Echtzeit-Threat-Detection mit Falco

SLAC 2026

12. Mai 2026

Wer sind wir?

NETWAYS Web Services bietet

- Public Cloud basierend auf OpenStack
  - Compute
  - Netzwerk
  - Block/Object Storage
  - Dedizierte CPUs/GPUs
- Managed Kubernetes®
- Managed AI Models
- Fertige Managed Plattformen (GitLab, Nextcloud, Observability Stacks)

Die NETWAYS Cloud in Zahlen

Womit arbeiten wir?

- **2 Rechenzentren** (colocated)
- **~1300 CPU Cores**
- **~1.4PB Ceph-Cluster** mit ~340 OSDs
- **GPUs** (A10, A40, RTX 6000 Blackwell)
- **~2000 VMs** von Kunden
- **~60 Kubernetes Cluster** von Kunden

Networks, systems and applications shall be monitored for anomalous behaviour and appropriate actions taken to evaluate potential information security incidents.

- ISO/IEC 27001:2022, Annex A, 8.16 Monitoring Activities

NETWAYS WEB SERVICES

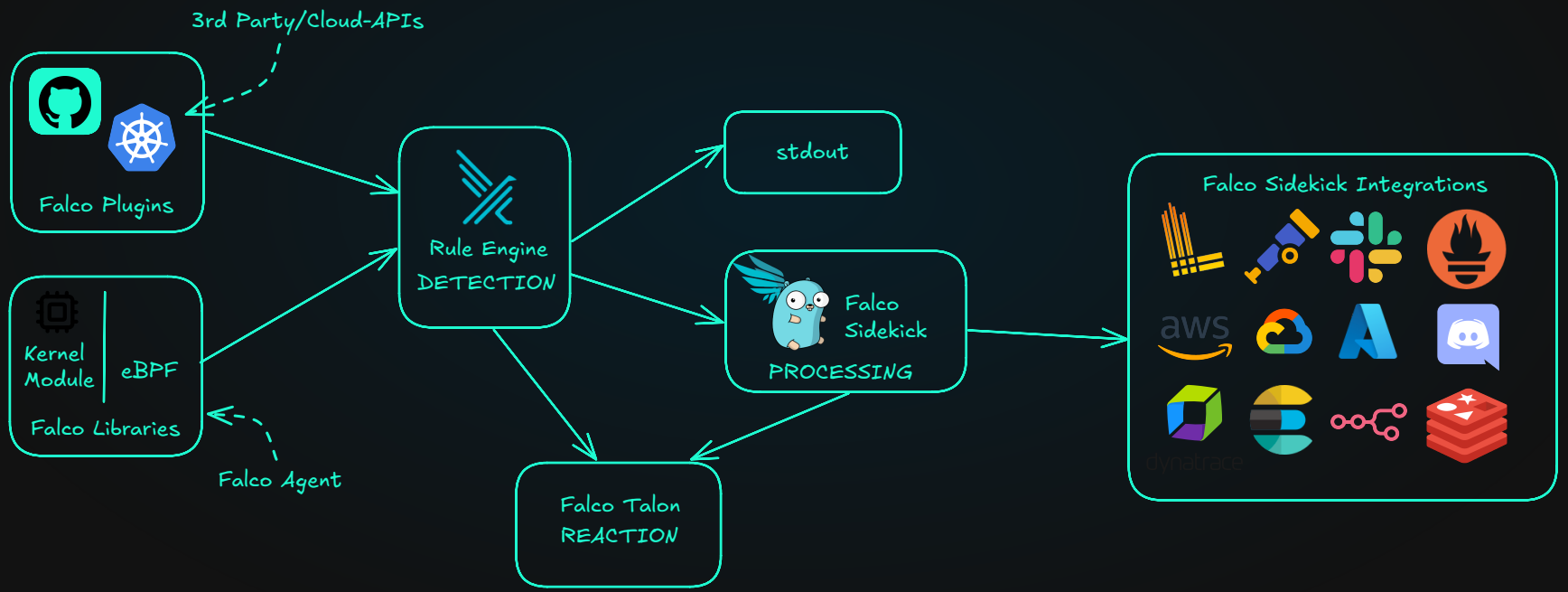
Falco im Überblick

Runtime Security für Hosts, Container, Kubernetes und Cloudumgebungen

Was ist Falco?

- Software zur **Echtzeit-Erkennung von Events**
- Baseline an Regeln inklusive
- Einsetzbar auf verschiedenen Ebenen:
  - Host (Bare Metal oder VM)
  - Container (Docker, Kubernetes, Nomad)
- Integrationen für externe Systeme, z.B.
  - Hyperscaler (AWS, GCP, Azure, RedHat)
  - Observability-Lösungen (Grafana, Datadog)
- Verschiedene Architekturen ermöglichen den Einsatz auch in Legacy-Umgebungen

Architektur von Falco



Falco im Kernel

Für die Verarbeitung von Events auf überwachten Endpoints beobachtet Falco **syscalls** und **Kernel Functions**. Hierfür wird einer von drei *Drivern* installiert:

🥉 Kernel Module

- ab Kernel-Version:
  - **x86-Systeme**: >= 2.6
  - **aarch64-Systeme**: >= 3.4
- muss bei Installation von Falco auf dem Host dauerhaft installiert werden und wird von Falco geladen.

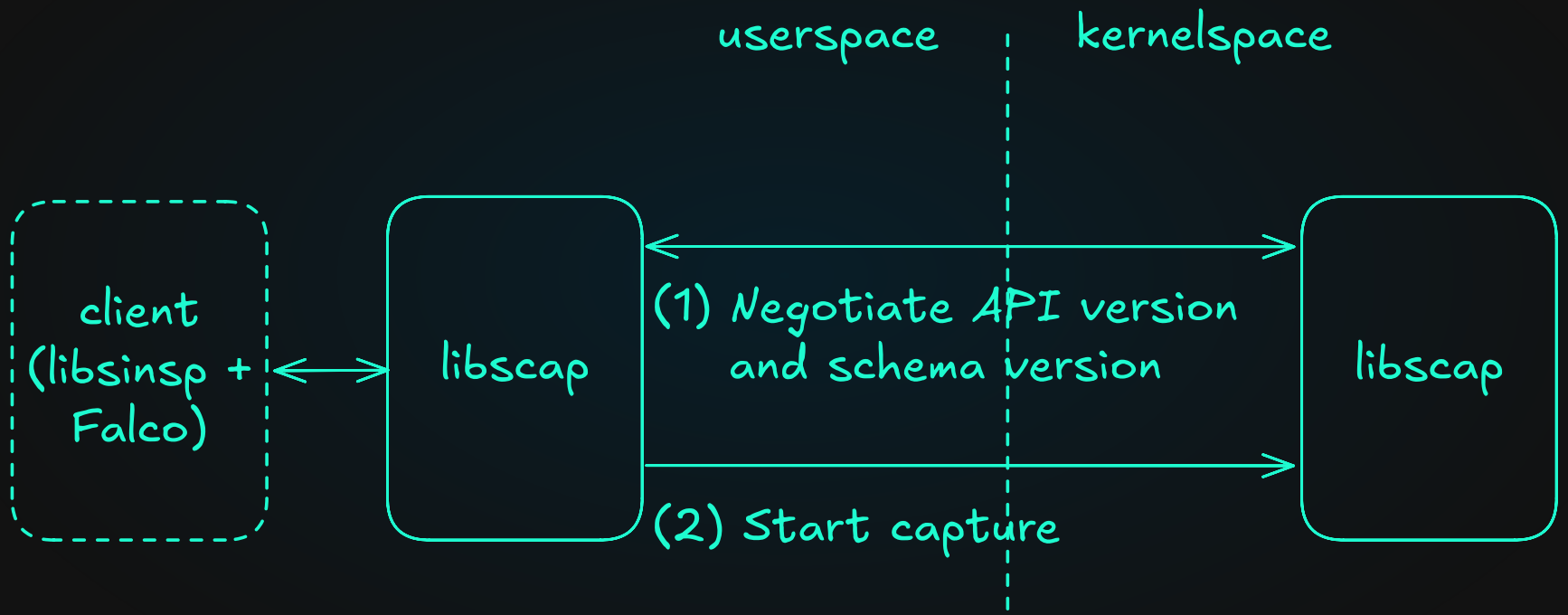
🥇 Modern eBPF Probe

- ab Kernel-Version: >= 5.8
- Weniger Overhead als das Kernelmodul, integriert in Falco
- Falco lädt das eBPF-Programm bei Bedarf
- *Least Privileged Mode* konfigurierbar

🥈 Legacy eBPF Probe

- ab Kernel-Version:
  - **x86-Systeme**: >= 4.14
  - **aarch64-Systeme**: >= 4.17
- verhält sich analog zur *Modern eBPF Probe*

Userspace vs. Kernelspace



Grafik aus der Dokumentation von Falco entnommen.

Falco Rules

```
- rule: shell_in_container
  desc: notice shell activity within a container
  condition: >
    evt.type = execve and
    evt.dir = < and
    container.id ≠ host and
    (proc.name = bash or
     proc.name = ksh)
  output: >
    shell in a container |
    user=%user.name
    container_id=%container.id
    container_name=%container.name
    shell=%proc.name parent=%proc.pname
    cmdline=%proc.cmdline
  priority: WARNING
```

- YAML-Objekte mit benötigten + optionalen Keys
- `condition` kann aus Macros und Lists zusammengebaut werden
- `output` kann auf Metadaten des beobachteten Events zurückgreifen

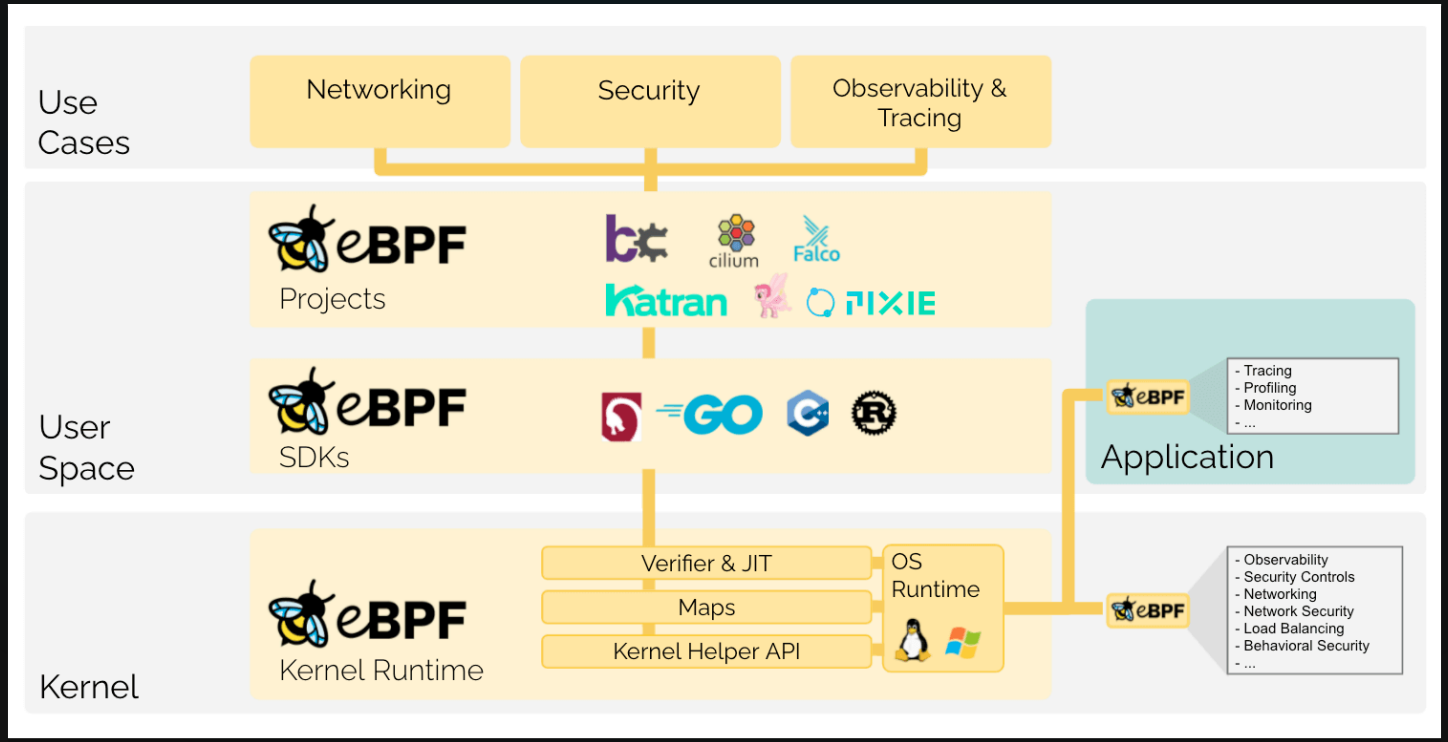
Exkurs: Was ist eBPF?

eBPF

Sichere, privilegierte Programme im OS-Kernel

- » Weiterentwicklung von BPF (*Berkeley Packet Filter*)
- » Ermöglichen **Erweiterung** der Funktionen des OS-Kernels **zur Laufzeit**
- » **Sandboxing** und **JIT-Kompilierung** für hohe Performance und Sicherheit
- » **Event-basiert** und an *Hooks* gebunden (syscalls, Tracepoints, usw.)
- » mehr und mehr Einfluss in den Bereichen **Networking, Security** und **Observability**

eBPF im Überblick



Grafik von ebpf.io entnommen.

Der eBPF Lifecycle

Stark vereinfacht

1. Identifizierung des gewünschten Hooks (z.B. `execve` syscall)
2. Laden des eBPF Programms an der identifizierten Stelle
3. Verifizierung der Korrektheit und Sicherheit des eBPF Programms:
   - » *Ist der ausführende Prozess ausreichend privilegiert?*
   - » *Ist die Größe des eBPF Programms im konfigurierten Rahmen?*
   - » *Ist die Komplexität des eBPF Programms niedrig genug?*
   - » *Kann die Speichersicherheit des eBPF Programms garantiert werden?*
   - » *Terminiert das eBPF Programm nachweislich?*
4. Kompilierung des eBPF Programms in Bytecode
5. Ausführung des eBPF Programms an der gewünschten Stelle (*Hook Point*, *kprobe* oder *uprobe*)

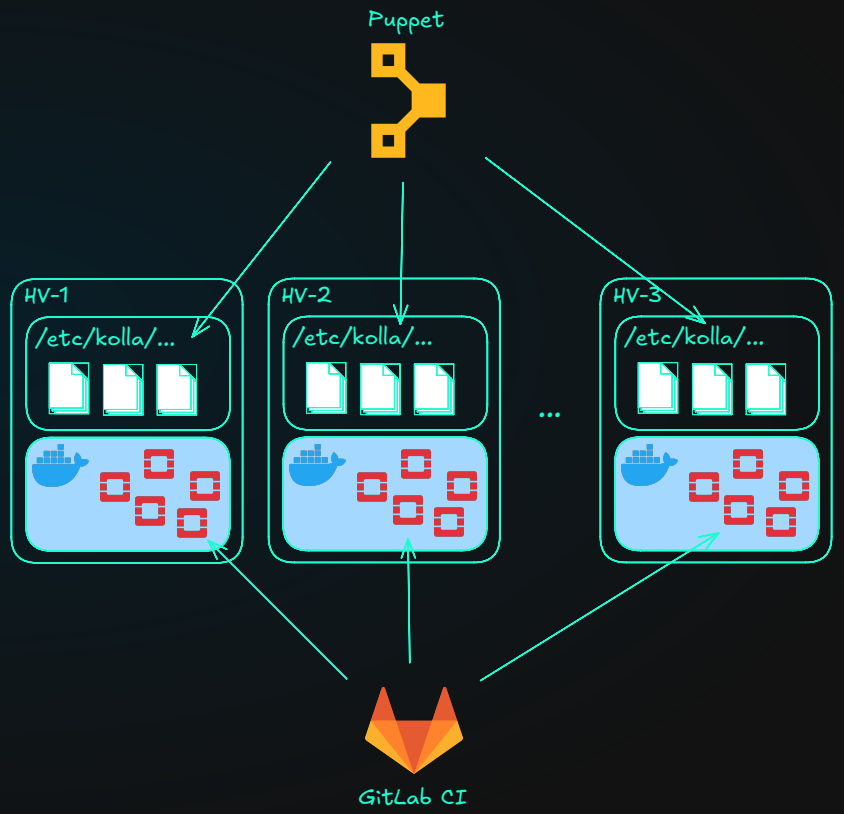
NETWAYS WEB SERVICES

Falco@NETWAYS

Falco@NETWAYS

Unsere Ausgangslage

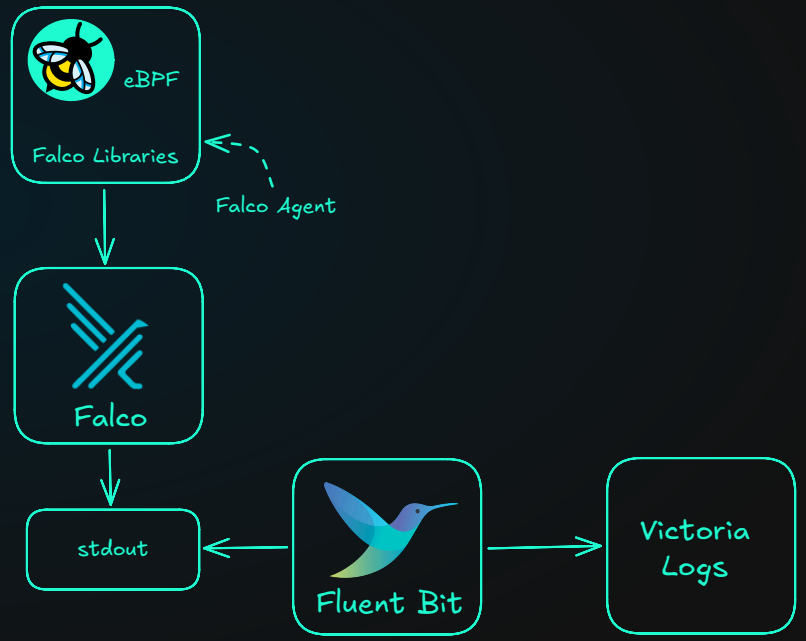
- OpenStack-Komponenten als Container aus dem OpenStack Kolla Projekt
- Patches etc. via GitLab CI
- Konfiguration via Puppet
- Deployment auf den Hypervisoren (Baremetal) via GitLab CI
- Kein Managementlayer unter OpenStack (Kubernetes, Yaook, etc.)



Falco@NETWAYS

Unser Falco-Setup

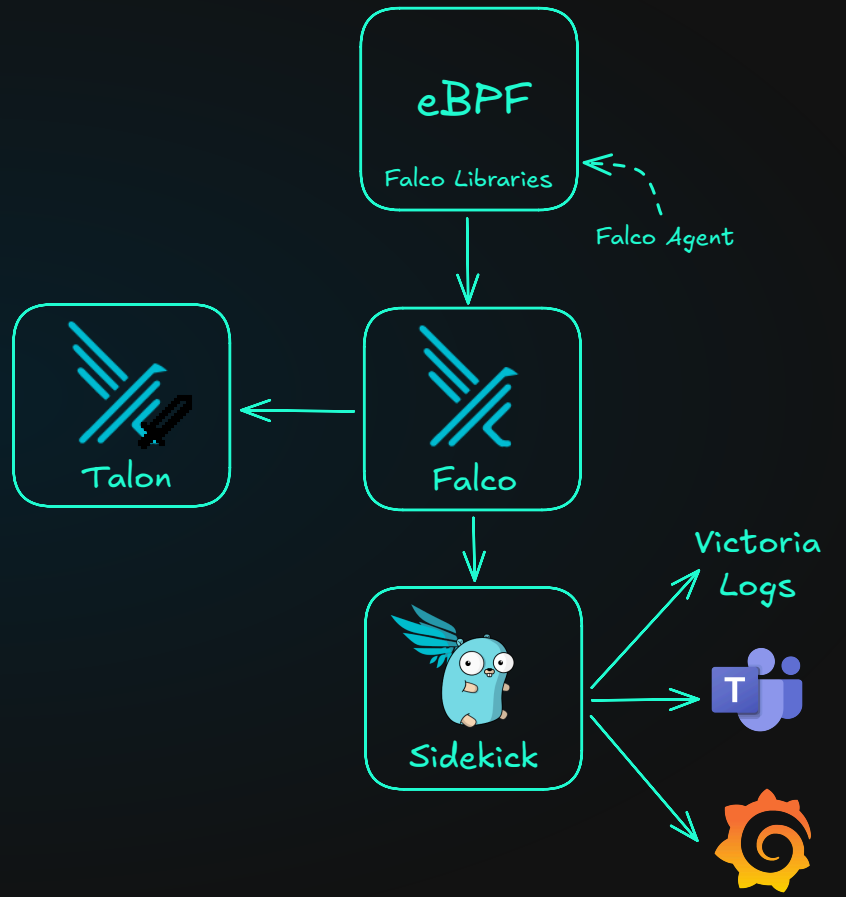
- Deployment mit Docker
- Baseline Ruleset mit wenigen Ergänzungen
- kein Falco Sidekick oder Talon im Einsatz
- Events werden v.a. für Audits benötigt:
  - Scraping via FluentBit
  - VictoriaLogs als Langzeitspeicher
  - keine Alarmierung oder (automatische) Reaktionen



Falco@NETWAYS

Nächste Schritte

- Verarbeitung von Events durch Falco Sidekick
  - Grafana IRM
  - On-Call Pager
  - OpenSearch
- Evaluierung von Falco Talons Entwicklung
  - Unterbindung von einzelnen auffälligen Prozessen (wie Tetragon)
  - *Quarantäne-Modus*
  - Mitschneiden von Shell-/Containersessions



Falco@NETWAYS

Unsere Evaluierung Falco vs Tetragon

	Falco	**Tetragon**
Technologie	Kernel Modul/eBPF	eBPF
Management in Kubernetes	Daemonset + Sidecar	Daemonset + Operator
Konfiguration	Rulefiles + Baseline-Regelset	CRDs + Sammlung an Beispielen
Lernkurve	**Flach**	**Steil**
Features	**Ausreichend** (Eventsourcing, einfache Reaktionen)	**Umfangreich** (Eventsourcing mit Custom Hooks, Reaktionen auf Processlevel)

Fazit und Takeaways

- Falco bietet einen exzellenten Einstieg für neue Nutzer:
  - Baseline an verfügbaren Regeln
  - abstrahierte Konzepte aus dem Kernel, menschenlesbar
  - Integrationen für verschiedene Eventquellen und -Destinationen
- Falco ist ein stiller Beobachter:
  - Events werden nur protokolliert und weitergeleitet
  - Falco Talon ist noch nicht vollends ausgereift
- Je nach Gesichtspunkt gibt es durchaus Tradeoffs:
  - Regelabstraktionen vs vollständige Kontrolle im Kernel
  - schlankes Deployment-Modell vs. umfangreicheres Managementlayer

NETWAYS WEB SERVICES

Vielen Dank

für eure Aufmerksamkeit

💻 **Demos**: netways-web-services/falco-k8s-demo

📚 **Falco Website**: falco.org

💬 LinkedIn: in/daniel-bodky