

## Re: [PATCH][RFC] kprobes: Add user entry–handler in kretprobes

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-11/msg04389.html>

---

- *From:* "Abhishek Sagar" <[sagar.abhishek@xxxxxxxxxx](mailto:sagar.abhishek@xxxxxxxxxx)>
  - *Date:* Wed, 14 Nov 2007 14:19:04 +0530
- 

On Nov 14, 2007 1:27 PM, Srinivasa Ds <[srinivasa@xxxxxxxxxx](mailto:srinivasa@xxxxxxxxxx)> wrote:

- 1) How do you map the entry\_handler(which gets executed when a process enters the function) with each instance of return probe handler.  
I accept that entry\_handler() will execute each time process enters the function, but to calculate time, one needs to know corresponding instance of return probe handler(there should be a map for each return handler).

Yes, a one–to–one correspondence between the entry and return handlers is important. I've tried to address this by passing the same kretprobe\_instance to both the entry and return handlers.

Let me explain briefly.

Suppose in a SMP system, 4 processes enter the same function at almost sametime(by executing entry\_handler()) and returns from 4 different locations by executing the return handler. Now how do I match entry\_handler() with corresponding instance of return handler for calculating time.

Right, and this scenario would also occur on UPs where the kretprobe'd function has nested calls. This has been taken care of (see below).

Now What I think is, there could be 2 solutions to these problem.

- a) Collect the entry time and exit time and put it in that kretprobe\_instance structure and fetch it before freeing that instance.

In case someone wants to calculate the entry and exit timestamps of a function using kretprobes, the appropriate place for it is not the entry and return handlers. Thats because the path from when a function is called or from when it returns, to the point of invocation of the entry or return handler is not O(1).

Looking at trampoline\_handler(), it actually traverses a list of all

Re: [PATCH][RFC] kprobes: Add user entry-handler in kretprobes

pending return instances to reach the correct return instance. So any kind of exit timestamp must be placed just before that and passed to the return handler via kretprobe\_instance (as you just suggested).

b) Or pass ri(kretprobe\_instance address to entry\_handler) and match it with return probe handler.

This is what I'm trying to do with this patch. I hope I've not misread what you meant here, but as you'll notice from the patch:

```
+ if (rp->entry_handler) {
+ copy = *regs;
+ arch_prepare_kretprobe(ri, &copy);
+ if (rp->entry_handler(ri, &copy))
+----- (entry-handler with ri)
+ goto out; /* skip current kretprobe instance */
+ *regs = copy;
+ } else {
+ arch_prepare_kretprobe(ri, regs);
+ }
```

the entry handler is called with the appropriate return instance. I haven't put any explicit "match" test here for ri. The reason is that the correct ri would be passed to both the entry and return handlers as trampoline\_handler() explicitly matches them to the correct task. Note that all pending return instances of a function are chained in LIFO order. S the entry-handler which gets called last, should have its return handler called first (in case of multiple pending return instances).

Another cool thing here is that if the entry handler returns a non-zero value, the current return instance is aborted altogether. So if the entry-handler fails, no return handler gets called.

Does this address your concerns?

--  
Regards  
Abhishek Sagar

-  
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxx  
More majordomo info at <http://vger.kernel.org/majordomo-info.html>  
Please read the FAQ at <http://www.tux.org/lkml/>

-  
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxx

Re: [PATCH][RFC] kprobes: Add user entry–handler in kretprobes

More majordomo info at <http://vger.kernel.org/majordomo–info.html>

Please read the FAQ at <http://www.tux.org/lkml/>