TASK 2: REPORT

Yun Seong Nam & Naif Almakhdhub

March 1, 2016

Contributions

Initially, Yun worked on `procfs` and Naif worked on exception handling. Yun located the place to print at `show_map_vma()` and the formatting. We worked together on counting (0,x). We have been working together for a large part afterwards (Meeting Tu/Thu and weekends) and the tasks almost merged. Creating the data structure, handling exception, patching instruction, restoring instruction and counting was mostly done by Yun. The part for invalidating/validating and handling page faults was mostly done by Naif.

Overall Design

Our final design can be summarized as follows:

1. After a task is initialized, we start monitoring upon calling `/proc/PID/maps`
2. In order to monitor a page accessed by the a current procedure, we modified `task_struct` structure in the `/include/linux/sched.h` file by adding a linked list of reference counter structure we created. Key variables for our reference counter structure are `pc` and `instruction`. A variable `pc` is for saving `pc` right after the faulty instruction (could be `pc`+4 or `pc`+2). It is used for recognizing undefined instruction that is made by us. Also a variable `instruction` is used for restoring instruction after the faulty instruction when we find an undefined instruction made by us.
3. Within `show_map_vma()`, we invalidate the hardware pte. This is the signal to monitor the page entries for given pid.
4. When the pte is accessed which we are monitoring, this will cause a `do_DataAbort()` exception. There, we validate the hardware pte, then patch a next instruction by saving the next PC and next instruction, and modify the next instruction as "undefined" and store actually next instruction and pc inside data structure we made for restoring later. This will return to user mode, execute the faulty instruction correctly.
5. Since we make next instruction undefined, this will cause exception again after executing faulty instruction correctly. When the `do_undefinstr()` is caused, we catch it and verify that this undefined instruction made by us. Once it is verified, we invalidate previous hpte again for future monitoring and restore the instruction we saved using `put_user` and silently return to user mode again.

Noteworthy features

For the **Heap**, we are able to count 0...9 successfully. For other parts, counting is done as: **(a)** the page is not initialized, hence we print <.> **(b)** Exists, but the young bit is not set, we print <0> **(c)**Young bit is set, that will print <x>

Known Bugs

1. The repeated monitoring(0..9) is triggered once we access `/proc/PID/maps`.
2. For the repeated monitoring(0..9), we only track page entries in `Heap`.
3. But we can track 0, x and . every pids/entries without triggering.
4. We check the `thumb mode` and `is_wide_instruction`, but does not check `branch instruction`.

Side Notes

We believe we are able to invalidate every pte entry within `../fault.c` as a procedure is created. However, since we faced the bugs which some page entries caused problem when we invalided them, we targeted **Heap** from `/proc/PID/maps` as safer choice in order to obtain counting until 9.