Problem Statement

tl;dr – Current seccomp filters are large and slow

- Containers/Userspace – large seccomp whitelists
- Kernel and libseccomp – preference toward smaller filters
- Libseccomp currently – sequential “if equal” statements

```c
if ($syscall == 42)
  action ALLOW;
```

- Default docker filter – 300+ cBPF instructions to process read() syscall
Example – Docker’s Default Libseccomp Filter
getppid() performance in a large filter

At the front of the filter

At the end of the filter
Proposed Solution – Binary Tree

• Libseccomp shall generate a cBPF binary tree for large filters
  
  ```
  if ($syscall > 100)
    if ($syscall > 150)
      if ($syscall == __NR_prctl)
        action ALLOW;
      if ($syscall == __NR_sysctl)
        action ALLOW;
    else # $syscall <= 150
      # other syscall ifs
  else # $syscall <= 100
    # other syscall ifs
  #default action
  action KILL;
  ```

• All syscalls in the filter can now be evaluated in 13 or less cBPF instructions!
Performance Comparison for a 300-Syscall Filter

![Graph showing time to execute getppid() syscall for different cases](image)

- **Current - best case**: 1120 TSC Ticks
- **Binary tree - best case**: 1470 TSC Ticks
- **Binary tree - worst case**: 1520 TSC Ticks
- **Current - worst case**: 22144 TSC Ticks