HCLSoftware

Open Source vs. Enterprise Software? Es geht auch zusammen!

Michael Würdemann
Lead Digital Sovereignty DACH

Michael.Wuerdemann@hcl-software.com

Mythbuster

Open Source und Digitale Souveränität

Warum der Lizenztyp allein weder Kosten noch Sicherheit oder Handlungsfähigkeit garantiert

HCLSoftware

Die drei Mythen in 30 Minuten

Orientierung vor Positionierung

Mythos 1

OSS ist kostenlos

Realität: Lizenzkosten sind nur eine einzelne Kostenart
Betrieb, Integration, Sicherheit, Updates und Support bleiben.

Mythos 2

OSS ist automatisch sicher

Realität: Offener Code ist auditierbar, aber Audits, Maintainer Trust und Build-Pipelines müssen aktiv gemanagt werden.

Mythos 3

Closed Source heißt Lock-in

Realität: Auch proprietäre Enterprise-Produkte können OSS nutzen, härten, supporten und kontrollierte Migration ermöglichen.

Souveränität = Transparenz + Kontrollfähigkeit + Betriebskompetenz + Anbieter-Verantwortung.

HCLSoftware

Mythos 1: „OSS ist kostenlos“

Korrektur: Die Lizenz kann frei sein, die Wertschöpfungskette ist es nicht.

Kosten liegen oft unter der Wasserlinie

Weiterentwicklung	Roadmap, Feature-Gaps, UX, Integrationen
Wartung	Patches, CVEs, Regression Tests, Versionierung
Betrieb	SLA, Monitoring, Backup, Skalierung, Hochverfügbarkeit
Support	L3-Know-how, Eskalation, Dokumentation, Training
Migration	Datenmodell, APIs, Customizing, Change Management

openDesk selbst belegt den Punkt

ZenDiS nennt für openDesk eine Community Edition „ohne Service und Support“.
Für die Enterprise Edition wurden Koordination der Weiterentwicklung, Service, Wartung, Qualitätssicherung, Pflege und regelmäßige Updates vergeben.

Mythbuster-Satz:

„Kostenfrei“ ist nicht dasselbe wie „kostenkontrolliert“ oder „betriebsfähig“.

Mythos 2: „Wenn jeder auditieren kann, wird es schon jemand tun“

Korrektur: Auditierbarkeit ist eine Option, kein Kontrollnachweis.

Offener Code reduziert Blindflug, aber nicht automatisch Risiko.

01

Maintainer Trust

Wer darf veröffentlichen? Wer kontrolliert Rechte, Tokens und Schlüssel?

02

Build & Release

Stimmen Git-Quelle, Release-Artefakt und Paketregistry überein?

03

Transitive Risiken

Welche indirekten Dependencies ziehen wir mit?

04

Betriebs-verantwortung

Wer patcht, testet, migriert und haftet praktisch für SLA?

Auditierbar **Gehärtet** **Betrieben** **Verantwortet**

Sicherheit entsteht erst durch den kompletten Prozess.

HCLSoftware

Aktuelle Gegenbeispiele: Supply Chain wird zur Angriffsfläche

Die Angriffe zielen nicht nur auf Code, sondern auf Vertrauen, Maintainer und Release-Pipelines.

2024	XZ Utils / liblzma	Backdoor in Release-Tarballs, CVE-2024-3094, potenziell SSH-Bypass.
2024	OpenJS Takeover Attempt	OpenSSF/OpenJS warnen vor social-engineering-basierten Maintainer-Übernahmen.
2024	Polyfill.io	Domain-/Projektübernahme führte zu injiziertem JavaScript auf 100k+ Sites.
2025	tj-actions / reviewdog	Kompromittierte GitHub Actions legten CI/CD-Secrets offen.
2025	npm Shai-Hulud	Selbstreplizierender Worm über kompromittierte Maintainer-Accounts, 500+ Packages entfernt.

Pattern **Moderne OSS-Angriffe kompromittieren Governance, Identität und Distribution, nicht nur einzelne Codezeilen.**

Quellen: CISA/CVE, GitHub, OpenSSF, Cloudflare/Sansec

Souveränität ist kein Lizenzmodell

Sie ist ein Managementmodell für Abhängigkeiten.

Vier Leitfragen für die Beschaffung

Kontrolle

Kann ich Daten, Betrieb, Identitäten und Schlüssel in meinem Rechtsraum kontrollieren?

Austauschbarkeit

Kann ich Komponenten ersetzen, ohne die Fachprozesse neu zu bauen?

Nachweisbarkeit

Kann ich Security, Compliance, Support und Lifecycle belastbar dokumentieren?

Verantwortung

Wer steht vertraglich gerade, wenn ein Maintainer verschwindet oder ein Projekt kippt?

Lizenztyp ist nur ein Input. Souveränität ist das Ergebnis des Betriebs- und Governance-Modells.

HCLSoftware-Position: OSS nutzen, härten, verantworten

Pragmatisch statt dogmatisch: Open Source ist Baustein, nicht Betriebsmodell.

Der Enterprise-Wert entsteht im kontrollierten Lifecycle

1
Inventarisieren
SBOM, Herkunft, Lizenz, Version

2
Härten
Security Reviews, Patches, Konfiguration

3
Integrieren
APIs, UX, Identity, Admin Policies

4
Betreiben
Updates, Support, SLA, Eskalation

5
Austauschen
Migration, Ersatzkomponenten, Exit-Pfade

Das ist der Unterschied zwischen „wir verwenden OSS“ und „wir liefern Enterprise-Software mit OSS-Komponenten“.

Quelle: HCLSoftware Open Source

HCLSoftware

Beispiel: HCL Sametime + Jitsi

OSS-Fundament plus Enterprise-Augmentierung für Webmeetings.

Jitsi ist eine leistungsfähige Open-Source-Basis. HCL Sametime macht daraus einen Enterprise-Service.

Jitsi-Basis

WebRTC, Videobridge/SFU, offene Architektur, breite Community

HCLSoftware-Augmentierung

Security Guidelines, UI/UX, persistenter Chat, Moderation, Admin-Policies, Directory-Bindung

Enterprise-Betrieb

Private Cloud oder on-prem, Kubernetes/Autoscaling, Support, Dokumentation, Lifecycle

Wir ersetzen nicht Open Source durch Proprietärdogma.
Wir machen Open Source enterprise-fähig, integrierbar und supportbar.

Quellen: HCL Sametime Docs/Community, Jitsi

HCLSoftware

OSS-Stack: Auch Plattformen sind Teil der Souveränitätsarchitektur

Beispielhaft: Linux, Red Hat Enterprise Linux, Kubernetes, OpenShift.

Die Frage lautet nicht: „OSS oder Closed Source?"

Die Frage lautet: „Welcher Stack ist kontrollierbar, supportbar und auditierbar?"

Betriebssystem	RHEL / Linux	HCLSoftware läuft häufig auf OSS-basierten Enterprise-Plattformen.
Container / Orchestrierung	Kubernetes / OpenShift	Skalierung, Automation, Load Balancing, Standardisierung.
Applikation	HCLSoftware Produkt	Härtung, Integration, Support, Release- und Migrationspfad.
Kunde / Betrieb	Eigenbetrieb / Private Cloud / Souveräne Cloud	Kontrolle über Daten, Identitäten, Betriebsmodell und Vertragsbeziehungen.

Konsequenz: OSS ist kein Gegenargument zu HCLSoftware, sondern oft Teil der HCLSoftware-Referenzarchitektur.

Quellen: HCL Sametime Docs

Vergleichsmatrix: Was kauft der Kunde wirklich?

Nicht Code gegen Code vergleichen, sondern Verantwortung gegen Verantwortung.

Kriterium	Community OSS DIY	OSS Enterprise/SaaS	HCLSoftware Produkt
Lizenzkosten	niedrig	mittel	Planbar im Vertrag
Betrieb & SLA	Kunde	Partner/Provider	HCL/Partner-Modell
Security & Patching	Eigenverantwortung	Teilweise gebündelt	Produkt-Lifecycle
Integration	Projektaufwand	Vendor-spezifisch	Produktintegration
Exit / Austausch	Selbst zu lösen	Abhängig vom Modell	Managed Migrationspfad

Kaufentscheidung: Welche Risiken will der Kunde selbst tragen, welche sollen vertraglich und technisch abgefedert werden?

HCLSoftware

5 Mythbuster-Fragen

Von der Ideologie zur belastbaren Entscheidung.

1	Wer finanziert Weiterentwicklung und Pflege?	Nicht: „Ist es frei?" Sondern: „Wer trägt dauerhaft Roadmap, Wartung und QA?"
2	Wer ist im Incident accountable?	Nicht: „Der Code ist offen." Sondern: „Wer patcht, kommuniziert und eskaliert unter SLA?"
3	Wie wird die Supply Chain kontrolliert?	SBOM, Signaturen, Build-Transparenz, Maintainer-Rechte, Dependency Governance.
4	Wie wird aus OSS ein Enterprise-Service?	Identity, Policies, Logging, Support, Monitoring, Skalierung, Datenschutz.
5	Wie bleibt der Kunde austauschfähig?	Exit-Pfade, Migration, Datenportabilität, Ersatz von Komponenten bei Lizenz- oder Projektänderungen.

OSS als Baustein anerkennen, dann Verantwortungsmodell, Security-Nachweise und Lifecycle-Plan abfragen.

HCLSoftware

Schlussfolgerung: Souveränität ist Accountability

Nicht Open Source gegen Closed Source, sondern ungemanagte gegen verantwortete Lieferketten.

3 Takeaways

1	OSS ist wertvoll	aber nicht automatisch kostenlos, sicher oder betreibbar.
2	Enterprise-Souveränität braucht Governance	Security, Support, Lifecycle, Compliance und Exit-Fähigkeit.
3	HCLSoftware nutzt OSS pragmatisch	gehärtet, integriert, unterstützt und mit Migrationsoptionen im Produktkontext.

Merksatz für die Diskussion

„Digitale Souveränität beginnt bei Transparenz, aber sie endet erst bei operationaler Verantwortung."

HCLSoftware

HCLSoftware

hclsw.de