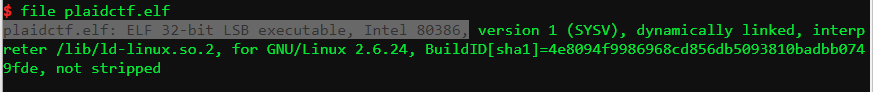
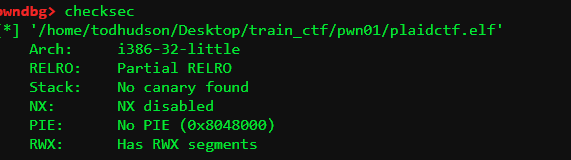
Plaidctf

Mở file check



Ok 32-bit

Mở checksec thử



Nhìn có vẻ không có gì cản trở

Tiếp theo vào code xem

0x0804855c <+21>: mov DWORD PTR [esp+0x8],eax

0x08048560 <+25>: mov DWORD PTR [esp+0x4],0x400

0x08048568 <+33>: mov DWORD PTR [esp],0x804a080

0x0804856f <+40>: call 0x80483b0 <fgets@plt>

Hàm main đẩy tham số vào, sau đó fgets, cho mình nhập vào buff, tối đa là 0x400 bytes

Sau đó nó gọi hàm echo

0x08048527 <+6>: call 0x80484fd <make\_response>

0x0804852c <+11>: mov DWORD PTR [esp],0x804a480

0x08048533 <+18>: call 0x80483c0 <puts@plt>

Trong hàm echo thì nó gọi hàm make\_response

0x08048503 <+6>: mov DWORD PTR [esp+0x8],0x804a080

0x0804850b <+14>: mov DWORD PTR [esp+0x4],0x400

0x08048513 <+22>: mov DWORD PTR [esp],0x804a480

0x0804851a <+29>: call 0x80483f0 <snprintf@plt>

Ta thấy có dòng này, nó gọi hàm snprintf, đẩy 3 tham số vào, debug lên thì ko có format string, chỗ này khả năng có thể khai thác

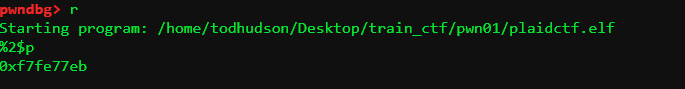
Sau đó ret về echo để puts ra màn hình

0x0804852c <+11>: mov DWORD PTR [esp],0x804a480

0x08048533 <+18>: call 0x80483c0 <puts@plt>

Sau khi hàm echo kết thúc thì nó ret về main và cho mình nhập tiếp, vậy đây là vòng lặp while, nhập vào rồi in ra, đại loại vậy

Mình thử exploit leak stack xem sao

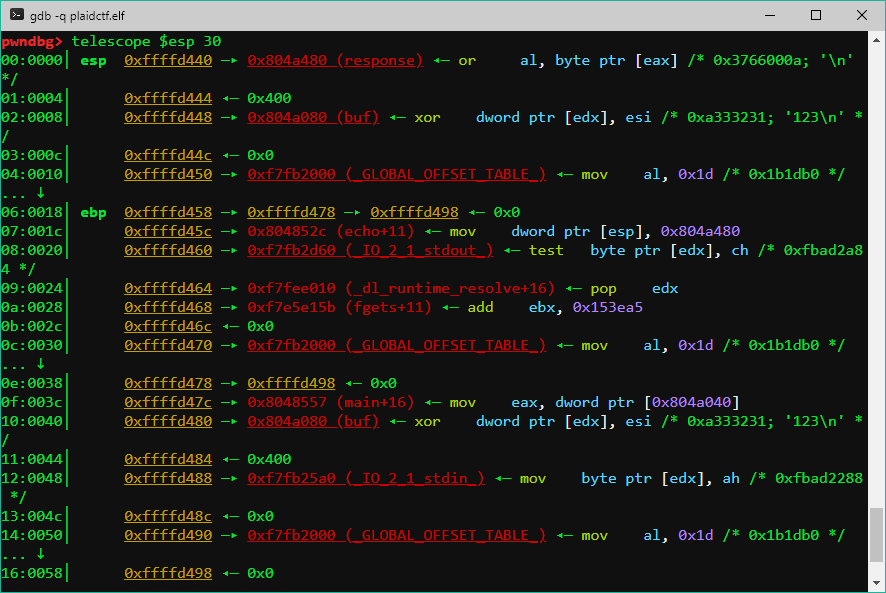


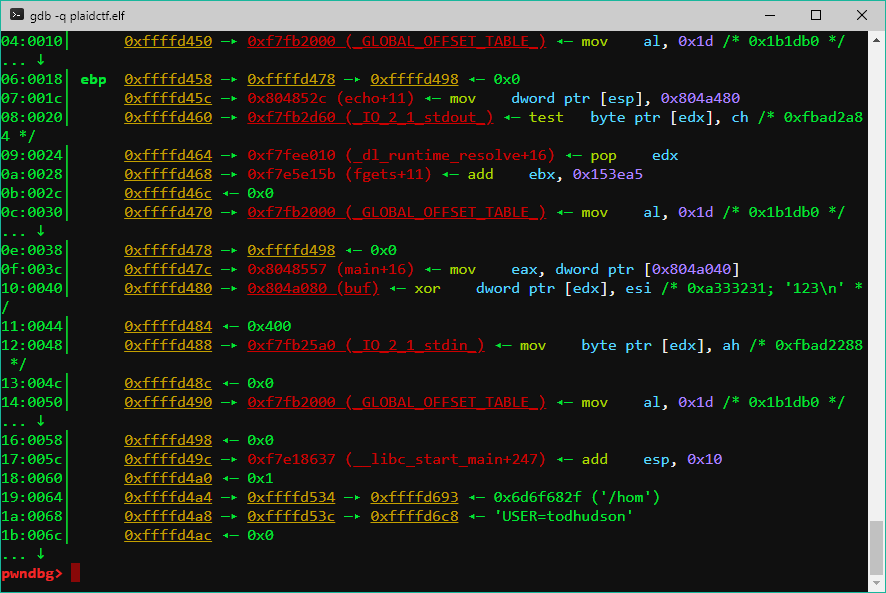
Ok ngon lành rồi, vậy là mình sẽ khai thác ở hàm snprintf, ở đây mình có 3 hướng:

1 là quăng shellcode vào, vì NX disable nên ngon lành rồi

2 là ret2libc, leak stack ra rồi ghi system vào, sau đó ROP để nhảy về system

3 cũng ret2libc, nhưng không ROP nữa mà sẽ nhảy ra khỏi while xuống ret main





Trước khi call snprintf thì mình check stack

19:0064│ 0xffffd4a4 —▸ 0xffffd534 —▸ 0xffffd693 ◂— 0x6d6f682f ('/hom')

1a:0068│ 0xffffd4a8 —▸ 0xffffd53c —▸ 0xffffd6c8 ◂— 'USER=todhudson'

Chỗ này ngon nè, mình sẽ dùng nó để ghi value vào stack

Hướng làm: leak memory stack để tính khoảng cách system, ret main, và binsh

Leak cả stack chứa fgets để tính libc , muốn ghi value thì phải có địa chỉ stack trong stack nên

Mình sẽ ghi địa chỉ stack vào 2 chỗ này,1 stack chứa 2 byte cao, 1 stack chứa 2 byte thấp

19:0064│ 0xffffd4a4 —▸ 0xffffd534 —▸ 0xffffd693 ◂— 0x6d6f682f ('/hom')

1a:0068│ 0xffffd4a8 —▸ 0xffffd53c —▸ 0xffffd6c8 ◂— 'USER=todhudson'

Rồi ghi value vào stack đã được ghi ở trên

3d:00f4│ 0xffffd534 —▸ 0xffffd693 ◂— 0x6d6f682f ('/hom')

3e:00f8│ 0xffffd538 ◂— 0x0

3f:00fc│ 0xffffd53c —▸ 0xffffd6c8 ◂— 'USER=todhudson'

Cứ thế ta sẽ ghi system binsh vào nơi cần ghi

Vì không giới hạn số lần nhập và buff cũng đủ lớn nên thoải mái