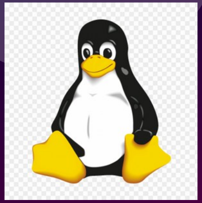
Beyond tcpdump – using eBPF and osquery for Linux Security Analytics



Southeast Linux Fest
June 7-9, 2024

David Hillman
Security Analyst at SoCo

[m] @dself2024:hobbysumo.com
[t] @dself2024
[mail] dself2024@hobbysumo.com

Agenda

- What is eBPF?
- What is osquery?
- What is observability?
- Why do we need any of this?
- Demos
- Questions:

What is eBPF?

State of the Linux world in 2013

- Linux kernel 3.18 considered “container ready”
- Original Berkeley Packet Filter design proved inadequate because filters are programs running on register-based machines. (it’s slow)
- Alexei Starovoitov introduces eBPF virtual machine design to take advantage of modern hardware
- eBPF proves 4x faster than original Berkeley Packet Filter design. This is due to just-in-time compilation and mapping to native instructions.

eBPF in detail

What

- In-kernel VM
- 64-bit JIT RISC
- Since kernel 3.15
- Not Turing complet

Why

- Security Monitoring
- Sandboxing
- Network filtering
- Process tracing

How

- Architecture
- LLVM and Clang programs
- Event-driven programming
- Plugins and Modules

What is osquery?

osquery in detail

What

- Developed in 2014
- OS Instrumentation
- SQL Tables represent OS info
- Extended using plugins

Why

- Host observability
- Configuration validation
- Random data extraction
- Troubleshooting

How

- Architecture
- osqueryd - daemon
- osqueryi - client
- Configuration

What is Observability?

- Observability is a modern way software development, support and security teams can discover problems in systems, ask fact-finding questions with data, pursue leads, and explore all aspects to solve those problems.

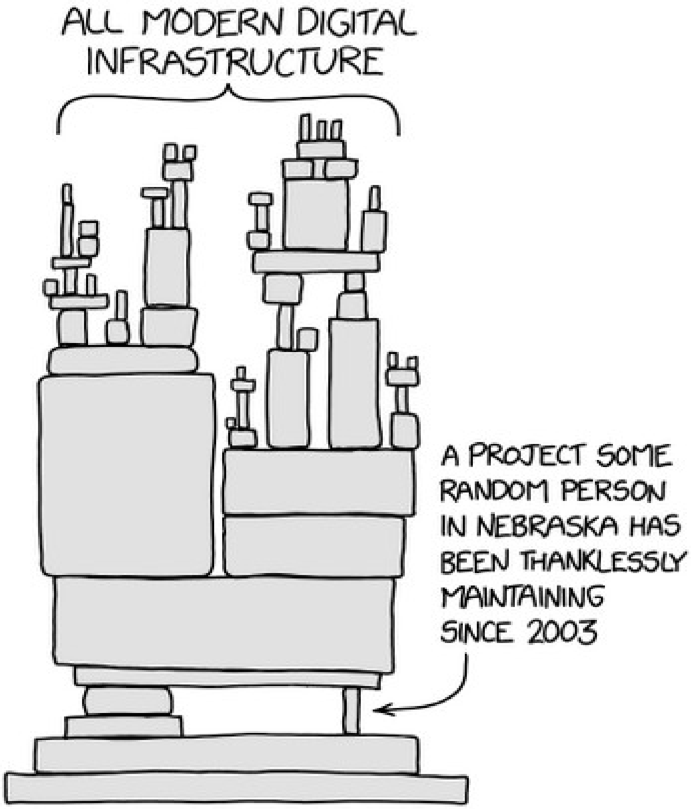
- Anything which can impact customer use of the system should be observable.

- Related to **control theory**, developed by Rudolf Kalman as part of control engineering.

eBPF, osquery and Observability

- eBPF is a game changer for observability because of deep insights into system behavior, performance and security.
- Osquery being a very high performance transport mechanism for traces, logs and events adds icing to the cake for observability.

Why do we need any of this?


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

State of the Linux world in 2024

- Perpetrators planned for over 24 months to gain trust and subvert the XZ project
- Andres Freund discovers the exploit by accident due to slow SSH operations
- XZ exploit is reported to CISA and given the highest CVSS score of 10

“Code was introduced, and it wasn’t easily apparent that it was attackable”
– Pete Allor, head of RedHat’s product security

“Code running as written is not the same as code running as designed”
– Raleigh Observer

Demos

Demo Architecture

Osquery and eBPF

- Osquery and eBPF together
- Plugin System
- Trailofbits ebpfpub

Detecting Badness

- Extracting some test events
- DNS profiling (getaddrinfo())
- SSH (XZ vulnerability)
- BlackLotus UEFI Malware (TPM 2)

Processing the Data

- FleetDM + Database
- Data Lake
- SIEM

Building Visualizations

- Flame Graphs
- Enclosure Diagrams

Questions