

Canaries

- The compiler modifies every function's prologue and epilogue regions to place and check a value (a.k.a a canary) on the stack
- When a buffer overflows, the canary is overwritten. The program detects it before the function returns and an exception is raised
- Different types:
 - random canaries
 - xor canaries
- Disabling Canary protection on Linux

```
$ gcc ... -fno-stack-protector
```
- Bypassing canary protection : *Structured Exception Handling (SEH)* exploit overwrite the existing exception handler structure in the stack to point to your own code

DEP/NX - Non Executable Stack

- The program marks important structures in memory as non-executable
- The program generates an hardware-level exception if you try to execute those memory regions
- This makes normal stack buffer overflows where you set eip to esp+offset and immediately run your shellcode impossible
- Disabling NX protection on Linux
 \$ gcc ... -z execstack
- Bypassing NX protection :
 - *Return-to-lib-c*
 return to a subroutine of the lib C that is already present in the process' executable memory
 - *Return-Oriented-Programming (ROP)*
 use instruction pieces of the existing program (called "gadgets") and chain them together to weave the exploit