hijack Plan wirteup

Author : yaseen

Checking the binary for all the mitigations with checksec we get the following protections enabled.

```
  Arch:       amd64-64-little
   RELRO:      Partial RELRO
   Stack:      No canary found
   NX:         NX enabled
   PIE:        No PIE (0x400000)
   Stripped:   No
```

``

```
└─$ file hijack
hijack: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically
linked, in
terpreter /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=be44560e6e8bd1671714da1a5e6f7b
54ddcf99bf, for GNU/Linux 3.2.0, not stripped
```

reversing

- The first `360 lines` in main are for printing the flash lights and the asci art so we ignore them

```
360   }
361   menu();
362   puts("\n\n");
363   puts("                                        __|__");
364   puts("                                --o--o--(_)--o--o--");
365   local_10 = 0;
366   do {
367     if (1 < local_10) {
368       puts("Your current balance is");
369       puts("$0");
370       puts("Please set down while the plane takes off");
371       return 0;
372     }
373     __isoc99_scanf(&DAT_00403146,&local_30c);
374     if (local_30c == 1) {
375       __isoc99_scanf(&DAT_0040314a,local_2f8);
376       local_18 = dlsym(0,local_2f8);
377       printf("%p\n",local_18);
378     }
379     else {
380       if (local_30c != 2) {
381                     /* WARNING: Subroutine does not return */
382         exit(1);
383       }
384       __isoc99_scanf("%lu %lu",&local_300,&local_308);
385       *local_300 = local_308;
386     }
387     local_10 = local_10 + 1;
388   } while( true );
389 }
390
```

- Prints the menu
- does a scanf to take our menu option
- double click on DAT_00403146 and see it takes %2d; tow decimal digits

option 1 in menu:

- another scanf, double click on DAT_0040314a to see it reads %10s; a string of max 10 characters
- calls `dlsym()` on our input string
- prints the result from dlsym
- A quick search on dlsym() function to figure out what it does yields this :

> ***DESCRIPTION***
>
> The *dlsym*() function shall obtain the address of a symbol defined within an object made accessible through a *dlopen*() call. The *handle* argument is the value returned from a call to *dlopen*() (and which has not since been released via a call to *dlclose*()), and *name* is the symbol's name as a character string.
>
> The *dlsym*() function shall search for the named symbol in all objects loaded automatically as a result of loading the object referenced by *handle* (see *dlopen*()). Load

- So it takes a symbol defind in glibc and returns it's address, so passing the string `system` would give us the system() address

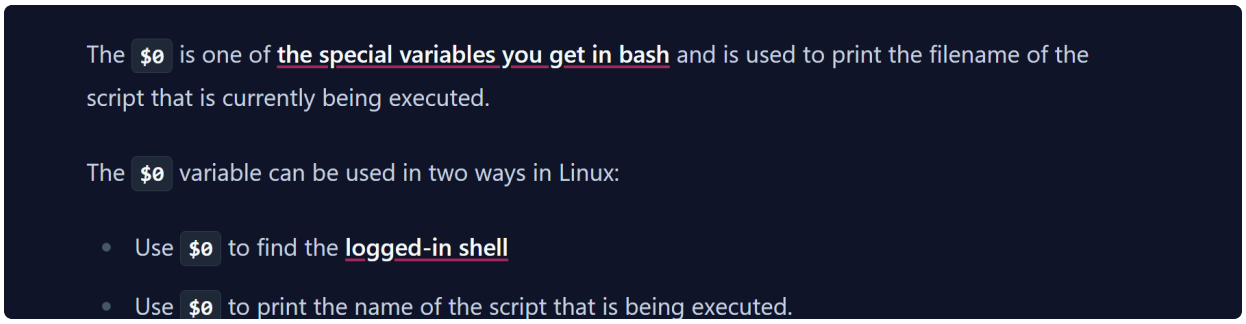
option 2 in menu:

- Reads two unsigned long integers from input.
- Stores the first in local_300
  ➜ local_300 is a pointer (e.g., unsigned long *)
- Stores the second in local_308
  ➜ local_308 is an unsigned long

  *local_300 = local_308;
- Assigns the value of local_308 to the memory pointed to by local_300.

conclusion

- We could reveal sysmte() address from option 1, and use it some address in option 2
- notice when the program finishes it does *puts("$0")* , another quick search on "$0" in linux, yields this:



- so $0 is the shell name, by calling it we get an interactive shell:

- check the solve.py