

LAB 2 – NULL POINTER DEREFERENCE

Changes

Before telling you about which files that I change, let take a look at how things working in the xv6. The first thing is forking a new process, as a memory perspective, the important function is “copyvm” that you can find it in a vm.c. Also we should change the location that program loads in exec() from 0 to PGSIZE.

To stop compiler from load into page 0 we should also change the make file.

- 1) **vm.c:** Instead of staring copying in copyvm from page 0, i.e. ‘i=0’, we should start from i=PGSIZE which is the next page, i.e. page 1;

```
for(i = PGSIZE; i < sz; i += PGSIZE){
```

- 2) **exec.c:** Load program into page 1 of the memory (userspace);

```
sz = PGSIZE;
```

- 3) **MakeFile:** Change start execution address to 0x1000 (4096) [each page is 4k]

```
_%. %.o $(ULIB)
$(LD) $(LDFLAGS) -N -e main -Ttext 0x1000 -o $@ $^
$(OBJDUMP) -S $@ > $.asm
$(OBJDUMP) -t $@ | sed '1,/SYMBOL TABLE/d; s/ .* //; /^$$/d' > $.sym
```

```
_forktest: forktest.o $(ULIB)
# forktest has less library code linked in - needs to be small
# in order to be able to max out the proc table.
$(LD) $(LDFLAGS) -N -e main -Ttext 0x1000 -o _forktest forktest.o ulib.o usys.o
$(OBJDUMP) -S _forktest > forktest.asm
```

These changes are enough for Null pointer dereference without kernel, but when we need have a syscall, we should change the required code for add syscall (like lab 1), plus additional changes in the syscall.c for checking pointers before passing (argptr());

- 4) **Sysproc.c:**

```
int
sys_getdrsvrptr(void)
{
    char *pointer ;

    if(argptr(0, &pointer, 1) < 0) {
        cprintf("DRSVR Found NULL POINTER IN KERNEL! :( \n");
        return -1;
    }
    else {
        cprintf("DRSVR Pointer has been set in kernel! :) \n");
        return 0;
    }
}
```

LAB 2 – NULL POINTER DEREFERENCE

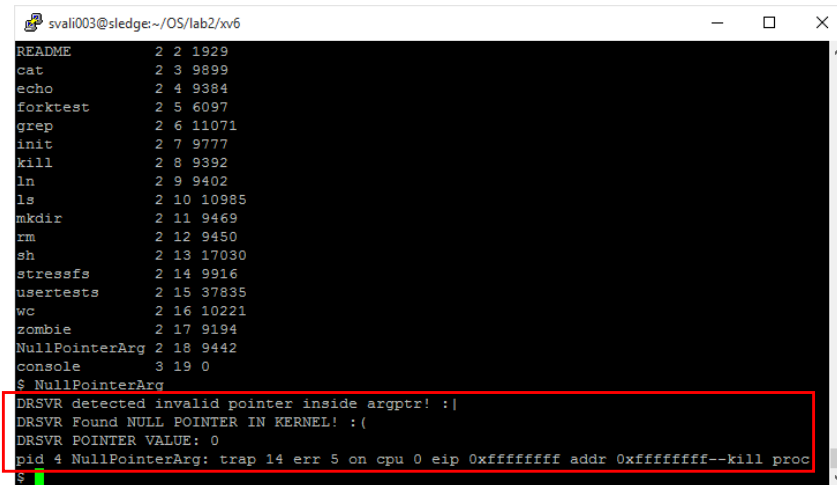
5) **Syscall.c:**

```
if((uint)i >= proc->sz || (uint)i+size > proc->sz || (uint)i == 0 ) {  
    // DRSVR - CS202 - LAB2  
    cprintf("DRSVR detected invalid pointer inside argptr! :| \n");  
    return -1;  
}
```

6) **Proc.h, Syscall.h, User.h, Usys.s, Defs.h; [For adding Syscall]**

7) **NullPointerArg.c:** User code for testing my job ☺

```
#include "types.h"  
#include "user.h"  
  
int main(int argc, char* argv[]){  
  
    int * myPointer = 0; // for null dereference  
    char * myPointer2 = 0; // for kernel test  
  
    getdrsvrptr(myPointer2); // syscall kernel test  
  
    printf(1,"DRSVR POINTER VALUE: %d\n",myPointer); // user test  
  
    return 0;  
}
```

A terminal window titled 'svali003@sledge:~/OS/lab2/xv6' displays a list of processes with their IDs, names, and PPIDs. The processes listed are: README (2, 2, 1929), cat (2, 3, 9899), echo (2, 4, 9384), forktest (2, 5, 6097), grep (2, 6, 11071), init (2, 7, 9777), kill (2, 8, 9392), ln (2, 9, 9402), ls (2, 10, 10985), mkdir (2, 11, 9469), rm (2, 12, 9450), sh (2, 13, 17030), stressfs (2, 14, 9916), usertests (2, 15, 37835), wc (2, 16, 10221), zombie (2, 17, 9194), NullPointerArg (2, 18, 9442), and console (3, 19, 0). Below the list, the user has entered '\$ NullPointerArg'. The output shows: 'DRSVR detected invalid pointer inside argptr! :|', 'DRSVR Found NULL POINTER IN KERNEL! :(', 'DRSVR POINTER VALUE: 0', and 'pid 4 NullPointerArg: trap 14 err 5 on cpu 0 eip 0xffffffff addr 0xffffffff--kill proc'. The last line is highlighted with a red box.

```
svali003@sledge:~/OS/lab2/xv6  
README      2 2 1929  
cat          2 3 9899  
echo        2 4 9384  
forktest    2 5 6097  
grep        2 6 11071  
init        2 7 9777  
kill        2 8 9392  
ln          2 9 9402  
ls          2 10 10985  
mkdir       2 11 9469  
rm          2 12 9450  
sh          2 13 17030  
stressfs    2 14 9916  
usertests   2 15 37835  
wc          2 16 10221  
zombie      2 17 9194  
NullPointerArg 2 18 9442  
console     3 19 0  
$ NullPointerArg  
DRSVR detected invalid pointer inside argptr! :|  
DRSVR Found NULL POINTER IN KERNEL! :(  
DRSVR POINTER VALUE: 0  
pid 4 NullPointerArg: trap 14 err 5 on cpu 0 eip 0xffffffff addr 0xffffffff--kill proc  
$
```

You can find my changes by searching for DRSVR or CS202 inside the code ☺ As you can see process has been trapped and killed because of Null pointer dereference.