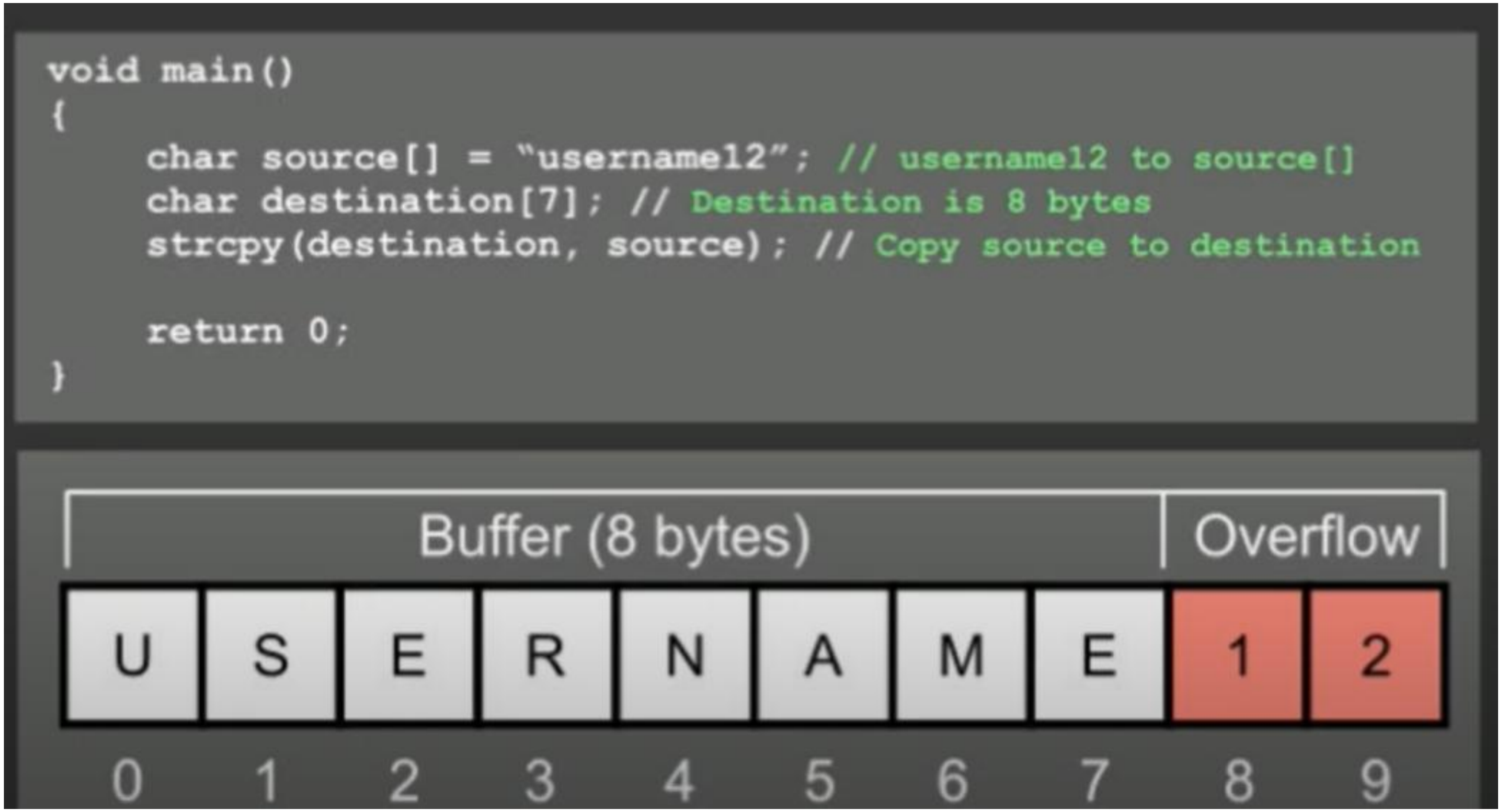


Variable Name	A								B	
Value	[null string]								1979	
Hex Value	00	00	00	00	00	00	00	00	07	BB



Variable Name	A								B	
Value	'e'	'x'	'c'	'e'	's'	's'	'i'	'v'	25856	
Hex Value	65	78	63	65	73	73	69	76	65	00

Buffer Overflow Example



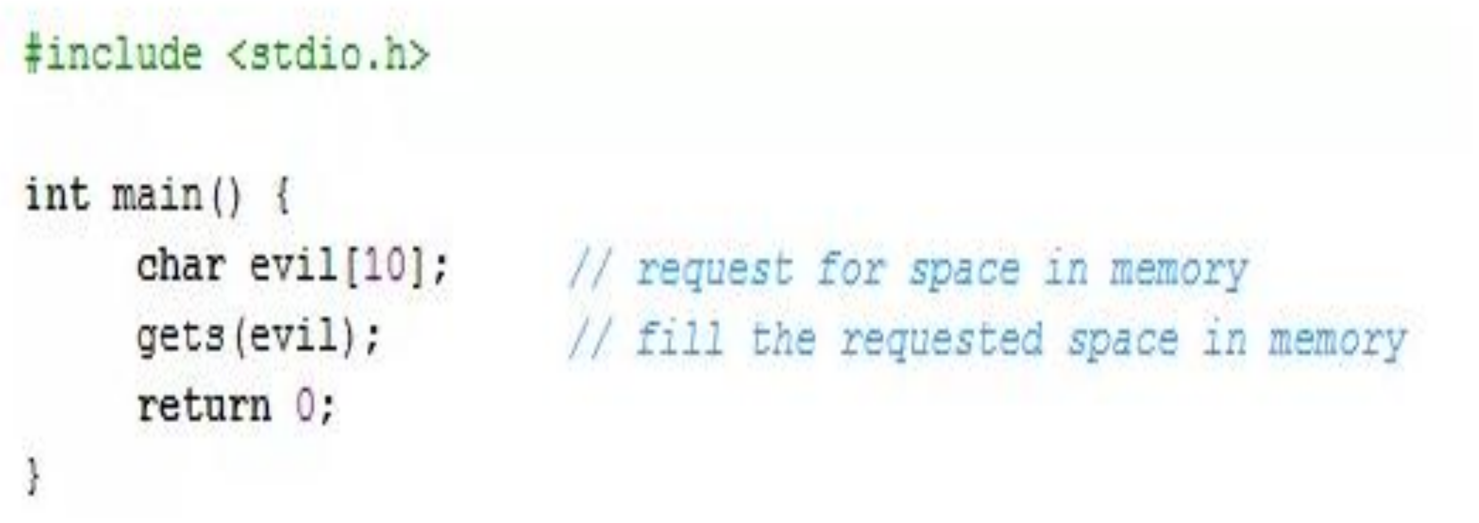
```
void main()
{
     char source[] = "username12"; // username12 to source[]
     char destination[7]; // Destination is 8 bytes
     strcpy(destination, source); // Copy source to destination

     return 0;
}
```

Buffer overflow

- Buffer overflow is a result of insufficient boundary checks when inserting data to a buffer. If we insert more data than the buffer holds, a buffer overflow occurs. It is easy, it is simple, but it is dangerous and the results from it might be severe.

Example of a vulnerable function



```
#include <stdio.h>

int main() {
     char evil[10];     // request for space in memory
     gets(evil);        // fill the requested space in memory
     return 0;
}
```

Integer Overflow

- Integer
  - A positive or negative whole number(1,2,3,-5,-6)
  - No Fractions/decimal value (2.1,-1.1)
- Usually has a **Fixed boundary**
  - A low end and high end range(say -30 to +30)
- Vulnerable software may allow an integer to go out of bounds
  - i.e wrap the integer value from positive to negative (after 29, 30 it **will not go to 31 but it will go -30**)
- This integer may allocate a memory location for a buffer
  - The buffer will now be too small, and a buffer overflow may occur

What is a Sandbox ?

In computer security, a sandbox is a security mechanism for separating running programs, usually in order to minimize system failures or software vulnerabilities from spreading.

What is a Sandbox ?

In general, a sandbox is an isolated computing environment in which a program or file can be executed without affecting the application in which it runs.

Sandboxes are used by software developers to test new programming code.

Sandboxing is frequently used to test unverified programs that may contain a virus or other malicious code, without allowing the software to harm the host device.

Importance of Sandboxes

- Cybersecurity professionals use sandboxes to test potentially malicious software. Without sandboxing, an application or other system process could have unlimited access to all the user data and system resources on a network.

- Sandboxing protects an organization's critical infrastructure from suspicious code because it runs in a separate system. It also allows IT to test malicious code in an isolated testing environment to understand how it works within a system as well as more rapidly detect similar malware attacks.