Contribution ID: **89** Type: **not specified**

seccomp and libseccomp performance improvements

Tuesday, 13 November 2018 17:00 (30 minutes)

This is probably a better fit as a CfP in either the containers or BPF microconferences.

seccomp is a critical component to ensuring safe containerization of untrusted processes. But at Oracle we are observing that this security often comes with an expensive performance penalty. I would like to start a discussion of how can we can improve seccomp's performance *without compromising security*.

Below is an open RFC I have in libseccomp that should *significantly* improve its performance when processing large filters. I would like to discuss other performance improvement possibilities - eBPF in general, eBPF hash map support, whitelists vs blacklists, etc. I would gladly take requests and ideas and try to incorporate them into libseccomp and seccomp as appropriate.

https://github.com/seccomp/libseccomp/issues/116

Several in-house Oracle customers have identified that their large seccomp filters are slowing down their applications. Their filters largely consist of simple allow/deny logic for many syscalls (306 in one case) and for the most part don't utilize argument filtering.

After invaluable feedback from Paul Moore and Kees Cook, I have chosen to pursue a cBPF binary tree to improve performance for these customers. A cBPF binary tree requires no kernel changes and should be transparent for all libseccomp users.

I currently have a working prototype and the numbers are very promising. I modified gen_bpf.c and gen_pfc.c to utilize a cBPF binary tree if there are 16 or more syscalls being filtered. I then timed calling getppid() in a loop using one of my customer's seccomp filters. I ran this loop one million times and recorded the min, max, and mean times (in TSC ticks) to call getppid(). (I didn't disable interrupts, so the max time was often large.) I chose to report the minimum time because I feel it best represents the actual time to traverse the syscall.

```
Test Case                       minimum TSC ticks to make syscall
---------------------------------------------------------------
seccomp disabled                                              138
getppid() at the front of 306-syscall seccomp filter          256
getppid() in middle of 306-syscall seccomp filter             516
getppid() at the end of the 306-syscall filter               1942
getppid() in a binary tree                                    312
```

As shown in the table above, a binary tree can siginficantly improve syscall performance in the average and worst case scenario for these customers.

I agree to abide by the anti-harassment policy

Yes

Primary author: HROMATKA, Tom

Presenter: HROMATKA, Tom

Session Classification: Birds of a feather (BoF)

Track Classification: BOF