Fady Youssef
3/3/2025
OS Security

---

1. C adds extra code and data that aren't needed for shellcode. This extra stuff can include unwanted characters like null bytes, which can mess up the shellcode. Shellcode needs to be small and clean, which is why it's written in assembly language.

2. Zeros can act like end-of-string markers in many programming languages. If a zero is encountered, it can stop the shellcode from running completely, causing it to fail.

3. Shellcode usually calls the execve system call to run the /bin/sh program. This opens a shell, letting the attacker run any commands they want.

4. **EAX**: System call number.
   EBX: Pointer to the program's filename.
   ECX: Pointer to the arguments array.
   EDX: Pointer to the environment variables array.

5. **Inline Assembly**: Writing shellcode directly in assembly language and embedding it in a C program.
   Hexadecimal Representation: Writing shellcode in assembly, converting it to machine code, and then representing it as hexadecimal values.

6. **XOR Encoding**: Using the XOR instruction to avoid null bytes by encoding and decoding the shellcode.
   Stack-based Approach: Pushing data onto the stack during execution to avoid having null bytes directly in the shellcode.