picoCTF Forky

Points: 500.
Description: In this program, identify the last integer value that is passed as parameter to the function doNothing().
Link: https://play.picoctf.org/practice/challenge/24?category=3&page=2

Basic Commands

drew@ubuntu:~/Desktop$ file vuln
vuln: ELF 32-bit LSB shared object, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=836c8d5ecaad6d64f4a358cf73d060d0c5050e87, not stripped

drew@ubuntu:~/Desktop$ strings vuln
/lib/ld-linux.so.2
fork
mmap
GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
vuln.c
main
doNothing
…..

Some interesting strings, but nothing that immediately pops out at me

First Execution

drew@ubuntu:~/Desktop$ ./vuln
drew@ubuntu:~/Desktop$

Looks like nothing happened lets look at the source code in Ghidra

Ghidra Analysis

```
/* WARNING: Function: __x86.get_pc_thunk.bx replaced

undefined4 main(undefined1 param_1)

{
  int *piVar1;

  piVar1 = (int *)mmap((void *)0x0,4,3,0x21,-1,0);
  *piVar1 = 1000000000;
  fork();
  fork();
  fork();
  fork();
  *piVar1 = *piVar1 + 0x499602d2;
  doNothing(*piVar1);
  return 0;
}
```

In the description of the challenge it states: “identify the last integer value that is passed as parameter to the function doNothing().” It looks like we need to find what piVar1 will return.

Radare2

```
[X] Disassembly (pd)
            0x565ff571      89e5            mov ebp, esp
            0x565ff573      53              push ebx
            0x565ff574      51              push ecx
            0x565ff575      83ec10          sub esp, 0x10
            0x565ff578      e8d3feffff      call sym.__x86.get_pc_thunk.bx ;[1]
            0x565ff57d      81c3571a0000    add ebx, 0x1a57
            0x565ff583      c745ec030000.   mov dword [var_14h], 3
            0x565ff58a      c745f0210000.   mov dword [var_10h], 0x21    ; '!' ; 33
            0x565ff591      83ec08          sub esp, 8
            0x565ff594      6a00            push 0
            0x565ff596      6aff            push 0xffffffffffffffff
            0x565ff598      ff75f0          push dword [var_10h]
            0x565ff59b      ff75ec          push dword [var_14h]
            0x565ff59e      6a04            push 4                      ; 4
            0x565ff5a0      6a00            push 0
            0x565ff5a2      e829feffff      call sym.imp.mmap           ;[2] ; void*mmap(void*ad
            0x565ff5a7      83c420          add esp, 0x20
            0x565ff5aa      8945f4          mov dword [var_ch], eax
            0x565ff5ad      8b45f4          mov eax, dword [var_ch]
            0x565ff5b0      c70000ca9a3b    mov dword [eax], 0x3b9aca00    ; [0x3b9aca00:4]=-1
            0x565ff5b6      e835feffff      call sym.imp.fork           ;[3]
            0x565ff5bb      e830feffff      call sym.imp.fork           ;[3]
            0x565ff5c0      e82bfeffff      call sym.imp.fork           ;[3]
            0x565ff5c5      e826feffff      call sym.imp.fork           ;[3]
            0x565ff5ca      8b45f4          mov eax, dword [var_ch]
            0x565ff5cd      8b00            mov eax, dword [eax]
            0x565ff5cf      8d90d2029649    lea edx, [eax + 0x499602d2]
            0x565ff5d5      8b45f4          mov eax, dword [var_ch]
            0x565ff5d8      8910            mov dword [eax], edx
            0x565ff5da      8b45f4          mov eax, dword [var_ch]
            0x565ff5dd      8b00            mov eax, dword [eax]
            0x565ff5df      83ec0c          sub esp, 0xc
            0x565ff5e2      50              push eax
            0x565ff5e3      e865ffffff      call sym.doNothing          ;[4]
            0x565ff5e8      83c410          add esp, 0x10
            0x565ff5eb      b800000000      mov eax, 0
            0x565ff5f0      8d65f8          lea esp, [var_8h]
            0x565ff5f3      59              pop ecx
            0x565ff5f4      5b              pop ebx
            0x565ff5f5      5d              pop ebp
            0x565ff5f6      8d61fc          lea esp, [ecx - 4]
            0x565ff5f9      c3              ret
            ; CALL XREF from sym.doNothing @ 0x565ff553
  4: sym.__x86.get_pc_thunk.ax ();
            0x565ff5fa      8b0424          mov eax, dword [esp]
            0x565ff5fd      c3              ret
            0x565ff5fe      6690            nop
  93: sym.__libc_csu_init (int32_t arg_2ch_2, int32_t arg_2ch, int32_t arg_28h);
            ; arg int32_t arg_2ch_2 @ esp+0x2c
            ; arg int32_t arg_2ch @ esp+0x30
            ; arg int32_t arg_28h @ esp+0x38
[0x565ff571]> db 0x565ff5e2
[0x565ff571]>
```

Setting a breakpoint at 0x565ff5e2 or push eax will allow us to find the exact value of the number being passed into the function doNothing().

```
hit breakpoint at: 0x565ff5e2
[0x565ff5e2]> drr
role reg         value    refstr
————————————————————————————————————————
A0   eax         d4faf720 eax,edx
```

Checking the registers once we hit the breakpoint tells us that EAX equals d4faf720 which looks like a hex number. When EAX is converted to decimal it equals -721750240.

Flag: picoCTF{-721750240}