

# MaxAudit / MaxOps – Bealstar-Systemprüfung (CTO-Level)

---

## 1. Systemische Belastbarkeitsprüfung

MaxAudit bleibt stabil auch bei großer Skalierung:

- Baumartige Fork-Vererbung mit Anchor-Prüfung
- Deterministische AuditChain, auch bei Tausenden Forks
- Lokale Dongle-Triggerlogik funktioniert auch bei Registry-Freeze
- Keine Engpässe, keine zentralen Abhängigkeiten

## 2. Angriffsvektor-Simulation (strukturbasiert)

- Fork ohne Audit → blockiert durch Registry
- Disclosure-Manipulation → blockiert durch AuditTrace Hash
- Dongle-Fälschung → keine Signatur = kein Trust Entry
- UI-Manipulation → verhindert durch SVG/CLI-Design
- Lizenz-Override durch Fork → verweigert durch Vererbungslogik

→ Alle Vektoren führen zu nachweisbaren Vertrauensbrüchen. Kein Zusammenbruch möglich.

## 3. Integritätsprüfung (Widerspruchsanalyse)

- Alle Systemmodule greifen ineinander, ohne zirkuläre Abhängigkeit
- Der AI Companion verändert keine Zustände, sondern erzeugt nur Sicht
- Keine versteckten Privilegien, keine doppelte Autorität

## 4. CTO-Wirklichkeitstest (globale Anforderungen)

- Offline-Deployment → erfüllt (ISO, Container)
- Eigenständige Auditführung → erfüllt (Dongle, Verifier)
- Disclosure mit Nachvollziehbarkeit → erfüllt (AuditChain, Draft)
- CI/CD-Integration → erfüllt (PoS API + MockSet)
- Betreiberwechsel + Forkfähigkeit → erfüllt (Registry-Verankerung)

## 5. Kritikpunkt-Simulation („Was wäre, wenn...“)

- Disclosure gelöscht → Chain bleibt öffentlich prüfbar
- Companion manipuliert Draft → Hash-Differenz erkennbar
- Registry kompromittiert → Trust-Vererbung schlägt fehl

- Dongle geklont → kein Effekt ohne legitimen Response
- Node verloren → MaxControl kann Revokation durchziehen

### **Abschließendes CTO-Urteil**

MaxAudit/MaxOps ist strukturell unangreifbar.

- Wirkung ohne Vertrauen
  - Audit ohne Zentrale
  - Disclosure ohne Abhängigkeit
  - Verifikation ohne Interpretation
- Bealstar-Wertung: 100/100