BPF Access Control and CO-RE in Android

Neill Kapron <nkapron@google.com>

Android BPF Goals

- Enable Modern BPF functionality
  - CO-RE
  - Libbpf helpers
- Enable secure vendor access to BPF tracepoints
- Build solid foundation for future use cases

Goals & Requirements

BPF Overview

Libbpf & CO-RE

Current Implementation

CO-RE + Access Control

Questions

Android Requirements

- Minimal impact to boot time
- Cognitive use of memory
- Control access of BPF program attach points

Goals & Requirements

BPF Overview

Libbpf & CO-RE

Current Implementation

CO-RE + Access Control

Questions

High Level BPF Overview

Goals & Requirements

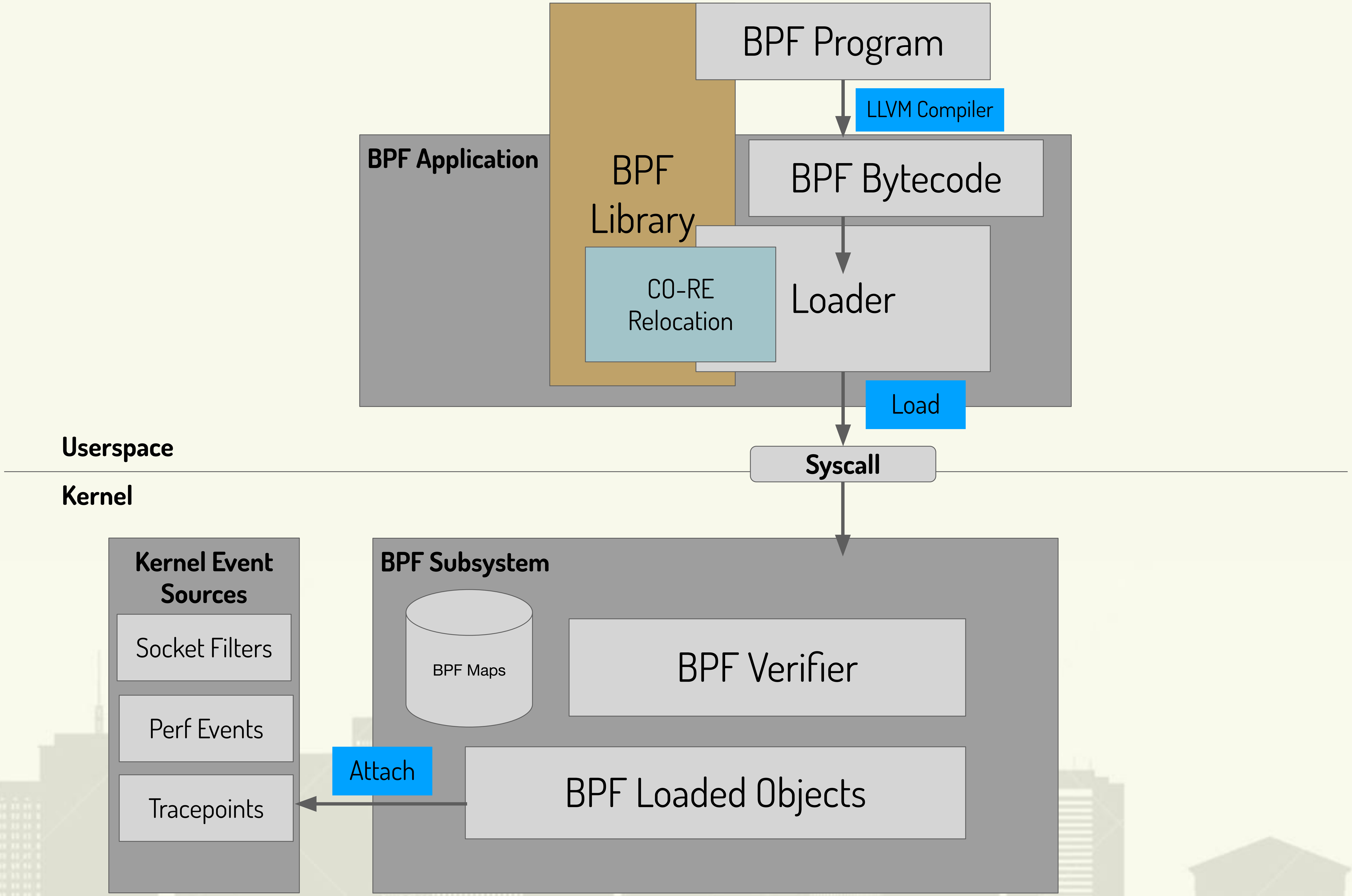
BPF Overview

Libbpf & CO-RE

Current Implementation

CO-RE + Access Control

Questions



Libbpf

- Primary userspace library used with BPF
- Provides helper functions to ease BPF application development
  - Loading BPF programs into kernel
  - Attaching loaded programs to kernel event sources
  - Map Creation & Access
  - Data Access
  - CO-RE Implementation
- Maintained as part of the kernel tree

Goals & Requirements
BPF Overview
Libbpf & CO-RE
Current Implementation
CO-RE + Access Control
Questions

CO-RE

- Compile Once - Run Everywhere
- Attempts to solve BPF program portability issues
- Implemented in userspace libraries
- Access structures which may have changed between kernel versions
- Marks fields as relocatable at compile time
- Uses running kernel's BTF info to relocate field offsets prior to loading BPF program into kernel
- Does not address the case where meaning of struct field changes

Current Implementation

- Single point of access for BPF
- BPFloader application in early init process
- Single threaded loading of all BPF programs
- Based on custom library to handle BPF syscalls
- Missing much of the modern BPF functionality

Goals & Requirements

BPF Overview

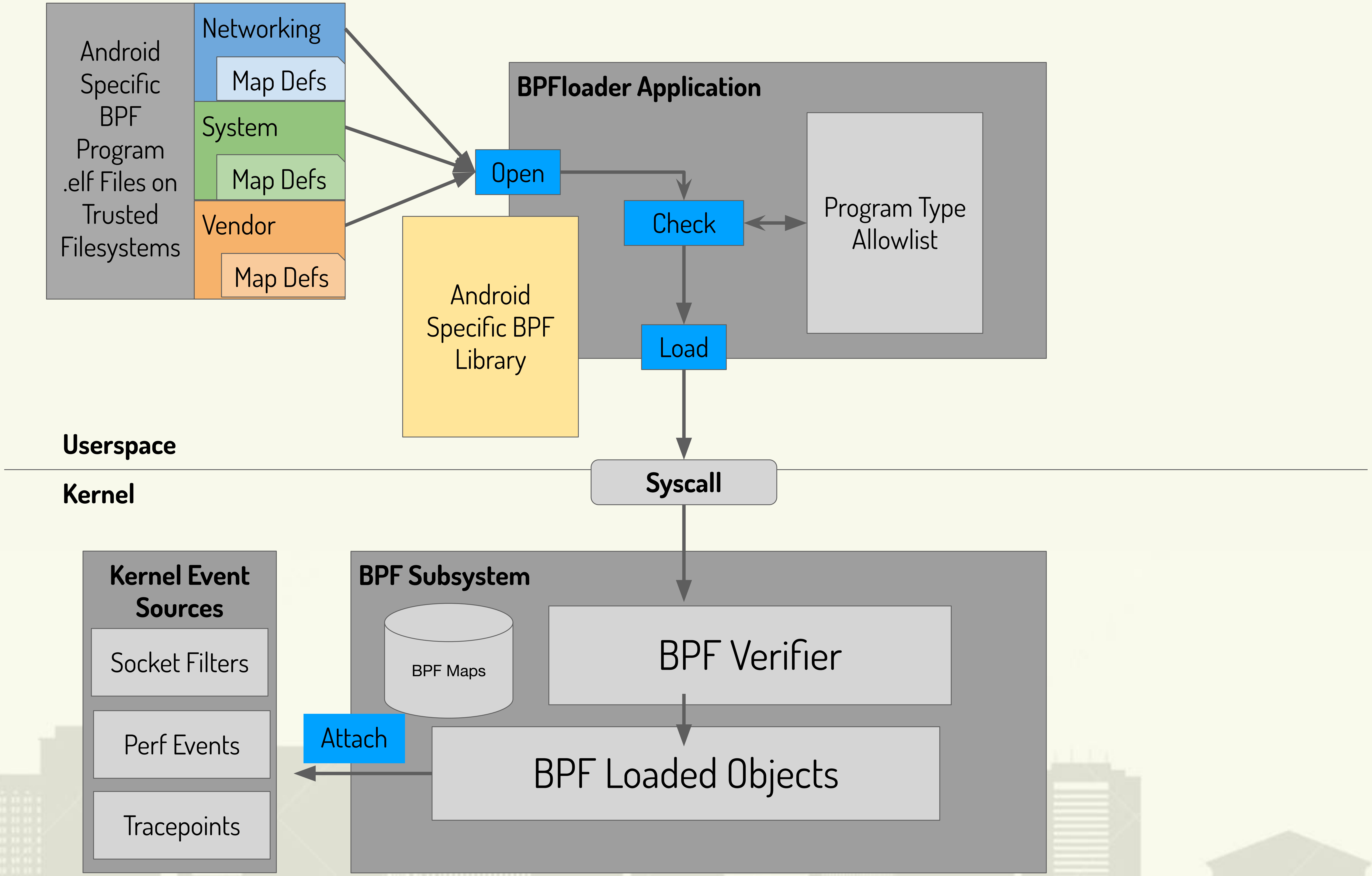
Libbpf & CO-RE

Current Implementation

CO-RE + Access Control

Questions

Android's BPFloader



Goals & Requirements

BPF Overview

Libbpf & CO-RE

Current Implementation

CO-RE + Access Control

Questions

Supported Program Types

Goals & Requirements

BPF Overview

Libbpf & CO-RE

Current Implementation

CO-RE + Access Control

Questions

BPF Program Types	Networking	System	Vendor
CGROUP_SKB	Y		
CGROUP_SOCK	Y		
CGROUP_SOCK_ADDR	Y		
KPROBE		Y	R
PERF_EVENT			R
SCHED_ACT	Y		
SCHED_CLS	Y		
SOCKET_FILTER	Y	Y	Y
TRACEPOINT		Y	R
XDP	Y		

Y = Supported, R = Requested

Possibilities for Enabling CO-RE in Android

- Implement custom Android-Specific library
  - Constant development and maintenance required as new BPF features are created
  - Can optimize for our specific use case
- Integrate Libbpf into existing bpfloader
  - Does not solve boot time problem
  - Potential for significant increase in memory usage
  - Potential problems with compatibility between vendor BPF programs and system libbpf library version
- Enable BPF programs to use libbpf natively
  - Allows developers and vendors to choose when their programs are loaded
  - Resolves compatibility issue between system libraries and vendor bpf programs
  - Requires additional work to develop access control mechanism
- Other approaches?

Attach Point Access Control

- **Need to verify that tracepoints are part of KMI before attaching**
  - KMI varies between kernel branches
  - KMI additions can occur post kernel release (requires allowlist updatability)
- Could be accomplished via allowlist in bpfloader
  - Check bpf program's attach points prior to loading into kernel
  - Allowlist must be dynamic and maintain support for all kernel versions
- BPF Program/Kernel Module based access control
  - Hook into the kernel's bpf_prog_load(), bpf_prog_attach() functions
  - First BPF program loaded as part of boot
  - Check subsequent bpf progs against running kernel's KMI
  - Enables the control of 'native' libbpf programs
  - Unknown- how to check source of bpf program in kernel?

Goals & Requirements

BPF Overview

Libbpf & CO-RE

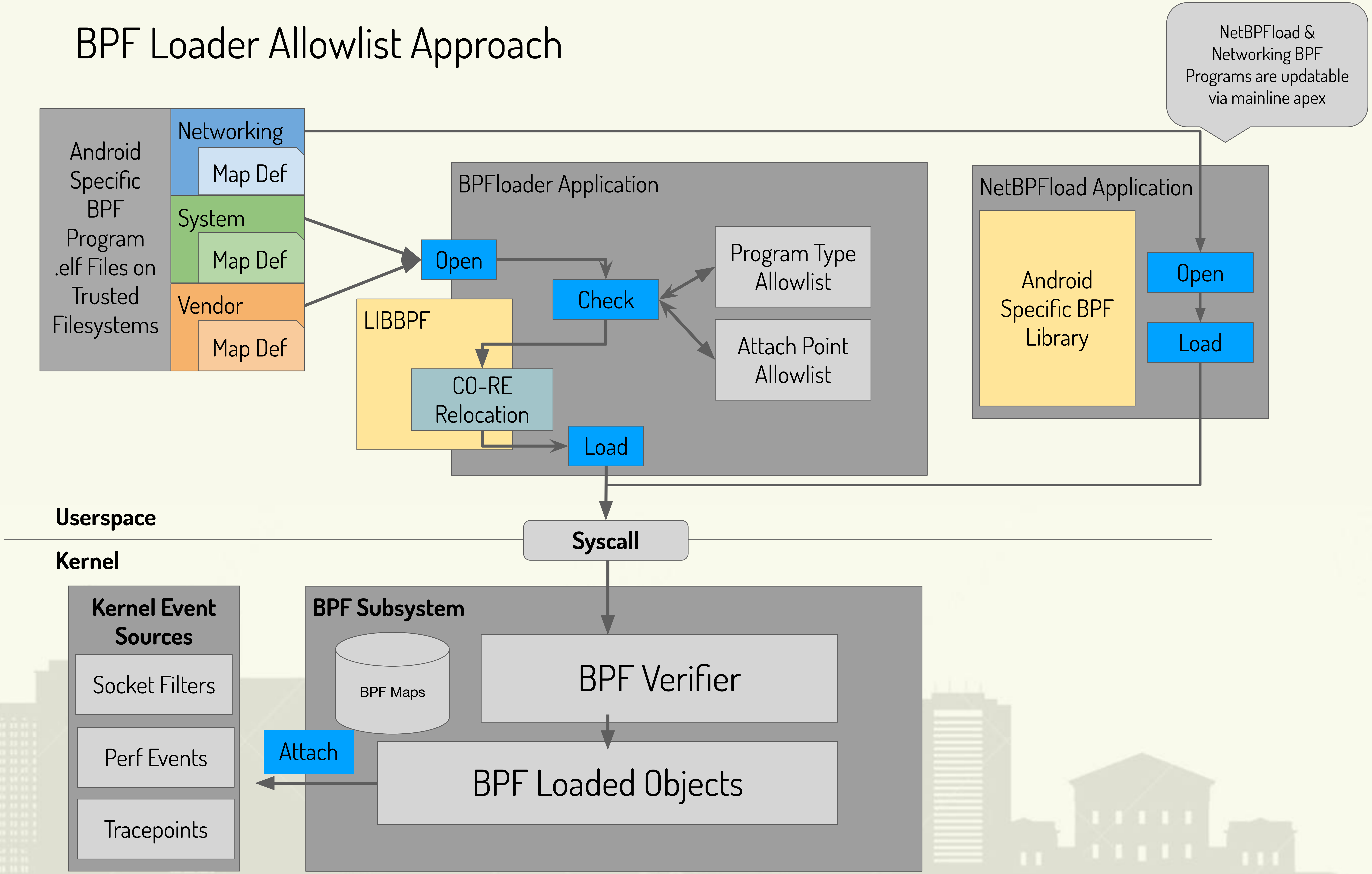
Current Implementation

CO-RE + Access Control

Questions

Attach Point Access Control

BPF Loader Allowlist Approach



Goals & Requirements

BPF Overview

Libbpf & CO-RE

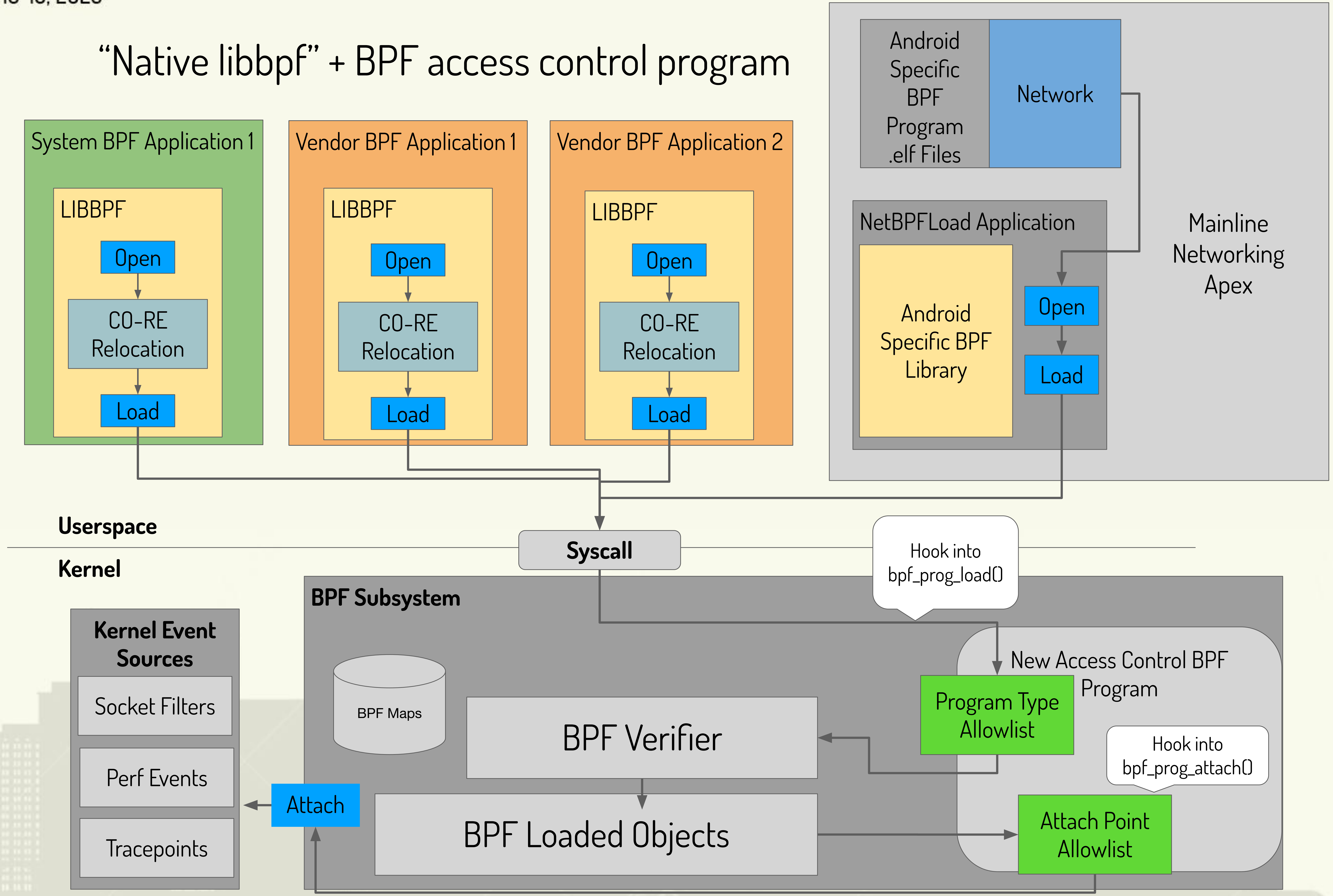
Current Implementation

CO-RE + Access Control

Questions

Attach Point Access Control

"Native libbpf" + BPF access control program



Goals & Requirements

BPF Overview

Libbpf & CO-RE

Current Implementation

CO-RE + Access Control

Questions

BPFloader Open Questions

- What is the compatibility story for libbpf?
  - Do we need a trampoline library for future API changes?
- What can be done to optimize loading at boot time?
- Can system BTF data be cached by loader process?
  - Refactor libbpf calls to allow passing in BTF object
- How do we update ACL with KMI changes?
- Extending Metadata for selinux policy?

Goals & Requirements

BPF Overview

Libbpf & CO-RE

Current Implementation

CO-RE + Access Control

Questions

'Native Libbpf' Open Questions

- Will this approach pass security review?
- How do we get KMI ACL from kernel
  - Do we create a subset of KMI?
- How to pair BPF object with source for program type verification?
- What can be done to optimize BTF memory footprint?

General Questions

- Do we need to be able to update BPFloader/system BPF programs via apex?
- Is there another approach to consider?
- What policies are needed regarding vendor responsibilities?
- Are there other program types we should enable?

Goals & Requirements

BPF Overview

Libbpf & CO-RE

Current Implementation

CO-RE + Access Control

Questions

Thank You!

Neill Kapron <nkapron@google.com>