Contribution ID: **290** Type: **not specified**

Revisiting eBPF Seccomp Filters

Monday 12 September 2022 15:30 (30 minutes)

Seccomp, the widely used system-call security module in Linux, is among the few that still exposes classic BPF (cBPF) as the programming interface, instead of the modern eBPF. Due to the limited programmability of cBPF, today's Seccomp filters mostly implement static allow-deny lists. The only way to implement advanced policies is to delegate them to user space (e.g., Seccomp Notify); however, such an approach is error prone due to time-of-check time-of-use issues and costly due to the context switch overhead.

Over the past several years, supporting eBPF filters in Seccomp has been brought up (e.g., by Dhillon [1], Hromatka [2], and our team [3]) and has raised many offline discussions on the mailing lists [4]. However, the community has not been convinced that eBPF for Seccomp is 1) necessary nor 2) safe, with opinions like "Seccomp shouldn't need it..." and "rather stick with cBPF until we have an overwhelmingly good reason to use eBPF..." preventing its inclusion.

We have developed a full-fledged eBPF Seccomp filter support and systematically analyzed its security [5]. In the proposed presentation, using the insight from our system, we will (1) summarize and refute concerns on supporting eBPF Seccomp filters, (2) present our design and implementation with a holistic view, and (3) open the discussion for the next steps.

Specifically, to show that eBPF for Seccomp is necessary, we describe several security features we build using eBPF Seccomp filters, the integration with container runtime like crun, and performance benchmark results. To show that it is safe, we further describe the use of root-only eBPF Seccomp in container-based use cases, which strictly obey current kernel security policies and still improve the usefulness of Seccomp. Further, we will go over the key designs for security, including protecting kernel data, maintaining consistent helper function capability, and the potential integration with IMA (the integrity measurement architecture).

Finally, we will discuss future opportunities and concerns with allowing unprivileged eBPF Seccomp and possible avenues to address these concerns.

Reference:
[1] Dhillon, S., eBPF Seccomp filters. https://lwn.net/Articles/747229/
[2] Hromatka, T., [RFC PATCH] all: RFC - add support for ebpf.
https://groups.google.com/g/libseccomp/c/pX6QkVF0F74/m/ZUJlwI5qAwAJ
[3] Zhu, Y., eBPF seccomp filters, https://lwn.net/Articles/855970/
[4] Corbet J.,eBPF seccomp() filters, https://lwn.net/Articles/857228/
[5] https://github.com/xlab-uiuc/seccomp-ebpf-upstream/tree/v2

I agree to abide by the anti-harassment policy

Yes

Primary authors: JIA, Jinghao (University of Illinois Urbana-Champaign); ZHU, YiFei (Google); ARCANGELI, Andrea (Red Hat); Dr FRANKE, Hubertus (IBM); FELDMAN-FITZTHUM, Tobin (IBM); Prof. CANELLA, Claudio (Graz University of Technology); Prof. SKARLATOS, Dimitrios (Carnegie Mellon University); Prof. GRUSS, Daniel

(Graz University of Technology); Prof. WILLIAMS, Dan (Virginia Tech); Prof. XU, Tianyin (University of Illinois at Urbana-Champaign)

Presenters: JIA, Jinghao (University of Illinois Urbana-Champaign); Prof. XU, Tianyin (University of Illinois at Urbana-Champaign)

Session Classification: eBPF & Networking

Track Classification: eBPF & Networking Track