Why are we still vulnerable to buffer overflows?

Why code written in assembly code or C are subject to buffer overflow attacks?

➡ Because C has primitives to manipulate the memory directly (pointers ect ...)

If other programming languages are “memory safe”, why are we not using them instead?

- Because C and assembly code are used when a program requires high performances (audio, graphics, calculus …) or when dealing with hardware directly (OS, drivers ….)

Why are we still vulnerable to buffer overflows?

Why code written in assembly code or C are subject to buffer overflow attacks?

➡ Because C has primitives to manipulate the memory directly (pointers ect ...)

If other programming languages are "memory safe", why are we not using them instead?

- Because C and assembly code are used when a program requires high performances (audio, graphics, calculus …) or when dealing with hardware directly (OS, drivers ….)

Notable Attacks

- **Heartbleed** (CVE-2014-0160)
  Bounds check failure in OpenSSL's Heartbeat extension revealing private keys

- **Ghost Vulnerability** (CVE-2015-0235)
  Buffer overflow in glibc **`gethostbyname()`** allowing remote code execution through DNS lookups

- **EternalBlue** (CVE-2017-0144)
  Buffer overflow that allows remote execution code in Samba Windows Service resulting in malware: *WannaCry* and *NotPetya*