System Programming

Programming Assignment #4 Kernel Lab

2013-11826 임주경

- Goal

이번 과제 커널 랩을 구현하면서 시스템 디버깅용 리눅스 파일 시스템 debugfs를 이용해 커널 모듈을 직접 구현하고, Kernel Level에 접근해 User Level에서 제한된 작업을 수행하도록 한다. 커널과 리눅스 시스템에 대해 익숙해지고자 한다.

- Part #1. Ptree

User level에서 프로세스에 대한 정보 접근에는 제한이 있다. 이를 커널 모듈을 이용해 프로세스 정보 (프로세스 트리)에 접근하여 원하는 데이터를 얻어내고자 한다.

Input은 데이터를 원하는 프로세스의 pid를 변수로 입력하며, echo 명령어를 사용한다. 따라서, echo 명령어의 정상적인 작동을 위해 write_pid_to_input 함수를 file_operations 구조체 dbfs_fops에 등록한다. 또한 ptree 디렉토리 내부에 input 파일을 생성한다. Write_pid_to_input 함수는 입력받은 pid로 pid_task 함수를 이용해 얻어낸 task_struct 구조체로 프로세스 정보를 얻으며, 얻어낸 정보에서 부모 프로세스의 pid를 추적해가며, pid = 1 인 init 프로세스까지 트리를 따라 이동하도록 구현한다. 이 과정에서 while 문을 사용하였다. Pid_task 의 첫번째 매개변수는 프로세스 구조체 포인터를 받으므로, find_vpid 함수를 이용한다. 원하는 output을 출력하기 위해서 blob 타입 구조체가 필요하다. 따라서, 편의상 debugfs_blob_wrapper 구조체를 새로운 타입 bw로 정의하였다. Bw 포인터 변수 my_bw를 정의하였고, 모듈 init 함수에서 debugfs_creat_blob 함수를 이용해 my_bw에 저장된 출력 내용을 파일에 담는다. Debugfs_create_file 의 두번째 매개변수의 mode 값은 0644로 설정하였고, blob을 담을 파일의 mode 값은 0444로 설정하였다. 부모 프로세스를 추적하는 과정에서 만약의 경우 부모 프로세스가 자식 프로세스를 wait 하지 않으면, parent 변수가 실제 parent 와 다른 값을 갖게 되므로, real_parent 변수를 사용하였다. Init 함수에서 출력의 결과 answer와 blob은 kmalloc을 이용해 동적할당 하였으며, 모듈 exit 함수에서 debugfs_remove_recursive 함수와 kfree 함수를 이용해 할당했던 파일과 디렉토리를 모두 제거하며 모듈을 마치도록 한다.

- Part #2. Paddr

디버그 파일 시스템 함수 코드 내에서 user_buffer로 받아온 패킷 구조체의 포인터를 패킷 구조체

변수로 정의해주기 위해서, packet 구조체를 선언해주었다. User_buffer에 저장된 pid, virtual address를 새로 정의한 packet 구조체의 변수 값에 초기화 시킨다. Pid_task 함수를 이용해서, user_buffer에 저장된 입력 패킷의 pid로 원하는 프로세스의 task 구조체를 얻어낼 수 있었다.

얻어낸 task 구조체와 virtual address 값을 이용해서 pgd->p4d->pud->pmd->pte의 추적을 이용해 physical address를 얻어낸다. 이때, pgd_offset, p4d_offset, pud_offset, pmd_offset, pte_offset_kernel 함수를 이용하였으며, 12-bit의 offset은 virtual address를 통해서도 얻을 수 있다. Pte를 통해 얻어낸 ppn과 offset을 합쳐, physical address를 얻어내도록 구현하였다.

Packet 구조체 포인터의 paddr 값을 얻어낸 physical address 값으로 설정하여 원하는 결과를 얻어내도록 한다. 이때, file_operations 구조체 dbfs_fops의 read 값으로 read_output 함수를 등록해 user_buffer 값을 정상적으로 읽어올 수 있도록 구현하였다.

- 마치며

이번 과제 구현에서 어려웠던 점은 리눅스 커널의 함수들의 용도와 매개변수 타입 및 반환 타입에 대해서 찾아봐야 했으며, 익숙하지 않은 디버깅 파일 시스템을 이용한 모듈 구현에서 파일을 읽기, 쓰기 관계가 어떻게 형성되는지 이해하는 것이 어려웠다. Debugfs와 kernel 참고자료를 통해 create 함수들에 입력될 매개변수 값 중 mode 값을 구해내는 과정에서 어려움을 겪었다. 또한 기존의 c언어에서 사용되던 함수들과 다르게 kernel에서 사용되기 위한 함수들을 이용해야 한다는 점이 많이 낯설었지만, 과제 구현하는 과정을 통해서 kernel과 리눅스 시스템, 파일 관리 체계에 대해서 조금은 이해를 할 수 있게 되었다.