Lab 3: Program loading and memory mapping

Ankit Goyal
ankit@cs.utexas.edu
CS380L

October 26, 2014

1 Setup

1.1 Hardware

Host Processor: 64 bit 4 core Intel(R) Xeon(R) CPU E3-1270 V2 @ 3.50GHz
Host Memory: 16GB
HyperThreading: Yes
Logical CPUs after Hyperthreading: 8
CPU frequency scaling: Disabled in BIOS (turned off Intel SpeedStep and C-states)

1.2 Software

Host Operating System: Ubuntu with 3.13.0-34-generic 64 bit kernel.

2 Creating the memory image of a new process

`sys_execve` is responsible for setting up the environment for running the program. Below are the steps taken by `sys_execve` which calls `do_execve_common`:

1. Check that `NPROC` limit is not exceeded (i.e., total number of process), if it is then exit. (L: 1443)
2. Allocate memory for data structure in kernel. (L: 1458)
3. Open the `exec` file using `do_open_exec` (L: 1469)
4. Now the kernel data structures are initialized and `exec_binprm` is called.
5. `exec_binprm` calls `search_binary_handler` which finds the binary format handler, in our case elf. So it finds `load_elf_binary`. (fs/binfmt_elf.c L:84 & 571)
    - `load_elf_binary` does consistency checks by making sure that it's an ELF format file by comparing the main number and ELF in `e_ident` field in header.
    - `load_elf_binary` reads the header information and looks for `PT_INTERP` segment to see if an interpreter was specified. This segment is only present for dynamically linked programs and not for statically linked.

Time Spent on the lab ≈ 30 hours

3 References

1. http://eli.thegreenplace.net/2012/08/13/how-statically-linked-programs-run-on-linux/
2. http://www.skyfree.org/linux/references/ELF_Format.pdf
3. http://linux.die.net/man/5/elf
4. http://articles.manugarg.com/aboutelfauxiliaryvectors.html
5. http://pubs.opengroup.org/onlinepubs/009695399/functions/sigaction.html
6. http://stackoverflow.com/questions/8116648/why-is-the-elf-entry-point-0x8048000-not-changeable
7. http://lxr.free-electrons.com/source/fs/exec.c#L1425