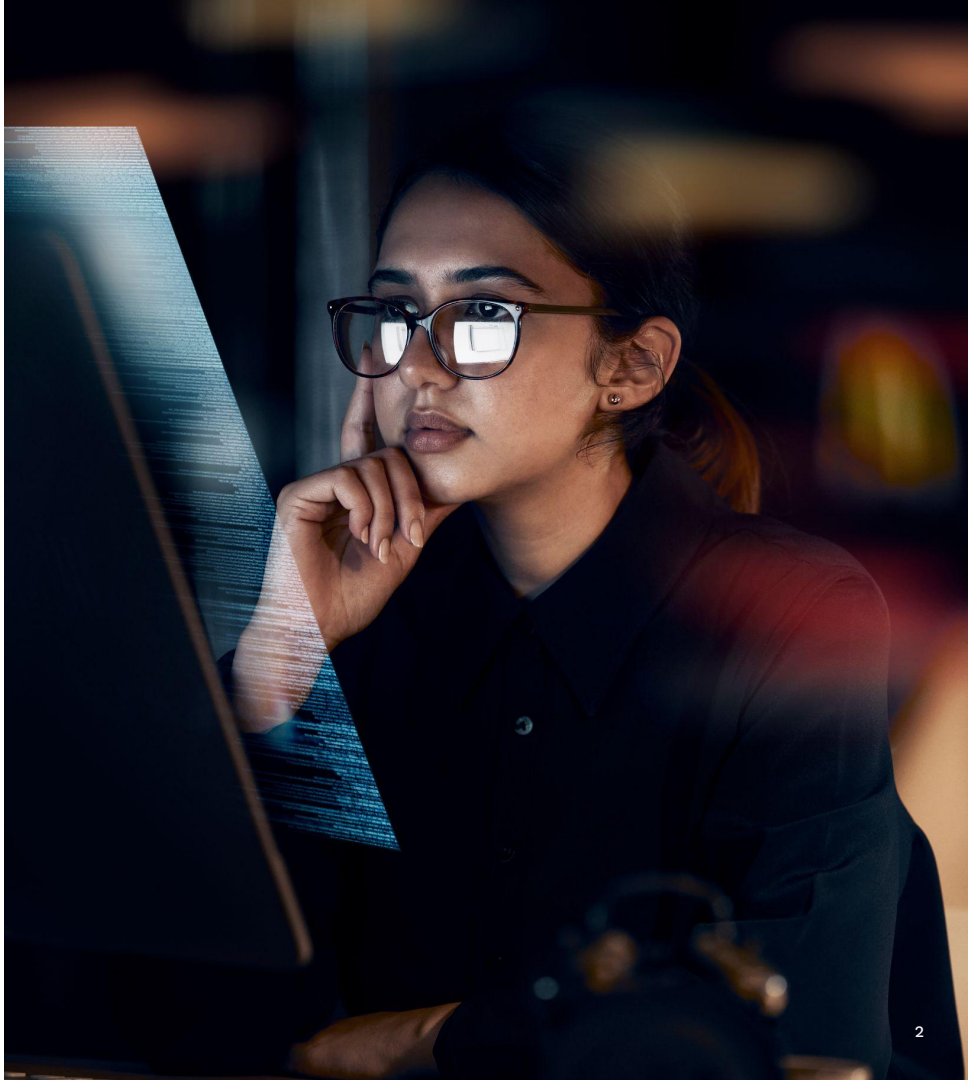
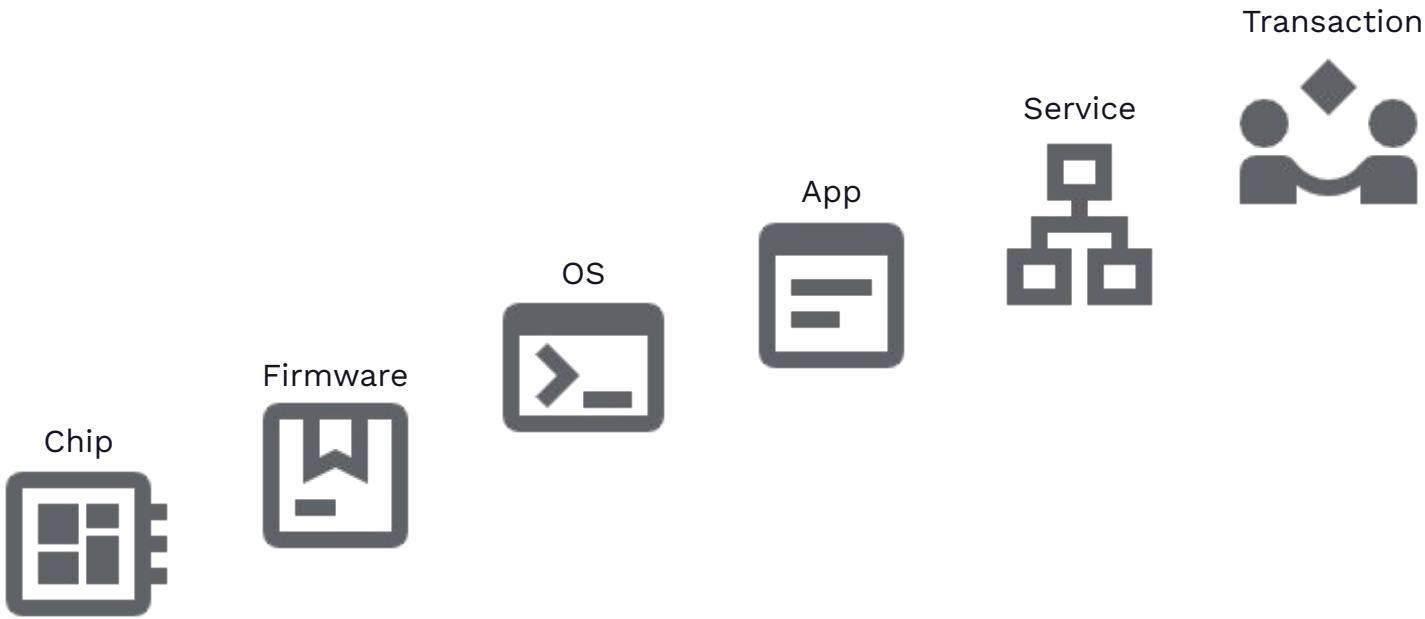
Invary

Runtime Integrity Measurement

Invary validates the Runtime Integrity of systems, verifying assumptions made about their security & confidentiality, while detecting threats that alter their intended behavior



Invary provides Runtime Integrity (Attestation) Solutions Centered Around the OS



Invary Background

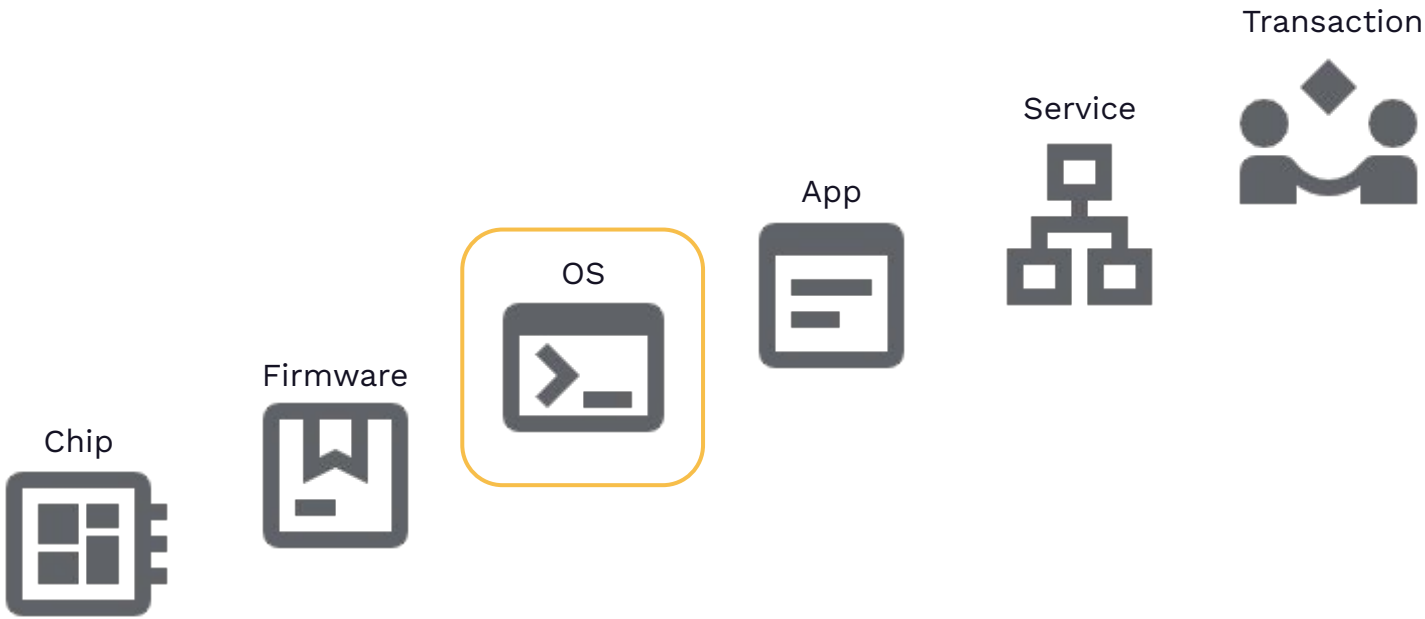
Invary exclusively licenses Runtime Integrity IP from the NSA's Laboratory for Advanced Cybersecurity Research

Licensed Patents: 7,904,278 & 8,326,579

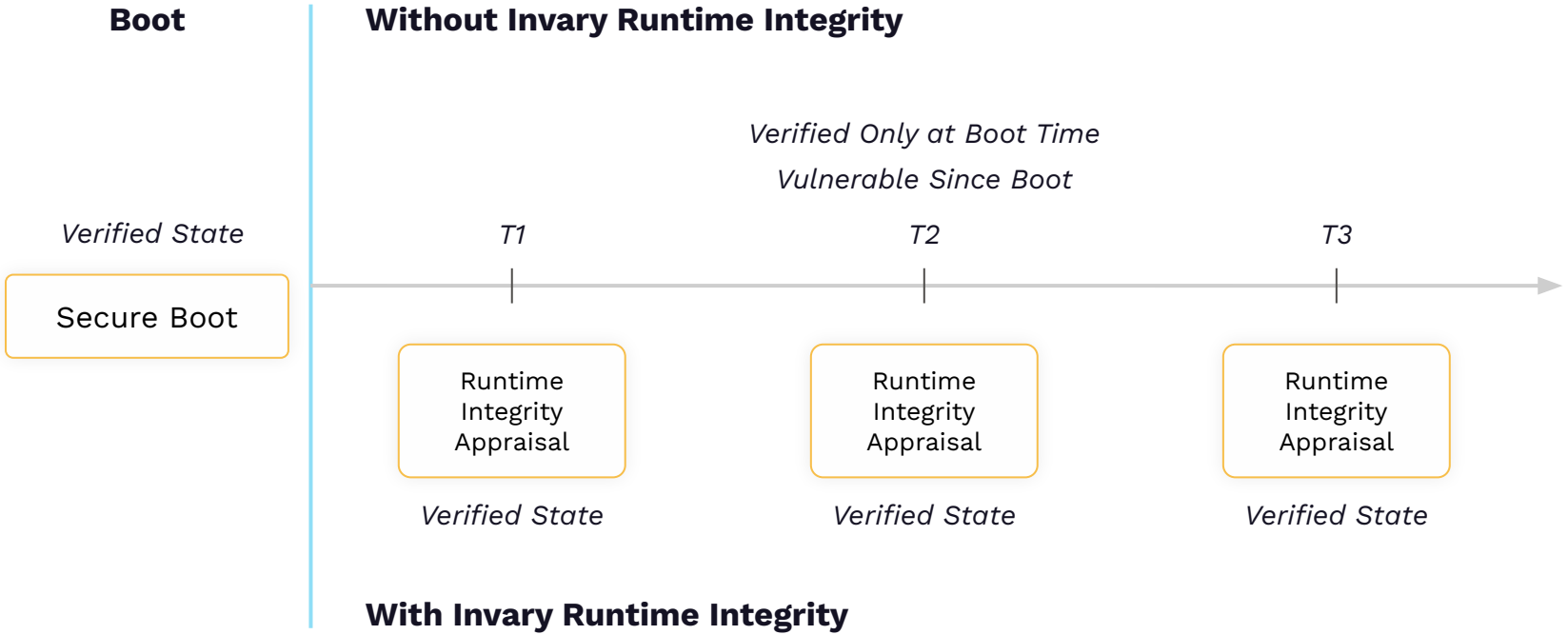
Invary performs Integrity Measurement research in collaboration with the NSA's LACR & the University of Kansas

Invary's team includes prominent Trusted System researchers, including our Founder Dr. Perry Alexander (publications), CTO Dr. Wesley Peck, and Advisor Peter Loscocco (NSA Cybersecurity Trust Mechanisms, retired)

Invary's solution serve a wide range of customers across the federal and commercial landscape. From General Dynamics to small businesses like KanREN



OS Runtime Integrity (Attestation)



Why Start With OS Runtime Integrity

- The OS is central to performing other attestations & security assertions
- Yet almost always the OS is assumed to be trusted and never verified, violating Zero Trust principals
- A compromised OS can deceive and falsify data necessary for other attestations, EDR/XDR, SIEM, CNAPP, SOAR, etc.. to function properly and be trustworthy

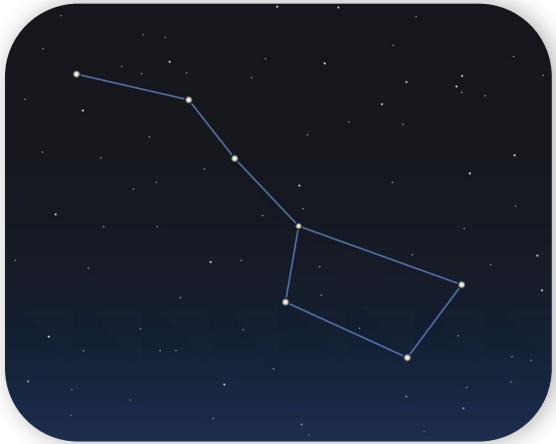
White Paper: How OS Runtime Integrity works in context of Drovorub, a rootkit disclosed by the NSA & FBI

How OS Runtime Integrity Works

Traditional approaches look through the entire haystack to find the needle

Invary Runtime Integrity knows a needle is present because the haystack's size or weight changed

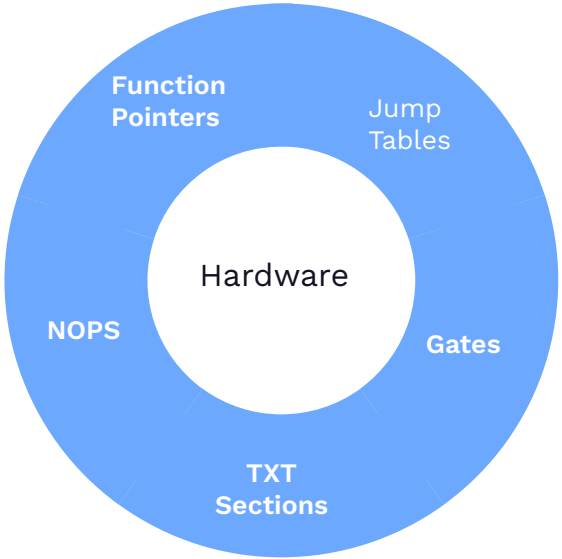
Lacks Integrity



Has Integrity

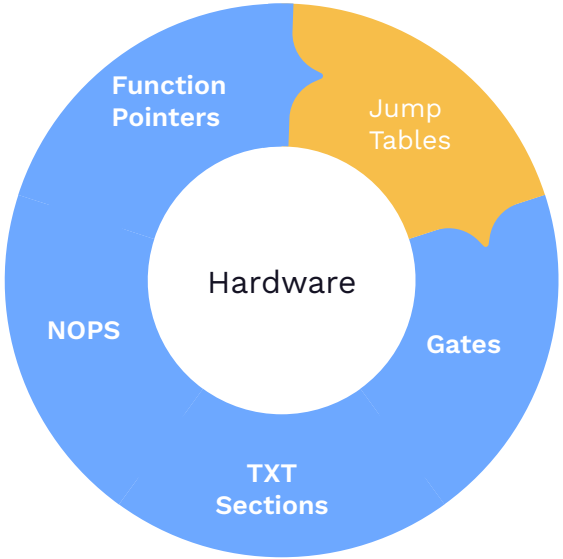
How OS Runtime Integrity Works

Baseline (Once)



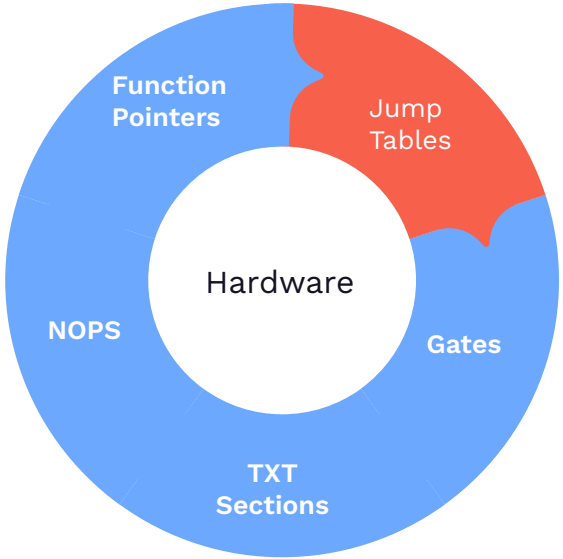
Analyze code and data structures **<u>once</u>**, to understand **the expected shapes** a kernel takes at **Runtime (in-memory).**

Measure



Periodic **sampling** of the kernel **in memory** to capture **the existing shape.**

Appraise



Appraise the current shape of the kernel to ensure it **matches an expected shape**.

How OS Runtime Integrity Works

Baseline (Once)

- A graph of kernel data structures & objects and their relationships
- ~ 1 million nodes in size
- Based on kernel data in memory & on disk
- Scoped to a distro & kernel minor version
- A Baseline can appraise any machine running that combo

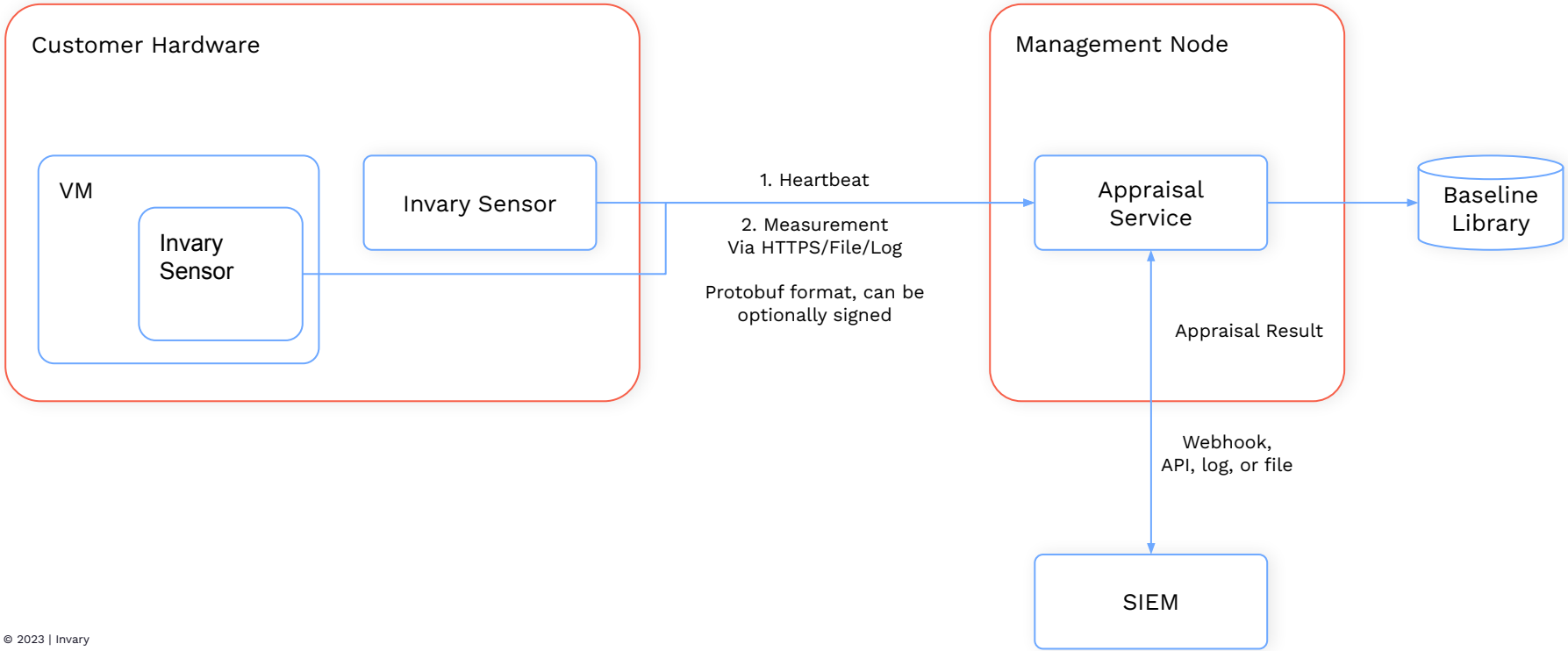
Measure

- Similar graph of data as baseline
- Taken on system under protection at Runtime
- Takes into account purposeful changes to the kernel at Runtime
- Measurements take ~300ms to complete

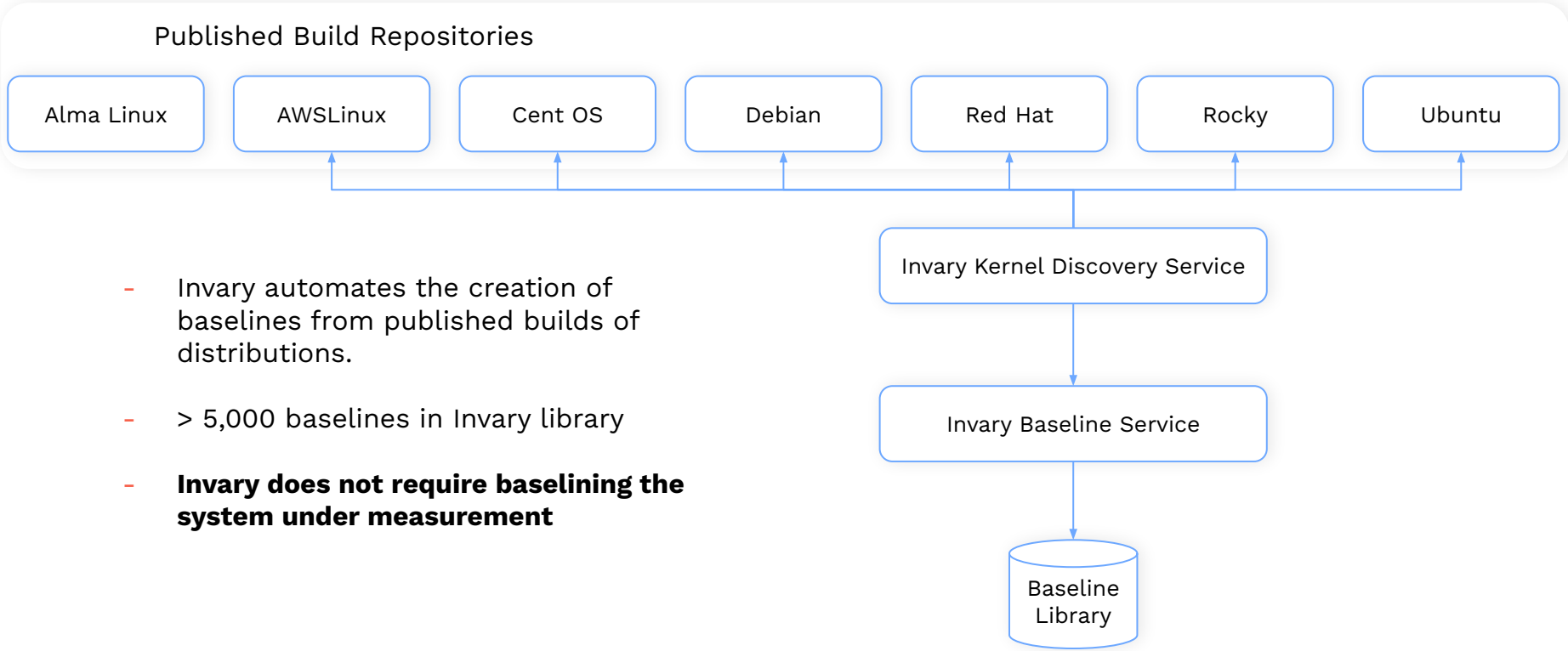
Appraise

- Appraise a measurement against the baseline
- High level have/don't have integrity signal
- Reports details where a kernel lacks integrity
- Finds rootkits, kernel defects, 3rd party impact on kernel
- Designed to appraise independent of measured machine

Data Flow



Invary Automates Baselining Process for Common Distro+Kernels



- Invary automates the creation of baselines from published builds of distributions.
- > 5,000 baselines in Invary library
- **Invary does not require baselining the system under measurement**

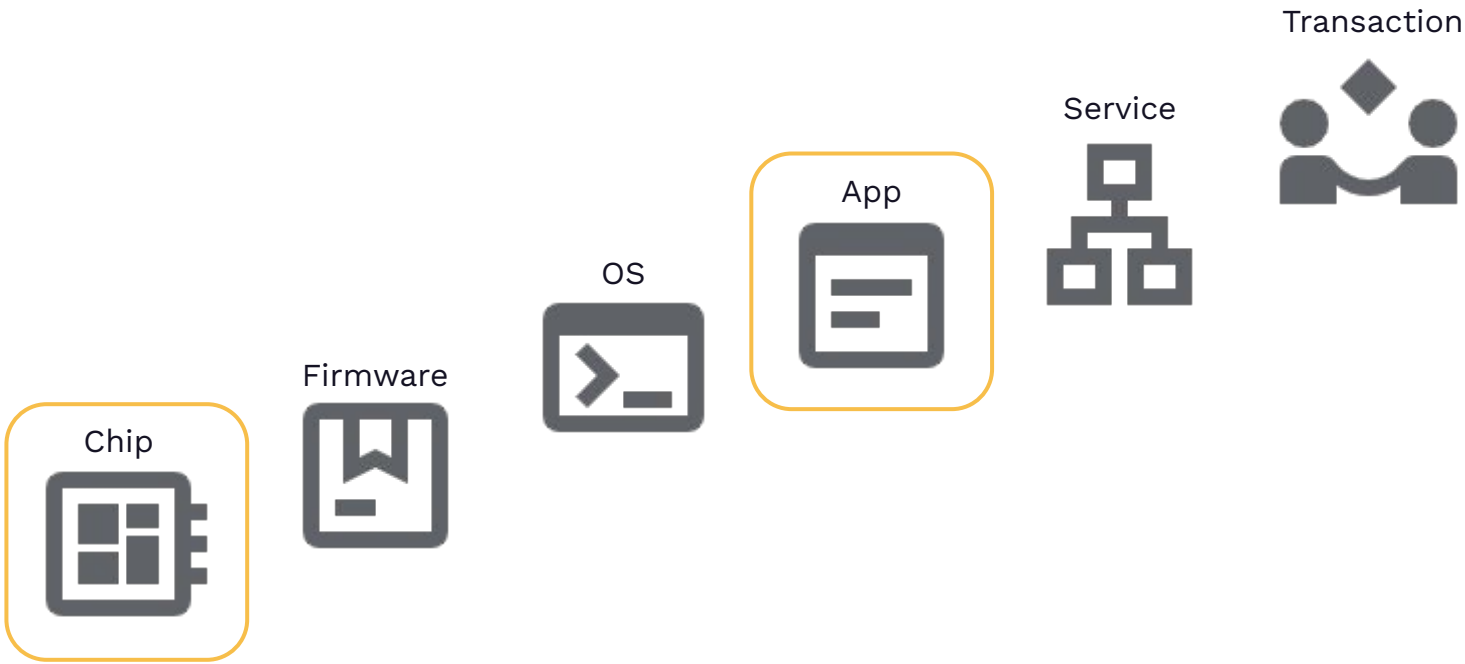
Footnote: Windows forthcoming

OS Details

- Uses eBPF, doesn’t extend the kernel attack surface
- Everything written in Rust
- SaaS, on-premise (air gapped), or single instance
- Tested via many rootkit techniques

OS Summary

- Finds rootkits deployed, often deployed via in-memory compromise at runtime
- Finds defects in the kernel (e.g. XZ, use after free CVE’s, etc..)
- Finds 3rd parties violating the kernel’s integrity
- Establishes OS Integrity for upstream and downstream attestation and security checks.



Continual... “attested Trusted Execution Environment”

In order to remove (limit) assumptions about a confidential computing environment at Runtime

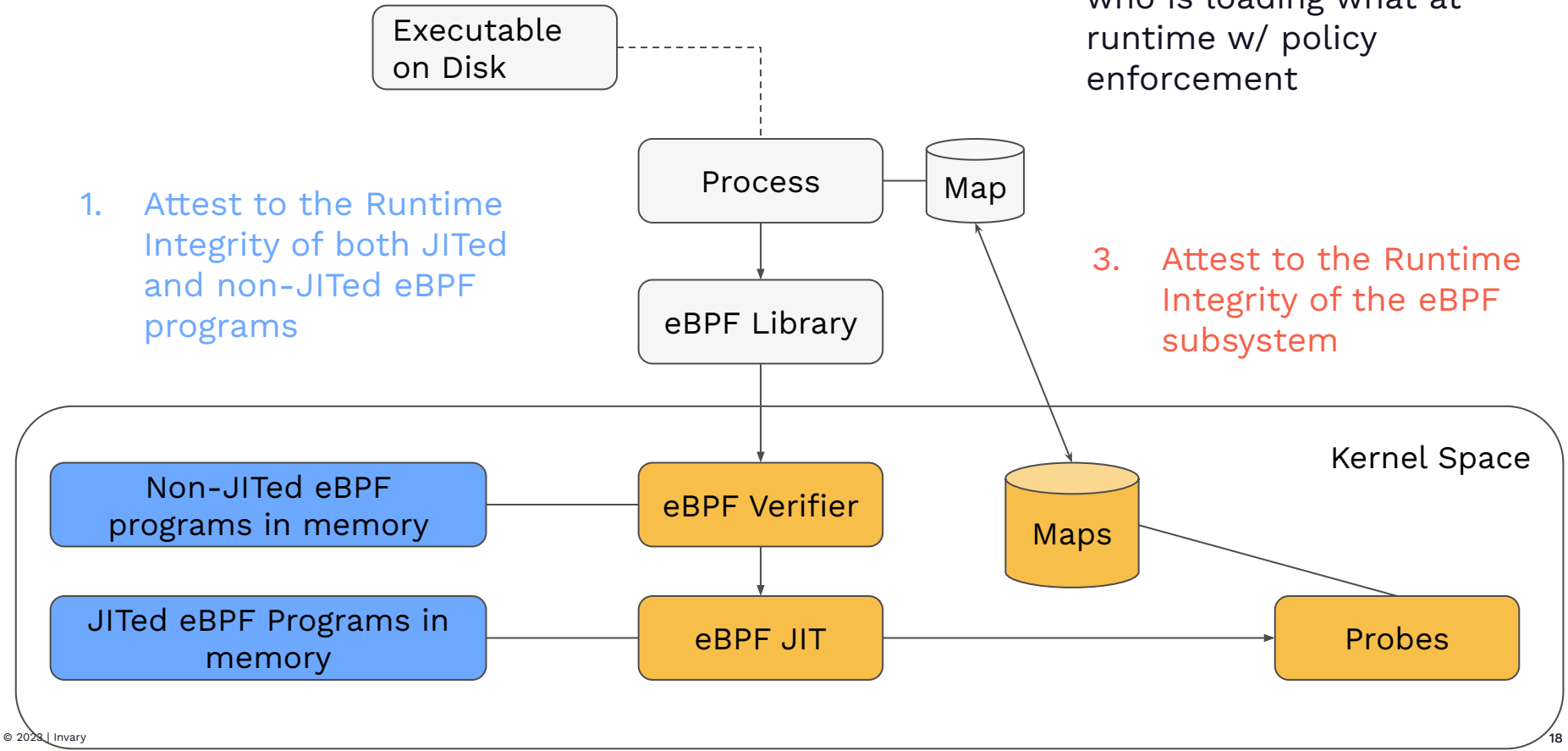
1. We perform Runtime Attestation of OS
2. Runtime Attestation of TEE’s (both hardware and confidential VMs)
    a. Which starts with hardware attestation
    b. And expands into any parts of the computing stack that is assumed to be trusted at runtime

Hardware Root of Trust

- AMD/Invary Project for CC Summit ‘24
  - Combining AMD SEV-SNP and OS Runtime Attestation for K8S Workload Management
  - MAAT - open source multiple attestation manager developed by the NSA Trusted Mechanisms team
- Attest to guest memory integrity
- Continued attestation of the TEE throughout Runtime to ensure underlying host / hardware hasn't changed during operation
- Currently working on prototype for ARM CCA

eBPF Runtime Integrity

2. Add observability of who is loading what at runtime w/ policy enforcement



1. Attest to the Runtime Integrity of both JITed and non-JITed eBPF programs

3. Attest to the Runtime Integrity of the eBPF subsystem

Multiple Attestation Managers at Runtime

- Common open source approach to performing and decisioning on multiple attestations from a diverse set of third parties at runtime
- NSA's MAAT as a reference architecture
  - Veraison, VirTEE, OAK, Maestro
- Research focus of Invary's CRADA with the NSA (along with Kernel Integrity and App Integrity)

A Few High Level Needs (Requirements?)

- Continual at Runtime
  - Passive and Active
- A usable history of attestation
  - What attestations changed and when and with what result
  - Chain of custody, and a sense of “freshness”
- Allows for diverse attestation mechanisms with rich data
  - Yet results in understandable common output (e.g. has, doesn’t not have integrity)
- Flexible toward ASP (Attestation Service Provider) implementation language and approach
- Usable by Eng/DevOps/DevSecOps
- Understanding a network of systems
- RATS compliant but not RATS restricted