Christoph Wegener

Open-Source vs. Closed-Source

Gibt es die perfekte Lösung?

Über die Vor- und Nachteile von Open-Source-Komponenten wird oft kontrovers diskutiert, leider bewegt sich die Diskussion dabei aber nicht selten auf einem sehr emotionalen Niveau. Doch gibt es ja auch einige Punkte, mit denen sich Open- und Closed-Source-Lösungen objektiv gegeneinander abgrenzen lassen.

Open-Source

Diese wollen wir aufgreifen und dabei die jeweils häufigsten Argumente für die ein oder andere Seite gegenüber stellen und kritisch hinterfragen. Wir beginnen dabei zunächst mit den Argumenten in Bezug auf Open-Source.

▸ **Viele Augen sehen besser**
Open-Source-Lösungen werden oft schon allein deswegen als fehlerfreier und damit auch sicherer eingeschätzt, weil viele Personen den Quellcode reviewen könnten. Auf den ersten Blick ein sicherlich zutreffendes Argument. Doch die Möglichkeit, dass viele Personen in den Code schauen können, bedeutet eben leider noch lange nicht, dass dies auch tatsächlich passiert.

Dies hat leider auch das Beispiel von OpenSSL deutlich gezeigt, wo ein sicherheitskritischer Bug über 18 Monate hinweg nicht entdeckt wurde: Möglicherweise durch zu sorglosen Umgang mit „Finden und Ersetzen“ hatte ein Entwickler dabei genau diejenigen Codezeilen auskommentiert, die Zufall bei der Generierung von Schlüsseln produzierten. Das Resultat waren nun Schlüssel (auch in Zertifikaten), die nicht mehr ausreichend zufällig und damit relativ einfach zu brechen waren.

▸ **Responsible Disclosure?**
Fraglich ist zudem auch, ob gefundene Fehler überhaupt im Sinne eines „Responsible Disclosure“ gemeldet werden oder ob der potenzielle Angreifer diese geheim hält und zum richtigen Zeitpunkt selbst gewinnbringend einsetzt. Aber auch im Closed-Source-Umfeld werden viele Lücken auf dem Schwarzmarkt gehandelt.

▸ **Schutz durch Unabhängigkeit**
Dieses Argument kann aus zwei Perspektiven betrachtet werden: Der Unabhängigkeit von Herstellerstrategien und der Unabhängigkeit im Punkte Support. Zum Punkt Herstellerstrategie ist anzumerken, dass zumindest heutzutage ein nicht unbedeutender Anteil größerer Open-Source-Projekte von Industrieunternehmen gesponsert wird. Und im Normalfall gibt natürlich der Sponsor auch die Richtung des Projekts vor.

Im Punkte Support hat man sicherlich durch die Quelloffenheit einen Vorteil: Wenn der Dienstleister oder Programmierer nicht mehr zur Verfügung steht, kann ein Mitbewerber auf Grundlage des Quellcodes dessen Aufgaben übernehmen. Ob dies immer reibungslos funktioniert, ist aber noch eine ganz andere Frage.

Closed-Source

Auf der anderen Seite gibt es natürlich auch Argumente in Bezug auf einen Einsatz von Closed-Source.

▸ **Geheim macht sicher?**
Beispielsweise kann „Geheimhaltung“ als Argument angeführt werden. Danach ist Software nur dann sicher, wenn der potenzielle Angreifer den Quellcode nicht kennt. Dieses Argument hat sich allerdings schon in der Kryptographie als unhaltbar erwiesen. Perfekte, also zu 100% fehlerfreie, Software gibt es sowieso nicht, weder mit offenen, noch mit geschlossenen Ansätzen. Und wo Fehler existieren, werden diese irgendwann auch gefunden.

▸ **Haftung beruhigt?**
Auch aus Gründen der Haftung wird Closed-Source-Komponenten oft der Vorzug gegeben, hat man hier doch scheinbar einen Hersteller, den man im Falle des Falles auch für Schäden haftbar machen kann. Ob sich das in der Praxis dann auch wirklich durchsetzen lässt, darf vor allem bei Auseinandersetzungen zwischen David (z.B. mittelständiges Unternehmen) und Goliath (z.B. Software-Riese) bezweifelt werden.

Randprobleme

▸ **Sicherheitsevaluierungen**
Bei der Evaluierung von quelloffenem Code stellt sich zudem das Problem, dass eine Evaluierung immer genau für einen ganz bestimmten Snapshot gilt. Damit ist aber eine Community-basierte Weiterentwicklung unter Beibehaltung der auf diesem Snapshot basierenden Zertifizierung versagt, zumal eine Evaluierung meist auch sehr kostspielig und zeitintensiv ist.

▸ **Standards**
Wenn man sich von Standards wegbewegt, tauchen zudem immer Probleme auf. Hier ist die Frage der Verbreitung der benötigten Software-Komponenten der entscheidende Punkt, denn Standards werden in der Hauptsache durch die Anzahl der Anwender gesetzt. Der Hauptaufwand von Migrationen liegt so fast immer in der Anpassung des Datenaustausches mit „Anderen“ – Anderen, die aber den Standard nutzen.

Wie es Euch gefällt

Bleibt die Frage offen, was denn nun zu bevorzugen sei. Diese Entscheidung sollte in jedem Fall von Sachfragen abhängig gemacht werden. Wichtige Kriterien können hier etwa die benötigten Funktionen, die zukünftige Erweiterbarkeit, die mögliche Unabhängigkeit im Punkte Support und nicht zuletzt auch die Kosten sein. Gerade im letzten Punkt ist aber immer eine globale Betrachtung notwendig, denn nicht selten ist die Software selbst zwar preiswert, die nachfolgenden Dienstleistungen kosten dann aber häufig ein Vielfaches der Lizenz. Und auch dies gilt für Open- und Closed-Source-Ansätze.

Ob nun eine Open- oder Closed-Source-Variante zum Einsatz kommt, sollte nach den Sachfragen entschieden werden. Eine generelle Empfehlung, was besser ist, kann es außerem definitiv nicht geben. Dies bleibt im Großen und Ganzen eine Einzelfallentscheidung. Oder wie sagt man auf Neudeutsch so schön: It depends.