Security Concerns in Proprietary and OpenSource Software

Dorian Rabl

AGENDA

1. Einführung
2. Grundprinzipien der Software Security
3. Sicherheitsaspekte von proprietärer Software
4. Sicherheitsaspekte von Open-Source-Software
5. Schlussfolgerungen und künftige Trends

EINFÜHRUNG

in Open Source und Proprietary Software

- **Open Source Software:** Quellcode öffentlich zugänglich, jeder kann den Code einsehen, ändern und weiterverbreiten.
- **Proprietary Software:** Quellcode geschlossen, nur vom Entwickler oder autorisierten Parteien einsehbar und änderbar.

(Wikipedia, 2024)

Open Source Software

Beispiele: Mozilla Firefox, Linux

Vorteile	Nachteile
Transparenz	Sicherheitsrisiken
Community Support	Variierende Qualität
Kosten	

Typische Geschäftsmodelle:

- Dienstleistungen (Beratung, Schulung, Support)
- Freemium-Modell (Grundfunktionen kostenlos, erweiterte Funktionen kostenpflichtig)
- Spenden und Sponsoring

(Wikipedia, 2024)

Proprietäre Software

Beispiele: Microsoft Office, Adobe Photoshop

Vorteile	Nachteile
Anbieterunterstützung	Kosten
Integrierte Sicherheitsfunktionen	Mangel an Transparenz
Rechtliche Verantwortung	Abhängigkeit vom Anbieter

Typische Geschäftsmodelle:

- Lizenzverkauf (einmalige oder periodische Lizenzgebühren)
- Abonnements
- Software as a Service

(Wikipedia, 2024)

- **Software security aims at ensuring the confidentiality of information, the trustworthiness of data (integrity), and the availability of data and the system. These qualities are at risk if software contains vulnerabilities.**

(Glasauer et al., 2024)

Prinzipien der Softwaresicherheit

SOLID-S Modell

1. Sicherheit durch Design

- Sicherheit muss von Anfang an in den Entwicklungsprozess integriert werden. Beispiel: Reduzierung von Abhängigkeiten zwischen Komponenten.

2. Prinzip des geringsten Privilegs

- Software sollte nur mit den minimal erforderlichen Rechten ausgeführt werden. Beispiel: Begrenzung der Zugriffsrechte für Anwendungen.

3. Risikobewertung und Bedrohungsmodellierung

- Regelmäßige Bewertung von Risiken und Modellierung potenzieller Bedrohungen. Beispiel: Identifizierung von Angriffsvektoren und Schwachstellenanalyse.

(Glasauer et al., 2024)

Prinzipien der Softwaresicherheit

SOLID-S Modell

4. Sichere Programmierpraktiken

- Verwendung bewährter Methoden beim Programmieren, um Sicherheitslücken zu minimieren. Beispiel: Nutzung aktueller Compiler und Code-Analyse-Tools

5. Kontinuierliches Testen und Auditieren

- Software sollte nur mit den minimal erforderlichen Rechten ausgeführt werden. Beispiel: Begrenzung der Zugriffsrechte für Anwendungen.

6. Sicherheitskultur

- Regelmäßige Bewertung von Risiken und Modellierung potenzieller Bedrohungen. Beispiel: Identifizierung von Angriffsvektoren und Schwachstellenanalyse.

(Glasauer et al., 2024)

Sicherheitsaspekte: Proprietäre SW

Vorteile	Nachteile
Integrierte Sicherheitsfunktionen: Oft besser integrierte und umfassend getestete Sicherheitsmerkmale durch den Anbieter.	**Mangel an Transparenz:** Nutzer können den Quellcode nicht einsehen, wodurch versteckte Sicherheitslücken bestehen können.
Professioneller Support: Dedizierter Support und regelmäßige Sicherheitsupdates vom Anbieter.	**Abhängigkeit vom Anbieter:** Nutzer sind auf den Anbieter angewiesen, um Sicherheitsupdates und Patches zu erhalten.
Rechtliche Verantwortung: Anbieter sind rechtlich verpflichtet, Sicherheitslücken zu beheben und die Sicherheit zu gewährleisten.	**Langsame Reaktion:** In einigen Fällen kann die Reaktionszeit des Anbieters auf Sicherheitsvorfälle langsamer sein.
Sicherheitszertifizierungen: Häufig verfügen proprietäre Softwareprodukte über Sicherheitszertifizierungen und Compliance mit Industriestandards.	**Insider-Bedrohungen:** Risiko von Sicherheitslücken durch Insider im Entwicklerteam, die schwer zu erkennen sind.

(Smith, 2021; Brown & Johnson, 2019; Wilson, 2020; Davis, 2022)

Sicherheitsaspekte: OSS

Vorteile	Nachteile
Transparenz: Der Quellcode ist offen und kann von jedem überprüft werden, was zu einer höheren Sicherheitsüberprüfung führt.	**Sicherheitsrisiken durch Offenheit:** Der offene Quellcode kann von böswilligen Akteuren ausgenutzt werden, wenn Sicherheitslücken nicht schnell genug behoben werden.
Community-Support: Unterstützung durch eine große und aktive Community, die kontinuierlich an der Verbesserung der Sicherheit arbeitet.	**Fehlender dedizierter Support:** Abhängigkeit von Community-Support, der variieren kann.
Schnelle Identifizierung und Behebung von Sicherheitslücken: Dank der breiten Überprüfung durch die Community.	**Variierende Qualität:** Die Qualität der Sicherheitsimplementierungen kann stark variieren, abhängig von den Beiträgen der Community.
Flexibilität und Anpassungsfähigkeit: Nutzer können den Code an ihre spezifischen Sicherheitsanforderungen anpassen.	**Komplexität:** Die Nutzung und Anpassung der Software kann komplexer sein und erfordert oft tiefgehendes technisches Wissen.

(Martin, 2020; Kim & Park, 2018; Johnson, 2019; Nguyen, 2021)

KÜNFTIGE TRENDS

Five Trends That Will Shape Open-Source Security In 2024

Forbes, 2023

KI in der Softwareentwicklung: Fluch oder Segen für die Sicherheit?

Heise online, 2023

Banken

DevSecOps bei Atruvia: Mehr als ein Framework

Handelsblatt, 2023

TECHNOLOGY SECURITY

How cybersecurity hackathons boost innovation and talent discovery

Techcircle, 2024

Halluzinierende KI wird bei Software-Entwicklung zum Sicherheitsrisiko

Der Standard, 2024

Zukunftstrends

In der Software-Sicherheit

Künstliche Intelligenz und Maschinelles Lernen

- Automatisierte Bedrohungserkennung, präventive Sicherheitsmaßnahmen

Erhöhte Sicherheitsregulierung

- Neue Compliance-Anforderungen: strengere Vorschriften und Standards, globale Zusammenarbeit

Motivation der Entwicklercommunity

- Hackathons und Bug-Bounty-Programme

(Glasauer, 2023; Tomas et al., 2019; Assal & Chiasson, 2019)

Conclusio

Beide Softwaremodelle bieten einzigartige Vorteile und Herausforderungen.
Sicherheit erfordert einen ganzheitlichen Ansatz, der sowohl technische als auch organisatorische Aspekte berücksichtigt.

1

- **Hybridlösungen**

  Kombination von Open Source und Proprietary Software für optimale Sicherheit.

2

- **kontinuierliche Weiterbildung**

  Sicherstellen, dass Entwickler und Sicherheitsprofis auf dem neuesten Stand sind.

3

- **Technologische Innovation**

  Nutzung von AI und ML zur Verbesserung der Softwaresicherheit.

Diskussion