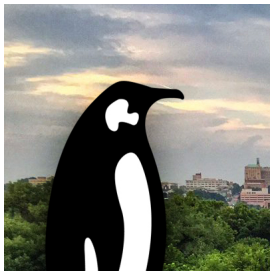

Contribution ID: **298** Type: **not specified**

When BPF programs need to die : exploring the design space for early BPF termination

Monday, 13 November 2023 10:30 (30 minutes)

In the rapidly evolving landscape of BPF as kernel extensions, the need for early termination is becoming increasingly critical, whether it's due to kernel stalling or the need to enforce execution time restriction on critical hook points :

- The recently added bpf_loop helper can be used to attach a very long running BPF program into the kernel which has been demonstrated to stall the kernel.
- Feature addition like BPF exception[1], which allows a programmer to throw errors and terminate the BPF program, is still limited by its inability to auto-cleanup resources that were allocated in runtime.
- Using BPF-orchestrators[2], operators want to impose execution time restrictions on critical hook points, so that the overall system performance is not hindered due to kernel extensions. Resource contention and other factors can severely affect the runtime of a BPF program, which based on the system load and demand, will call for immediate termination before throttling subsequent requests.

Thus, the situation now demands a mechanism to allow abrupt termination of any running BPF program as well as perform cleanup of allocated kernel resource to avoid resource leaks.

In this talk, we will provide an overview of the design space for BPF termination, discussing the advantages and limitations of each approach. Then, we will introduce a fast-path approach to allow abrupt termination of kernel extensions which leverages the verifier's knowledge to achieve zero-bookkeeping safe cleanup. In order to create a fast-path, all helper function calls are patched to halt further resource allocation or make costly function calls. The patching mechanism makes exceptions for helpers which release resources such that all existing locks and reference can be efficiently released. Due to the verifier guarantees based on range analysis and branch traversal, it is guaranteed to cleanup resources regardless of runtime branch decisions.

Towards the end of this talk, we will discuss about some known limitations and assumptions in our work. Most importantly, we invite the community to give feedback on how we can refine this work for a contribution towards the Linux upstream.

[1] : https://lwn.net/Articles/944372/
[2] : Sahu, Raj, and Dan Williams. "Enabling BPF Runtime policies for better BPF management." Proceedings of the 1st Workshop on eBPF and Kernel Extensions. 2023.

Primary authors: WILLIAMS, Dan (Virginia Tech); SAHU, Raj (Virginia Tech)

Presenters: WILLIAMS, Dan (Virginia Tech); SAHU, Raj (Virginia Tech)

Session Classification: eBPF & Networking

Track Classification: eBPF & Networking Track