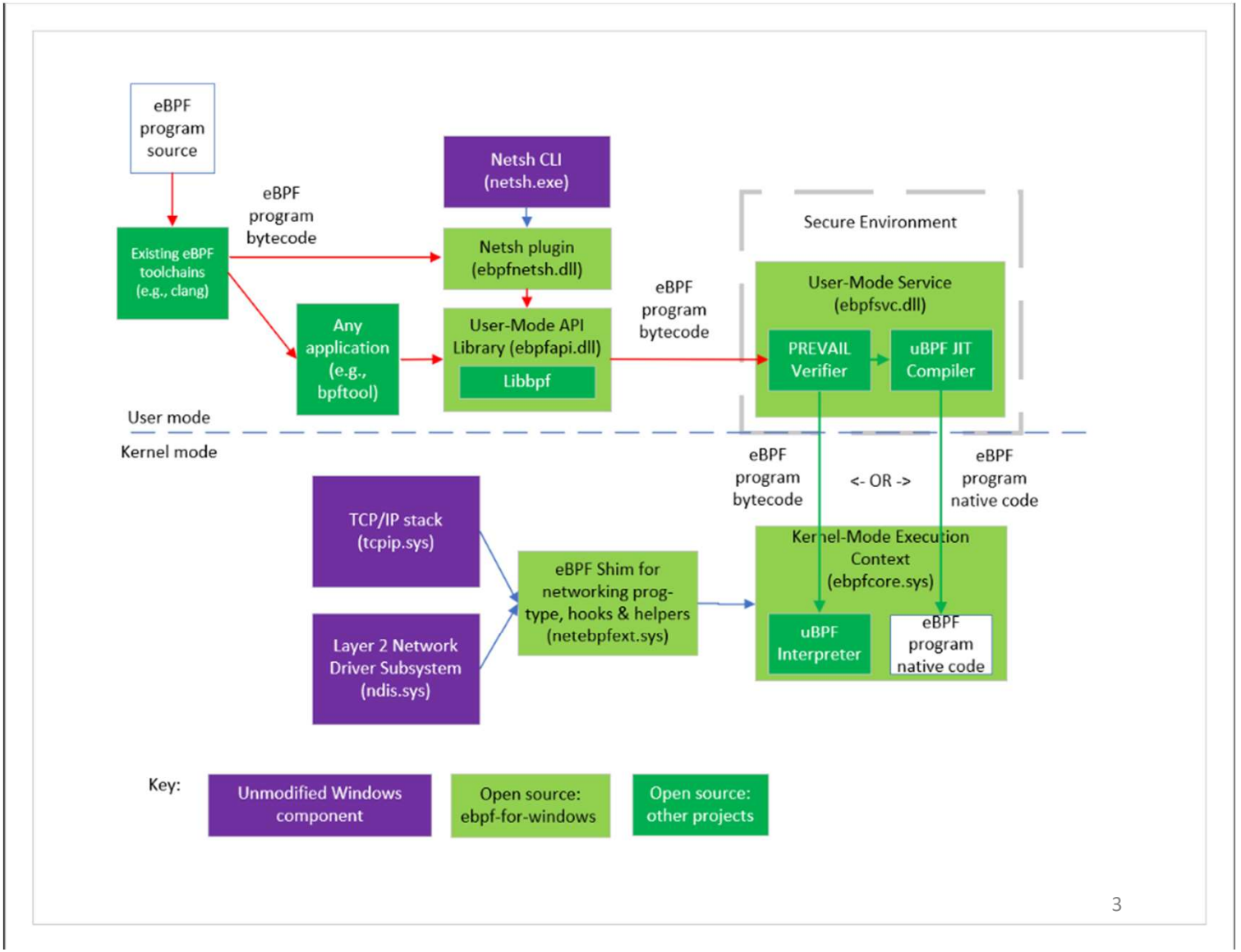
eBPF For Windows

Open Source – MIT License

- Please join us at our weekly triage meetings
  - We actively encourage external contributions
    - Weekly triage meetings open to all: Monday's at 8:30am PST. Please attend and contribute!
    - Code, design and feature requests.
  - We are inclusive by design and our team includes:
    - Microsoft folks who know Windows as well as researchers and Industry experts
- Re-use existing projects where we can
  - Prevail Verifier - vbpf/ebpf-verifier: A new eBPF verifier, using abstract interpretation (github.com)
  - uBPF - iovisor/ubpf: Userspace eBPF VM (github.com)
- Only build what we need to
  - Windows specific components

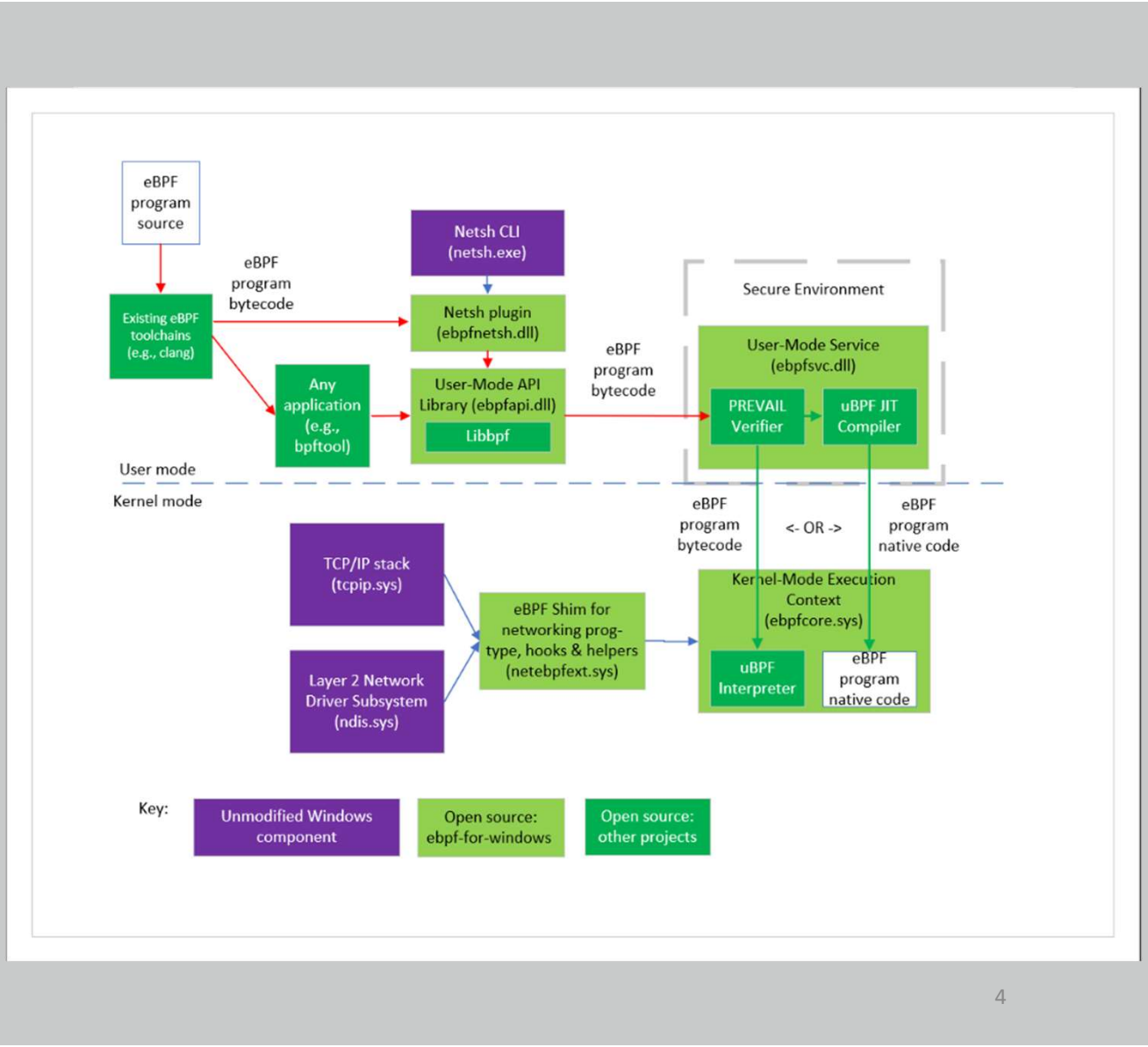
Architecture

- Open-sourced verifier and jitter
- Open-sourced infrastructure for running eBPF programs on top of Windows
- eBPF ISA version 1
- Network attach points for XDP and bind() based on Windows API (more underway)
- Concepts such as CGROUPS, TC on Linux work differently to Windows
- Only looking at cross-platform permissive licensed code



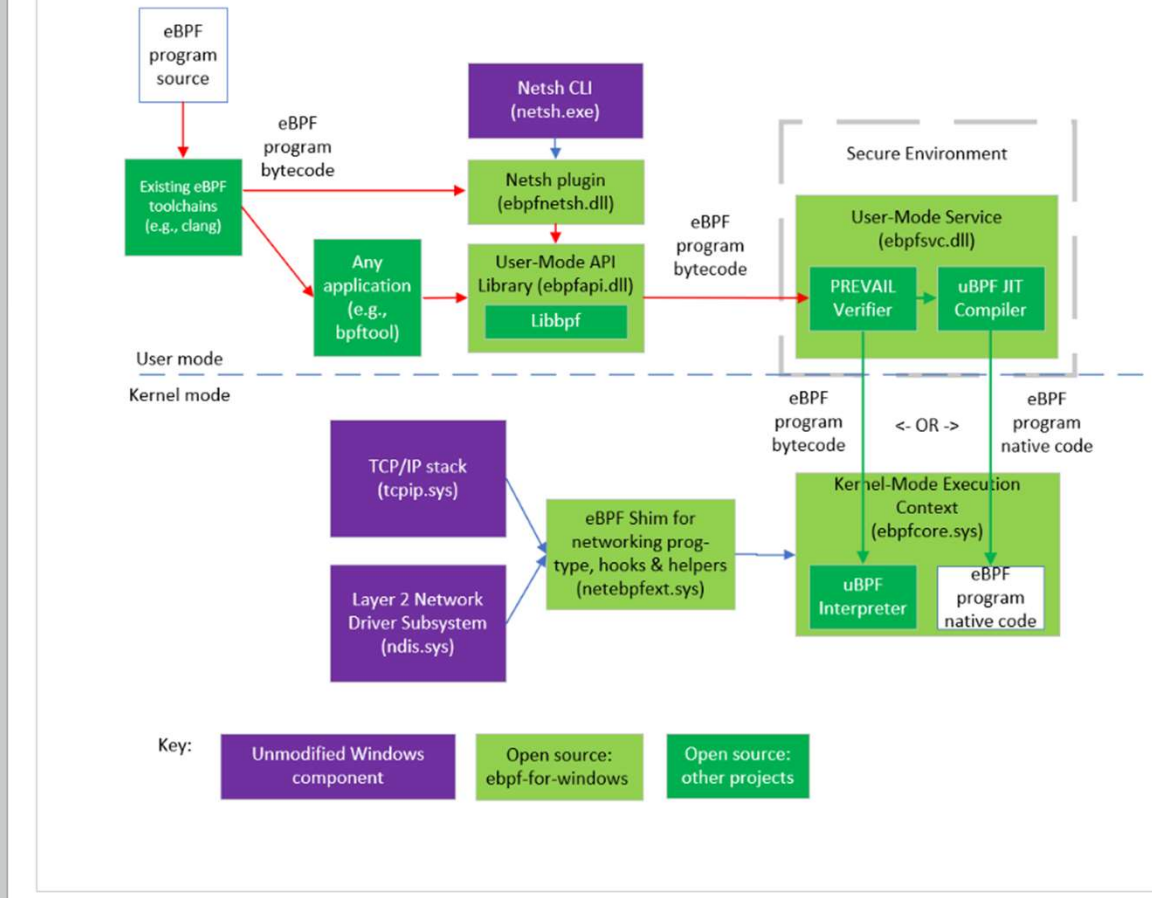
Architecture

- Decouple eBPF building blocks from the OS kernel
  - Aim to allow new attach points without change in the kernel
  - Separate verifier and jitter from the kernel
- Build a floating Execution context (EC)
  - Execution context is where the eBPF program runs
  - Execution context abstracted from platform
  - Runs in kernel mode
  - Also allows tests to run in user mode
- Program type, Helpers and hooks **defined by extensions**
  - Extension (helpers/hooks) developed independent of the EC or the verifier or the jitter.
  - Extensions register via Network Module Registrar (public API)
  - Allow serviceability and updatability of extensions
  - Attach points bound at runtime
  - Program hook/helper signatures dynamically verified



Architecture

- Extensions allow devs to define **and** implement eBPF program types and associated contracts
- Developers can abstract *one or more* OS APIs to define new (custom) hooks and helpers
- Allows scenario specific business logic to run as eBPF programs while extensions implement necessary OS contracts
- Allows eBPF programs to run on top of downlevel OS
- Detailed documentation at [docs/eBPFExtensions.md]
- Examples:
  - Windows Filtering Platform callout APIs could be turned into an extension
    - With bind layer for monitoring bind() operations
    - With connect layer for monitoring connect() operations
  - NDIS LWF callouts could provide XDP contracts
  - Sample at [tests/sample/ext]



What next?

- XDP support
  - Generic XDP as NDIS Lightweight Filter Driver layer
- More observability hooks and helpers based on public APIs
- Allow signing of eBPF programs; discussion #693
- Security hardening
- Up-to-date list publicly available on GitHub
  - Also discussed in the weekly triage meeting open to all

Demo

eBPF-for-windows demo with bind() hook · Discussion #706 · microsoft/ebpf-for-windows (github.com)

Q&A