# bpfbox: Simple Precise Process Confinement in eBPF

**William Findlay**[1] Anil Somayaji David Barrera

[1]will@ccsl.carleton.ca

October 17, 2020



Carleton
UNIVERSITY

# Outline of Talk

**Motivation**

**Architecture**

**Policy**

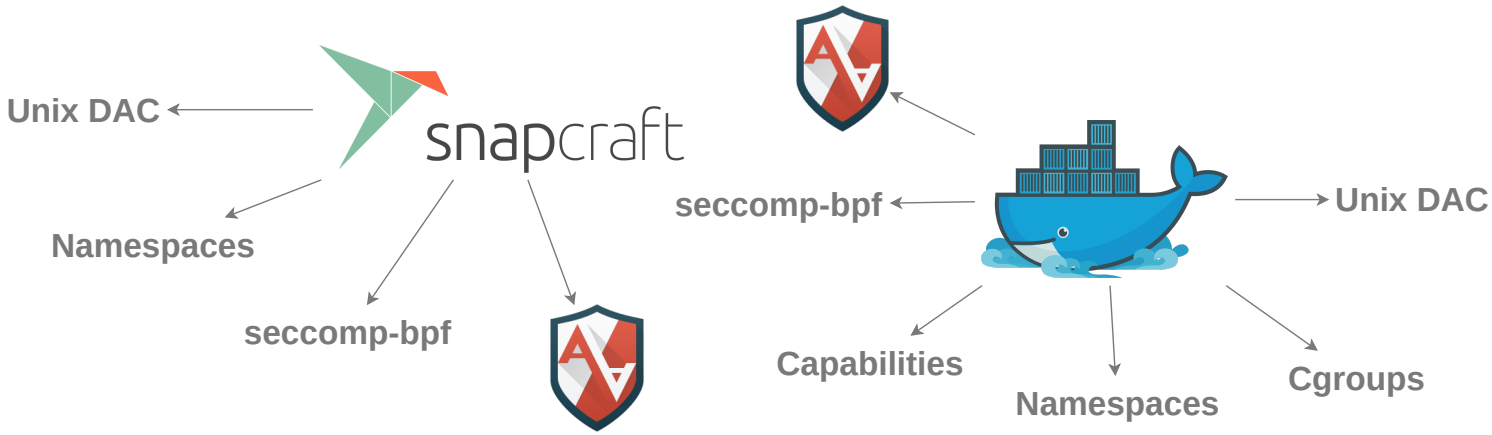
**Performance**

**Conclusion**

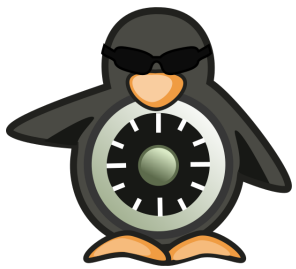
# Motivation

# The Status Quo

- Existing process confinement mechanisms are **complex**



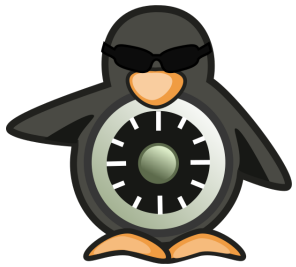

- Existing process confinement mechanisms are **difficult to use**



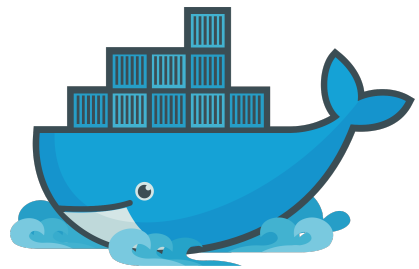
- Can we do any better?

# Stakeholders as Policy Authors

- **Security experts** define the policy



- **Application authors** and **packagers** define the policy



- **End users** define the policy

???

# eBPF Changes the Game

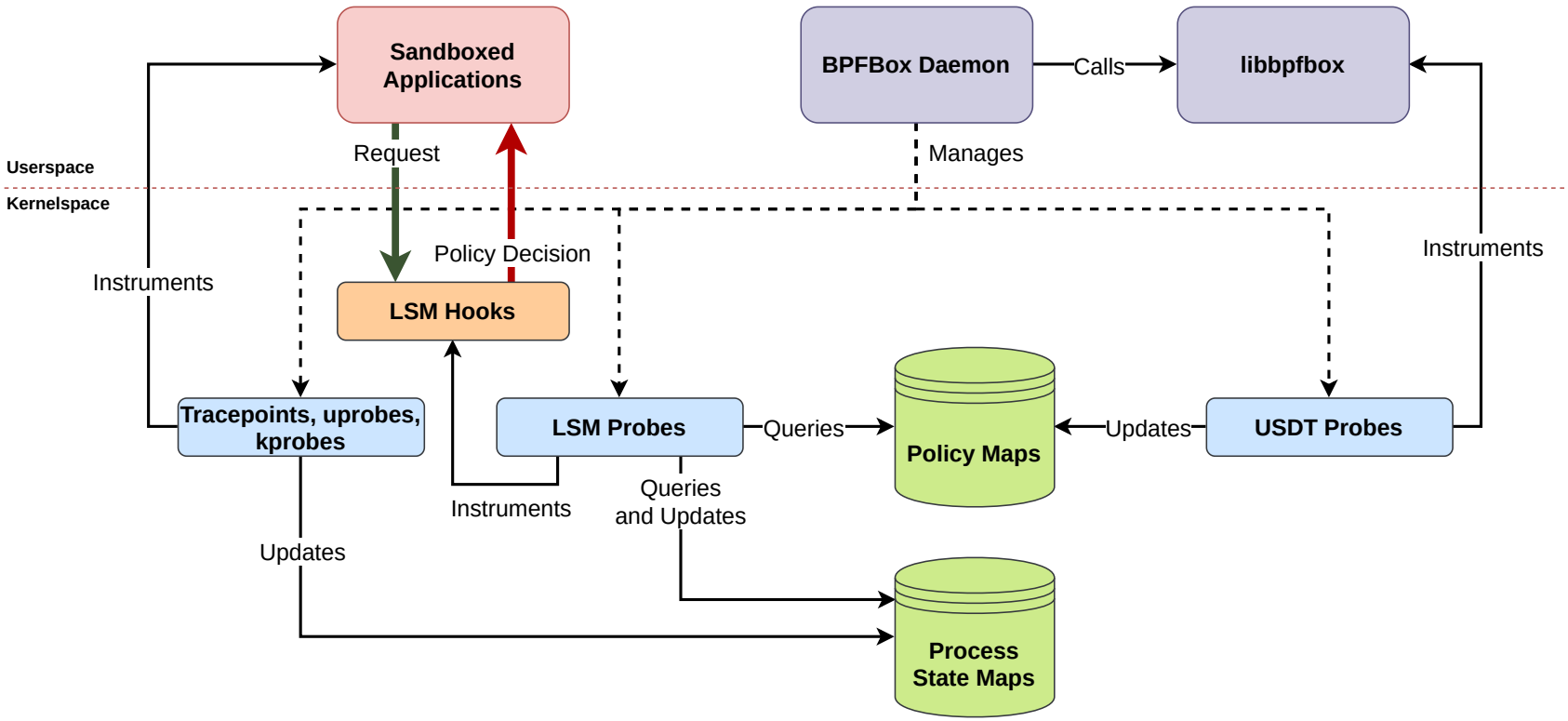
TODO

# Architecture

# bpfbox Architecture

- TODO: Python3 bcc
- TODO: KRSI
- TODO: Lines of userspace code
- TODO: Lines of kernelspace code
- TODO: Compare w/ SELinux, AppArmor

# bpfbox Architecture



# Policy

# bpfbox Policy

- Write bpfbox policy at the **function call** level:

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

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

#[func add_user]
#[allow] {
    append("/etc/passwd")
    append("/etc/shadow")
}
```

# Performance

# Performance

TODO

# Conclusion

# Acknowledgements

TODO

# Contributions

- First full policy enforcement engine written in eBPF
- Integration of userspace and kernelspace state with LSM layer enforcement