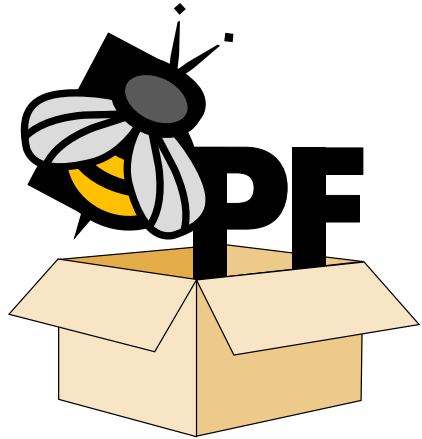
bpfbox: Simple Precise Process Confinement with eBPF and KRSI

William Findlay

October 28, 2020

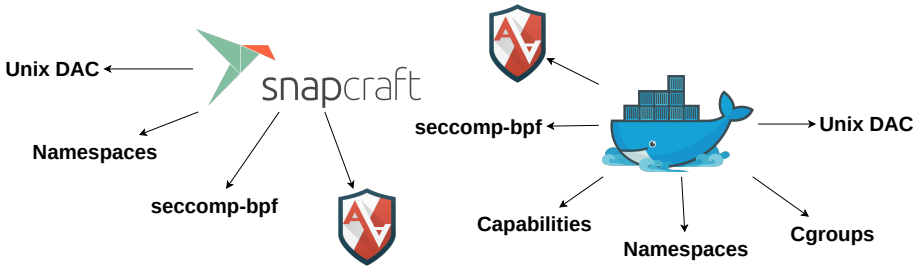
bpfbox at a Glance

- bpfbox is a novel **process confinement mechanism** for Linux using eBPF
- Users write per-application policy in a **simple policy language**
- Policy is enforced by attaching **BPF programs** to **LSM hooks**
  - Integrates userspace and kernelspace state into policy decisions



Motivation

- Existing process confinement mechanisms are **complex**



- Existing process confinement mechanisms are **difficult to use**


SELinux


AppArmor


TOMOYO

- Can we do any better?

eBPF Changes the Game

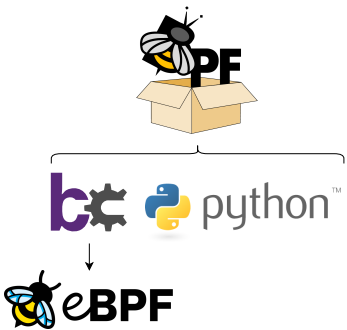
eBPF enables:

- Fine-grained system introspection
- Integration of **cross-layer state** (kprobes, uprobes, etc.) with policy enforcement (LSM probes)
- Rapid prototyping
- Safe production deployment of new security solutions

We have an opportunity to **rethink process confinement** from the ground up.

bpfbox Implementation

- Userspace daemon using the Python3 bcc framework
- Kernelspace components are all eBPF
  - LSM probes (KRSI), kprobes, uprobes, tracepoints
  - Under 2000 source lines of kernelspace code
- Thanks to eBPF, bpfbox is **light-weight**, **flexible**, and **production-safe**
  - Works out of the box on any vanilla Linux kernel ≥ 5.8



Our Policy Language

Rules and Directives

Rules specify access to system objects:

- `fs(file, access)`
- `net(socket, access)`
- `signal(prog, sig)`
- etc.

Directives augment blocks of rules:

- `#[directive]` syntax
- Specify **actions to be taken** on a block of rules
- Add **additional context** to a block of rules

Our Policy Language

Policy at the Function Call Level

- `#[func "foo"]` → Apply rules only within a call to `foo()`
- `#[kfunc "foo"]` → Same thing, but for kernel functions

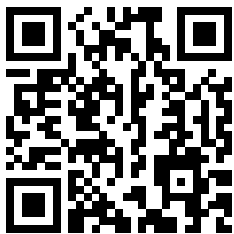
```
#![profile "/sbin/mylogin"]

#[func "check_password"]
#[allow] {
    fs("/etc/passwd", read)
    fs("/etc/shadow", read)
}

#[func "add_user"]
#[allow] {
    fs("/etc/passwd", read|append)
    fs("/etc/shadow", read|append)
}

/* ... */
```

Acknowledgements

Special thanks to:

- **Alexei Starovoitov** and **Daniel Borkmann** (creators of eBPF)
- **K.P. Singh** (creator of KRSI)
- Fellow **bcc contributors** (an awesome eBPF framework)

This work was supported by NSERC through a Discovery Grant.



github.com/willfindlay/bpfbox

Check out the project on GitHub!