Kernel Lab Report

2019-12333 정윤서

사용 가상머신 커널 version: 5.15.0-1036-aws

1. 커널 랩의 목표

커널 랩에서는 linux kernel module을 구현함으로써 커널 소스 코드를 수정하고 recompile 할 필요 없이 동적으로 새로운 기능을 추가하는 방법에 대해 배울 수 있다. 이번 랩에서는 ptree와 paddr라는 두 가지 기능을 위한 kernel module을 구현하였다. Ptree는 유저가 입력한 input에 대한 Process tree를 file에 write하고, paddr는 user process에서의 특정 virtual address에 대한 physical address를 반환한다. 그런데 두 가지 기능 모두 user space와 kernel space간의 interaction이 필요하다. 일반적으로 user application에서 kernel interface를 활용하기 위해서는 system call을 해야 하지만, 커널 코드의 수정 없이 kernel module을 활용하기로 하였으므로 kernel module에서의 user-to-kernel interface를 지원해 주는 debugfs를 활용한다. 이번 랩에서는 linux kernel module과 debugfs api를 이용해 debugfs에 대한 write, read system call을 hook함으로써 kernel functionality를 extend하는 방법을 실습해 볼 수 있었다.

2. 구현방식

첫째, ptree(dbfs_ptree.c)에서는 write_pid_to_input이라는 write용 debugfs handler를 구현하여 /sys/kernel/debug/ptree/input 파일에 대한 write system call을 hook한다. 이때 write_to_input()에서는 user_buffer로 주어지는 입력값(pid)에 대한 task_struct를 찾고, parent 속성을 이용해 iterative하게 거슬러 올라가며 init process(system)까지의 process들을 linked list로 연결한다(list_head와 list_add 이용). 여기서 얻어진 ptree를 /sys/kernel/debug/ptree/ptree에 write해야 하는데, dbfs_module_init에서 ptree 파일을 생성할 때 'debugfs_create_blob'을 이용하여 생성하였으므로 ptree 파일을 열고 write할 필요 없이 static으로 선언한 blob struct를 수정해 주기만 하면 된다. 이를 위해 앞서 만든 process linked list를 순회하며 blob을 차례차례 수정하고, 방문한 process의 메모리는 해제해 주었다.

둘째, paddr(dbfs_paddr.c)에서는 read_output이라는 read용 debugfs handler를 구현하여 /sys/kernel/debug/paddr/output 파일에 대한 read system call을 hook한다. Read_output()에서는 read() 함수의 인자로 주어진 struct에 담긴 vaddr에 해당하는 PTE를 찾기 위해, 각 level의 page table offset을 하나씩 찾아가는 과정을 반복한다. 다섯 번째 page table에서 찾은 PTE가 valid하다면, page size가 4KB이므로 해당 pte에서 extract한 ppn 값을 12번 left shift한 것을 VPO와 or연산하여 user_buffer로 주어진 struct의 paddr에 넣어 주었다(copy_to_user).

3. 느낀 점

커널 코딩에 익숙하지 않아 어떤 함수를 사용할 수 있고, 어떤 함수를 사용할 수 없는지 구분하는 것이 어려웠다. 또 가장 처음에는 ptree 파일을 blob을 이용해서 생성한다는 생각을 하지 못했기 때문에 write_pid_input 내에서 ptree 파일을 열고 쓰는 과정에서 계속 오류가 났고, kernel crash가 발생해서 전반적으로 버그를 잡는 시간이 연장되는 바람에 난항을 겪었던 것 같다. 이번 랩을 통해 system call 코드를 직접적으로 수정하지 않고도 기능을 추가하고, 특정 종류의 file system을 통해 유저 단과 커널 단의 interface를 형성할 수 있는 방법에 대해 배울 수 있었다.