CS342 - Project 4

Process inspector kernel module

Çelik Köseoğlu, Melis Kızıldemir - 25 April 2017

Report

Introduction

In this project we implemented a kernel module, which retrieves Process Control Block information. The functionalities our module supports are as follows:

- Printing Process Information
  - Process Name, ID
  - Parent Name, ID
  - Children Names, IDs
- Printing Virtual Memory Information
  - VM region start address
  - VM region end address
  - VM region block size
  - Total memory used in KB
- Printing File Open File Information
  - Names of the files opened by the process
  - Sizes of the files opened by the process
  - Block sizes of files opened by the process
  - Block start addresses of the files opened by the process

Usage

Pid of the process to be inspected is given as an argument to the module when instantiating it. Ellie is the name of the kernel module.

```
$ insmod ellie.ko pid=1234
```

The following command prints the output from the kernel module

```
$ dmessg | tail -50
```

Finally, remove the module

```
$ rmmod ellie.ko
```

Implementation

Printing Process Information

The pid is taken from the user as an argument. The pid_task() function is used to get the task_struct of the associated process.

the task_struct contains fields like comm and id which are the process name and process id respectively.

for printing the parent process information, we have used task_struct's parent field which points to the parent process' task_struct

for printing the children information, we have used the children field of the task_struct. This field is a pointer to a list_head. C-lang contains macros to traverse the list. Check the code for more information.

Printing Virtual Memory Information

The task_struct contains the mm field which contains the memory information. It has a linked list which contains virtual memory node information. We compute the virtual memory block size by subtracting the start address (vm_start) field of the vm_area_struct from the end address field (vm_end). A process has many of these blocks which are stored in a linked list. We add all these blocks to find the total memory used by the process in bytes. Then we convert it into KB by dividing into 2^{10}.

Printing File Access Status

To see which files are being used by the process, we first need to find out the number of files currently used by the process. We were able to access this from the count field of file_struct. We then traverse the file descriptor array (fd_array) of the file and if the file path (f_path.dentry) is not null we printed it. It is important to note that we used dentry_path_raw() method to be able to list the full file path, not just the file name.

To learn about the file size, block address and block size of the file, we needed to get the stat of the file which is type kstat. Check code for more information about this.

Sample Run

We will be inspecting Google Chrome browser in this run. Assume that there are multiple instances of chrome running of the machine. we get one of the ids by typing

```
$ pidof chrome

     >3455 3426 3413 3399 3396 3343 3329 3314
```

Now we can select an arbitrary one to load our module.

```
$ insmod ellie.ko pid=3343
```

Then we print the output from out module by typing

```
$ dmessg | tail -30

     >Process: [chrome] with pid: [3329]
     >Parent Process: [chrome] with pid: [3325]
     >Child Process: [chrome] with pid: [3413]
     >Child Process: [chrome] with pid: [3426]
     >Child Process: [chrome] with pid: [3455]
     >File count of the process: 8 -> (*read below for what it is 5 + 3)
     >open file at path: [/opt/google/chrome/icudtl.dat]
     >size of file: 10130464 bytes
     >first block address in dec: 19792
     >block size: 4096 bytes
     >open file at path: [/opt/google/chrome/snapshot_blob.bin]
     >size of file: 1279724 bytes
     >first block address in dec: 2504
     >block size: 4096 bytes
     >open file at path: [/opt/google/chrome/natives_blob.bin]
     >size of file: 334730 bytes
     >first block address in dec: 656
     >block size: 4096 bytes
     >open file at path: [/opt/google/chrome/chrome_100_percent.pak]
     >size of file: 515532 bytes
     >first block address in dec: 1008
     >block size: 4096 bytes
     >open file at path: [/opt/google/chrome/chrome_200_percent.pak]
     >size of file: 813163 bytes
     >There are too many blocks. Printing information about just the first one
to avoid confusion. Check report about this.
     >Start address in dec: 7821279195136
     >End address in dec: 7821279199232
     >Total size of block: 4096 bytes
     >Total memory used by process: 385500K
```

*By default all processes have 3 files open. The standard input, standard output and the standard error with file descriptors 0,1,2 respectively. So, we don't print information about these files.