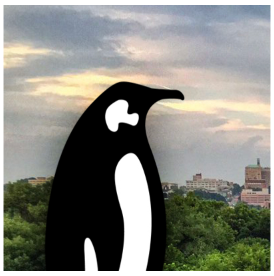

Contribution ID: **90** Type: **not specified**

Function return hook integration with Function graph tracer

Tuesday 14 November 2023 15:10 (20 minutes)

Currently, there are two different function return hook mechanisms: kretprobe (rethook) and function graph tracer. Both have similar but different ways to hook the function return. They both modify the return address on the stack (or link-register) with their trampoline code and save the correct return address to their own list for each task. The difference is how they allocate the per-task list. Kretprobe allocates a linked list of storage objects when it is initialized. Users can specify the maximum number of concurrently used objects at that point. However, the object list is not shared among kretprobes. On the other hand, function graph tracer allocates a shadow stack array for each task when it is enabled. This is simpler but consumes more memory at that point. However, this shadow array will be shared among all functions, so each function takes up very little memory.
The problem is that if both mechanisms are used at the same time, they both allocate such memory independently, and the function return is hooked twice or more. This is inefficient. To avoid this, we can integrate them. However, it might be better to remove kretprobes and use fprobe's exit handler to simplify the solution.
In short, there are two different function return hook mechanisms, kretprobe and function graph tracer. They both have similar but different ways to hook the function return. The problem is that if both are used at the same time, they both allocate such memory independently, and the function return is hooked twice or more. This is inefficient. To avoid this, we can integrate them or remove kretprobes and use fprobe's exit handler.

In this session, we will talk about the background and how to do this, and my expectations. BPF and other tools may need some more work about this change.

Primary author: Mr HIRAMATSU, Masami (Google)

Presenter: Mr HIRAMATSU, Masami (Google)

Session Classification: Tracing MC

Track Classification: LPC Microconference: Tracing MC