DPI in the kernel with BPF

Peter Parkanyi

peter@redsift.io

DPI IN BPF

Motivation

- Server/desktop monitoring using the same software
- Easy deployment using a single static binary, no modules
- Detect DNS/TLS traffic on any port. (DNS packeting vs. nslookup)
- Containers
- Kubernetes
- No middleboxes in the cloud
- Multiple backends, including our own

DPI IN BPF

Solution

- Userspace agent in Rust
- Socket filters for TLS
- XDP for DNS
- Other BPF probes to monitor file access, generic traffic volumes
- Heavy use of perf events + epoll
- We can see which processes access which files, how much they read/write, and where they send how much data

DPI IN BPF

What we learned

- XDP only sees incoming traffic. In hindsight, this makes sense.
- Raw sockets impact performance
- Can't iterate maps means only detection, no active components
- linux/tools/bpf is not extremely helpful during development
- The fact that it compiles doesn't mean it will load (see above)
- Profiling probes is challenging
- 4096 insns & unrolled loops is a fun headspace
- Probes inserted in containers are wonky