ioctl command whitelisting in SELinux

Jeff Vander Stoep

08/21/2015

Acknowledgements

Stephen Smalley

Nick Kralevich

Dan Cashman

Mark Salyzyn

Paul Moore

Rom Lemarchand

NAME:

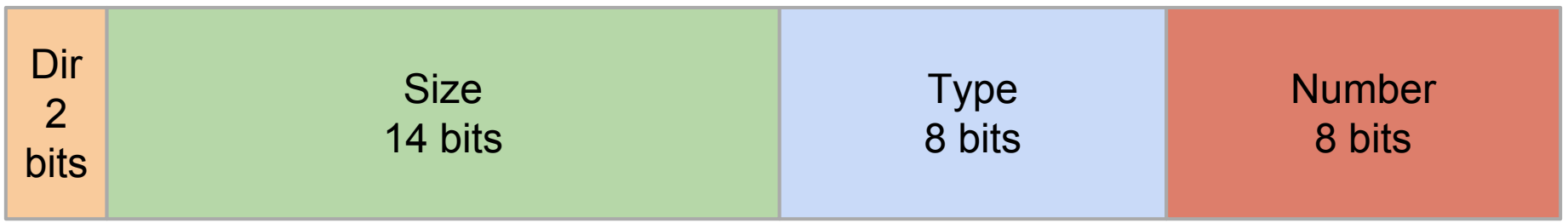
int ioctl(int filed, int command, ...);

CONFORMING TO:

No single standard. Arguments, returns, and semantics of ioctl() vary according to the device driver in question (the call is used as a catch-all for operations that don't cleanly fit the UNIX stream I/O model).

ioctl(2)

loctl command



Motivation

- Protect user privacy - Limit access to persistent device identifiers
  - E.g. MAC address can be used by apps to fingerprint a device. Used to create an in-app DRM, licensing, etc
- Protect the kernel - Reduce attack surface.
  - Limit access to driver i/o. - e.g. GPU
  - Limit leaking of information - e.g. kernel pointers.

[...] the security of an SELinux system depends primarily on the ***correctness of the kernel and its security-policy configuration****.*

http://en.wikipedia.org/wiki/Security-Enhanced_Linux

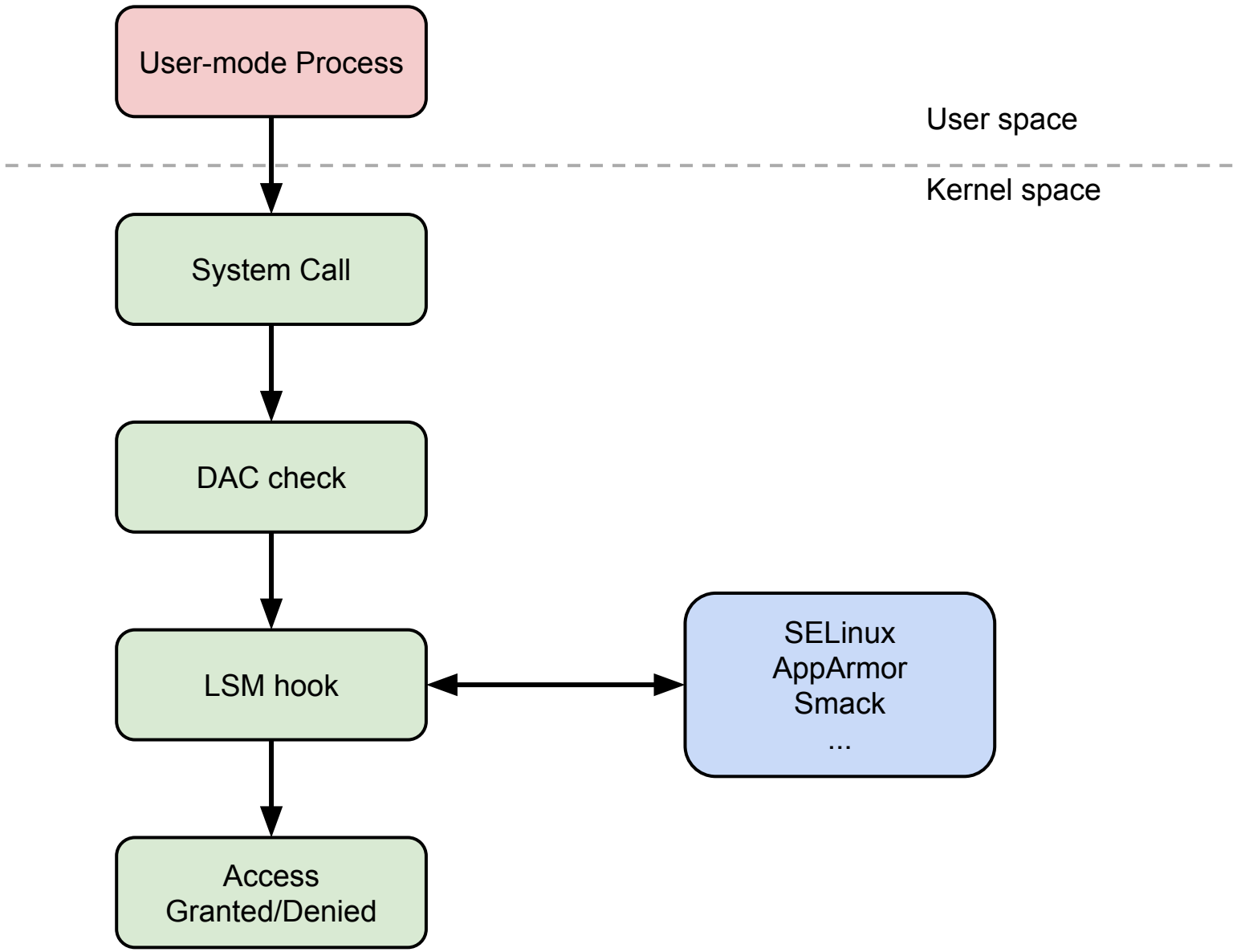
Some numbers

Kernel crash analysis - ~500 kernel crashes across multiple types of devices

~45% of crashes happened in a system call

~15% of crashes happened in an ioctl call

Linux Security Module



Why use SELinux?

Selinux and system operations

- chown
- kill
- setuid
- ipc_lock
- mmap
- DAC override
- mknod
- ...

capable(CAP_ CHOWN)

SELinux and ioctls

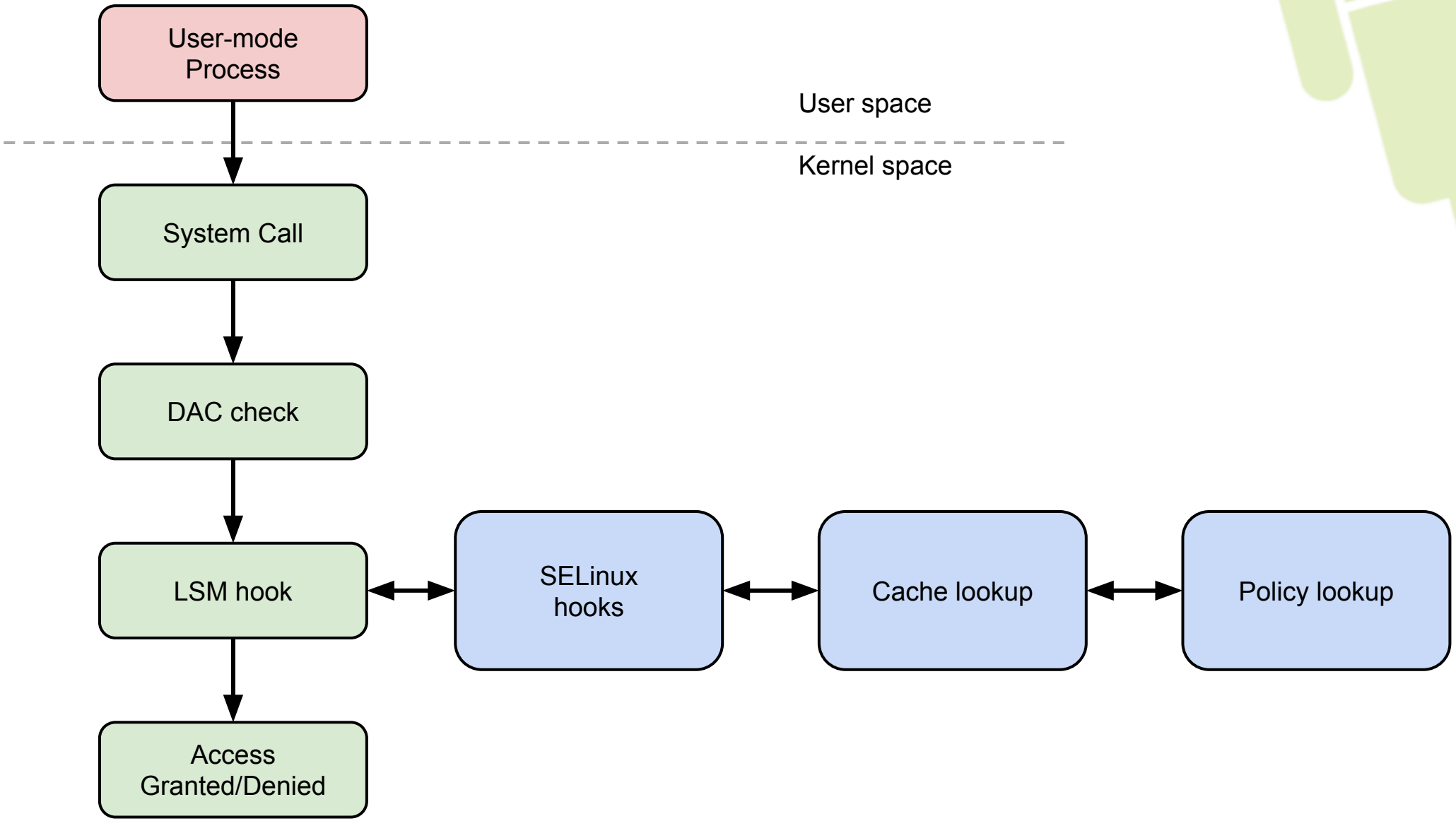
- Benign functionality
  - driver version
  - socket type
  - ...

- Dangerous functionality
  - debugging capabilities
  - read/write/execute to physical memory
  - privacy sensitive data
  - information leaks

Constraints

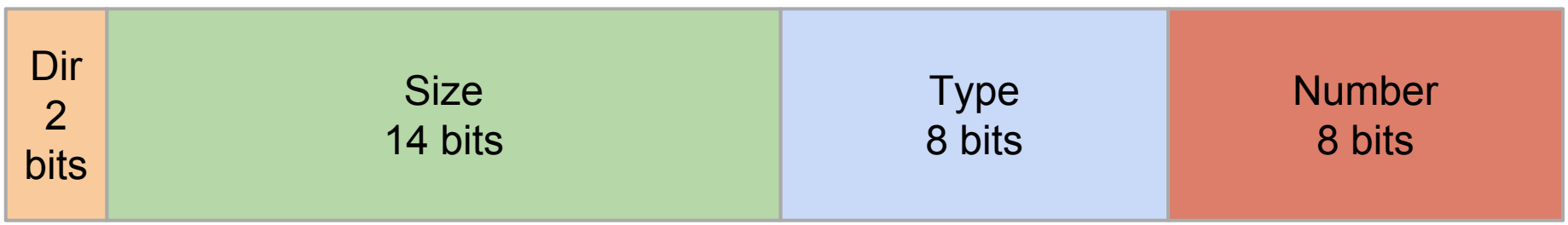
- Performance:
  - many ioctls are performance sensitive e.g. network and graphics
  - thousands of ioctl calls per second. ~150000 ioctl calls during device boot.
- Targeted whitelisting
  - support existing policy.
- Optimize for ioctls with a large command set
  - small command sets adequately protected with existing ioctl command.

SELinux Architecture



Architecture

- Only examine ioctl type and number. Size and direction are considered to be arguments
  - allowxperm <source> <target>:<class> ioctl unpriv_app_socket_cmds
  - auditallowxperm <source> <target>:<class> ioctl priv_gpu_cmds
- Use information regarding ioctl distribution to create a constant permission check time
  - Commands are grouped by type, so cache commands by type

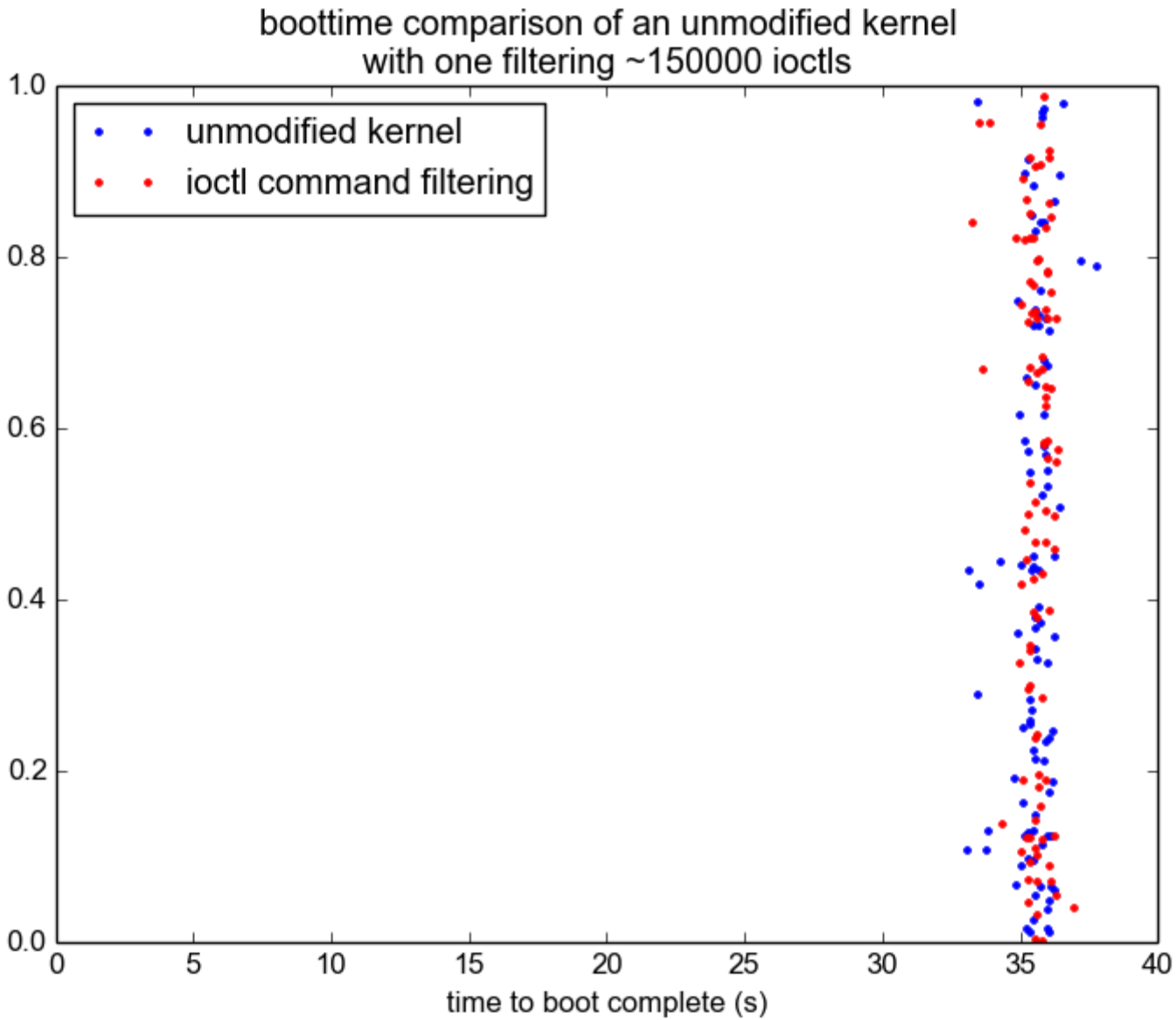


Extended Permissions

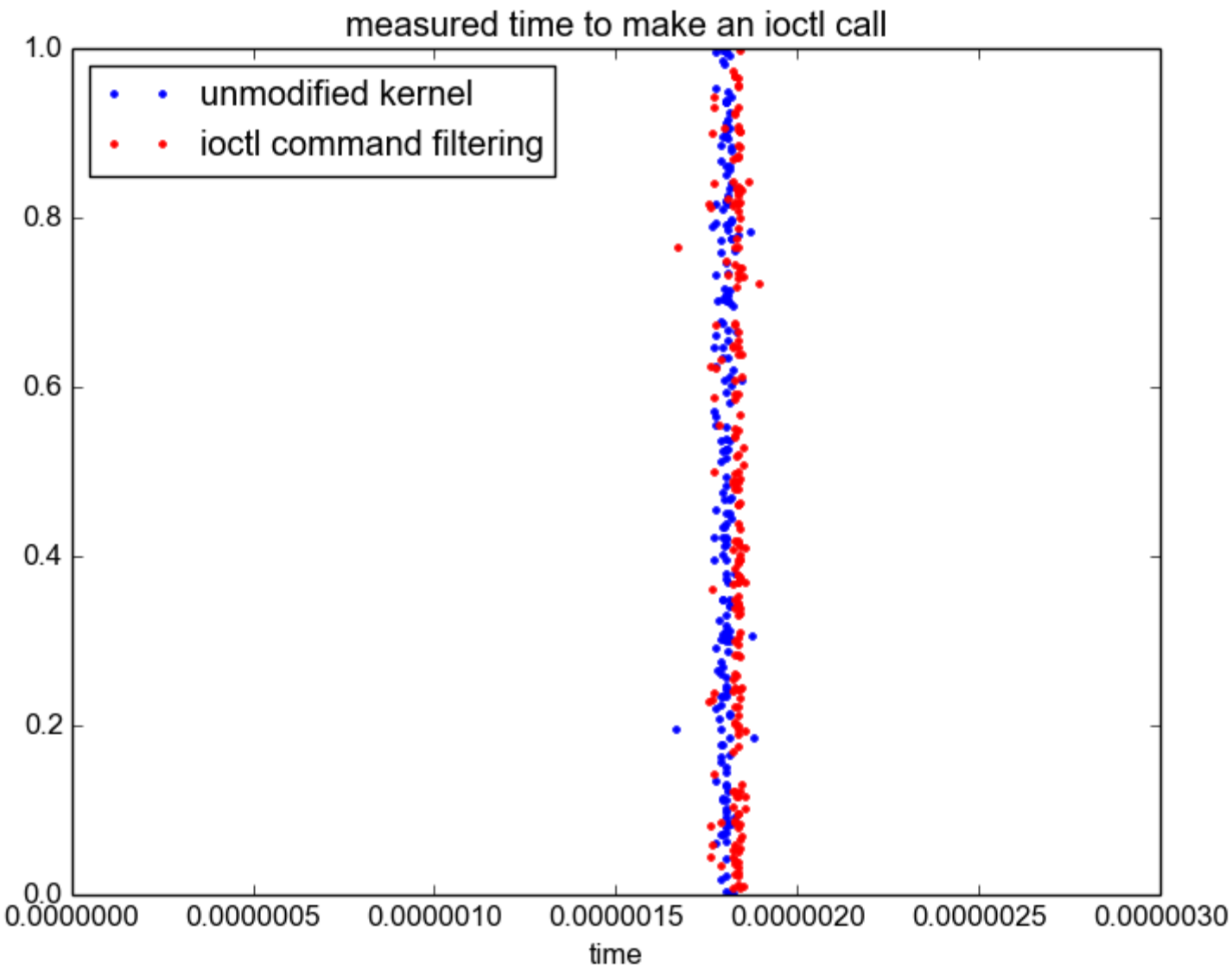
- Provide additional permissions in the Access Vector Cache (AVC).
  - In increments of 256 bits

```
struct avc_entry {
        u32                     ssid;
        u32                     tsid;
        u16                     tclass;
        struct av_decision      avd;
+        struct avc_xperms_node  *xp_node;
 };
```

Boot performance: 150000 ioctl calls



Individual ioctl calls



Case Study

Blocking third party app access to MAC address

Fuzzing the GPU

Questions?