Risikominimierung bei kommerziell genutzter Open Source

Stefan Haßdenteufel[1] · Harald Fleischhauer[2]

[1]SSW Schneider Schiffer Weihermüller
Stefan.Hassdenteufel@ssw-muc.de

[2]Hensoldt GmbH
harald.fleischhauer@hensoldt.net

Zusammenfassung

Einsatz und Weitergabe von Open Source Software in kommerziellen Produkten stellt Software-Hersteller vor die Herausforderung, Betriebs- und IT-Sicherheit für fremden Source Code technisch sicherzustellen und haftungsrechtlich zu verantworten. Die formalen Abläufe gängiger Softwareentwicklungsprozesse eignen sich hierbei dafür, die Aufgabe des Schwachstellen-Monitorings und die Behebung von Schwachstellen zu integrieren. Untersucht wird die Eignung linearer und agiler Entwicklungsmethoden für eine Erweiterung durch Schwachstellenanalyse von eingesetzter Open Source Software. Im Vordergrund stehen hierbei insbesondere Wirksamkeit und Aufwand. Aus juristischer Sicht stellt sich die Frage, ob der Software-Hersteller für schwachstellenbehaftete Software gegenüber seinen Kunden haftet. Dies ist v.a. vor dem Hintergrund des bei fast allen Open Source Lizenzen vereinbarten vollständigen Ausschlusses von Gewährleistungs- und Haftungsansprüchen bedeutend. Es wird in diesem Zusammenhang erläutert, welche gewährleistungsrechtlichen Verpflichtungen für Hersteller von kommerziellen Software-Produkten bestehen und inwieweit der Hersteller gegenüber seinen Kunden haftungsrechtlich verantwortlich ist.

1 Einleitung

Free and Open Source Software (kurz: FOSS) ist in der modernen Softwareentwicklung nicht mehr wegzudenken. Waren es in den 90iger Jahren noch überwiegend private Nutzer und akademische Institutionen die auf die Vorteile der freien Softwarelizenzen gesetzt haben, so besteht heutzutage auch ein wesentlicher Anteil kommerzieller Entwicklungen aus Open Source:

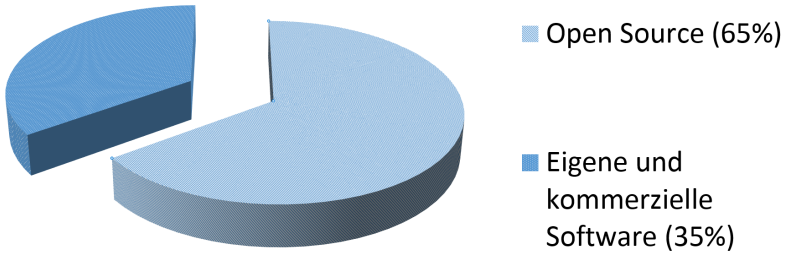

Abb. 1: Anteil Open Source an gewerblichen Softwareentwicklungen in 2016[1]

[1] Quelle: https://www.blackducksoftware.com

P. Schartner · A. Baumann (Hrsg.) · D•A•CH Security 2017 · syssec (2017) 238-249.

In der Praxis wird FOSS häufig in kommerziellen Produkten eingesetzt. Der FOSS-Verwender (kurz: Verwender) bedient sich dazu einzelner Komponenten, die von FOSS-Rechteinhabern (kurz: Rechteinhaber) programmiert wurden. Anschließend wird das Produkt – zumeist kostenpflichtig – vertrieben. Immer wieder relevant werden in diesem Zusammenhang das Vorhandensein von Sicherheitslücken und die möglichen juristischen Folgen sowohl für den Rechteinhaber wie auch den Verwender.

Die Feststellung und Behebung von Schwachstellen in Softwareprodukten ist oft ein kostspieliger und zeitraubender Prozess. Insbesondere der große Anteil an Open Source Software stellt die Hersteller vor die Aufgabe für effiziente Möglichkeiten der Erkennung, Analyse und der Risikomitigierung zu sorgen.

Die meisten professionellen Softwarehersteller haben bereits einen eingeführten Entwicklungsprozess. Die Nutzung bestehender Prozesse um die Schwachstellenanalyse zu erweitern liegt dann nah. Von allen Dingen eignen sich hierbei die Phasen der manuellen und automatisierten Tests für eine Erweiterung.

Die Verwendung von FOSS in kommerziellen Produkten wirft daneben zahlreiche juristische Fragestellungen auf, die von erheblicher Brisanz für Unternehmen sein können. Dies gilt insbesondere auch bei der Verwendung von FOSS, die Schwachstellen – z.B. in Form von Sicherheitslücken – aufweisen. In diesen Fällen stellt sich die Frage nach der **Haftung für Schäden**, die aufgrund fehlerhafter FOSS eingetreten sind. Softwarefehler können zu erheblichen Schäden führen, insbesondere wenn sie über einen längeren Zeitraum nicht entdeckt werden ([Schn17]: Kapitel M Rn. 1299). Zudem hat der Hersteller von Software in Hinblick auf sein Produkt zahlreiche Verpflichtungen, insbesondere **Instruktionspflichten**, **Produktbeobachtungspflichten** sowie **Gefahrnabwendungspflichten**. Bei letzterer kommen insbesondere die Warnung, der Rückruf des Produkts sowie die Beseitigung der Schwachstelle in Betracht.

2 Technische Betrachtung

Ein typisches Produkt der Softwareentwicklung besteht aus einer Vielzahl von Komponenten aus unterschiedlichen Quellen. Hierbei unterliegen Eigenentwicklungen im Idealfall bereits bestehenden Sicherheitsmaßnahmen, wie vorgegebener Coding Rules oder Evaluierungs- und Testprozesse, die als fester Bestandteil der Qualitätssicherung auch die IT-Sicherheit berücksichtigen. Der inzwischen größere Anteil (siehe Einleitung) kommt aus Quellen Dritter. Hierbei ist es sinnvoll, zwischen kommerziell beschaffter Software (Commercial Of The Shelve, kurz: COTS) und FOSS zu unterscheiden.

Bei COTS ist man in besonderem Maße auf den Support des Herstellers angewiesen, da die Software meist nur als „Closed Source“ zur Verfügung gestellt wird und die vertraglichen Bedingungen häufig kein Reverse Engineering erlauben. Hier bieten die Hersteller häufig Serviceverträge an, die die Durchführung von (Sicherheits-)Updates, Bugfixing etc. zum Inhalt haben.

FOSS hingegen wird ohne Gewinnerzielungsabsichten durch deren Urheber, Beitragende oder den Verwender veröffentlicht. Die Software kann (und wird) Schwachstellen und Fehler enthalten, die mit Integration und Weitergabe Teil des eigenen Produktes werden. Die Herausforderung hierbei ist, mögliche Sicherheitsrisiken vor Auslieferung zu erkennen und zu beheben.

Funktionales Fehlverhalten (Bugs) wird z.B. im Rahmen von Software-Integrationstests festgestellt und kann anschließend durch Anpassungen am Quellcode oder der Konfiguration behoben werden. Der offene Quellcode und die in FOSS-Lizenzen obligatorische Erlaubnis zur Modifikation und Distribution schaffen hierbei die notwendige Voraussetzung.

Zur verlässlichen Feststellung von IT-Schwachstellen (IT Vulnerabilities) hingegen sind automatisierte Verfahren (z.B. statische Codeanalyse) meist nicht ausreichend. Code Reviews und manuelle Analysen sind erforderlich, um festgestellte Problemfälle zu verifizieren (False Positive) und weitere Problemfälle zu finden (False Negative). Eine ausschließlich automatische Detektion und Bewertung ist häufig nicht ausreichend.

Dem gegenüber steht der große Anteil an quelloffener Software in aktuellen Softwareprodukten. Eine exakte und vollständige Analyse komplexer Architekturen ist vielfach zu zeitaufwändig, insbesondere vor dem Hintergrund regelmäßiger Updates, Anpassungen und Erweiterungen.

Es muss daher ein Weg gefunden werden, IT-Sicherheitsrisiken zu minimieren, dabei aber auch eine effiziente Vorgehensweise zu unterstützen. Ein Ansatz hierbei ist, das IT Security Monitoring zum Teil des Entwicklungsprozesses (EW-Prozess) zu machen. Gängige Softwareentwicklungsverfahren eignen sich dabei grundsätzlich zur Behandlung von IT-Sicherheitsrisiken in FOSS.

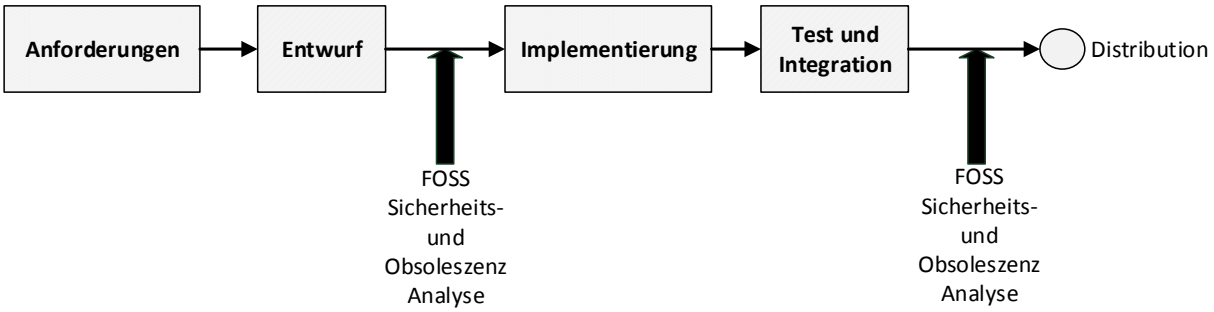

Abb. 2: FOSS Analyse im linearem EW-Prozess

Bei linearen Entwicklungsabläufen[2], wie sie insbesondere im Behördenumfeld für Auftragsentwicklungen gefordert werden, werden die Phasen der Entwicklung nacheinander einmalig durchlaufen. D.h. jeder Folgeabschnitt baut auf dem Vorhergehenden auf. In einer vereinfachten Darstellung sind das die folgenden Phasen:

Definition der Anforderungen: Mittels Requirement Engineering werden die Kundenforderungen an das zu erstellende System festgelegt und in eine syntaktisch widerspruchsfreie, eindeutige und atomisierte Form gebracht. Kennzeichnend bei dieser Vorgehensweise ist, dass alle Anforderungen an das System in dieser Phase festgelegt werden. Hierbei werden auch die Anforderungen zur Einhaltung der Informationssicherheitsziele formuliert.

Entwurf: Auf Basis der Anforderungen findet ein Systementwurf in Form einer Architektur und eines Designs statt. Hierbei werden die Entscheidungen zur Realisierung getroffen. Dies umfasst u.a. Wahl und Herkunft eingesetzter Softwarekomponenten, Schnittstellen untereinander sowie externer Verbindungen. Für geplante FOSS werden auf Basis von Name und Version

[2] Das Konzept des Wasserfallmodells: http://cartoon.iguw.tuwien.ac.at/fit/fit01/wasserfall/konzepte.html oder das V-Modell als Beispiel für ein lineares Entwicklungsverfahren: http://ftp.uni-kl.de/pub/v-modell-xt/Release-1.1-eng/Dokumentation/pdf/V-Modell-XT-eng-Teil1.pdf

regelmäßige Überprüfungen in Schwachstellendatenbanken wie z.B. der „National Vulnerability Database" (https://nvd.nist.gov/) durchgeführt. Komponenten mit einsatzkritischen Schwachstellen können somit frühzeitig identifiziert und ersetzt werden. Weiteres Kriterium ist die Verbreitung und Herkunft einer FOSS-Komponente. Eine vertrauenswürdige Quelle sowie eine nachweislich große Bekanntheit und häufige Nutzung sind Indikatoren für sichere Software. Spätestens vor Beginn der Implementierung sollten zusammen mit dem Critical Design Review (CDR)[3] auch die Einhaltung der IT-Sicherheitsanforderungen im aktuellen Entwurf bewertet werden.

Implementierung: Mit der Entwicklung erfolgt die Realisierung des Designs. Es werden proprietäre Anteile implementiert und geplante externe Komponenten (FOSS und COTS) zum eigenen Code hinzugelinkt und ggf. modifiziert. Da sich die Implementierungsphase über Monate oder länger hinziehen kann, ist eine regelmäßige Überprüfung der eingesetzten Versionen auf neue Schwachstellen sinnvoll um zeitnah auf neue mögliche Bedrohungen reagieren zu können.

Test und Integration: Zusammen mit dem Design werden Systemtests definiert, die den Nachweis der Umsetzung aller Anforderungen belegen. Bereits während der Entwicklung wird die Funktionalität von Modulen, Komponenten und des Gesamtsystems durch automatisierte Tests überprüft. Hierbei werden u.a. auch Testfälle berücksichtigt, die IT-sicherheitsrelevante Aspekte innerhalb der proprietären Codierung überprüfen.

Kurz vor Auslieferung erfolgt eine weitere abschließende Sicherheits- und Obsoleszenz Prüfung enthaltener FOSS.

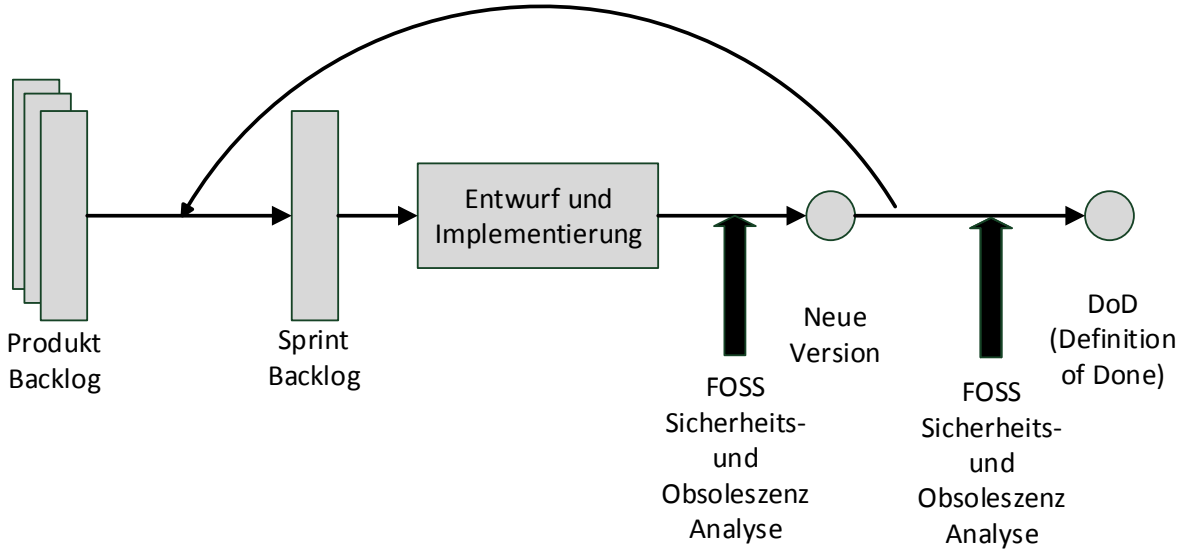

Abb. 3: FOSS Analyse im agilen EW-Prozess

Iterative Softwareentwicklungsverfahren wie z.B. „SCRUM", nähern sich in wiederholenden Schritten einem abgestimmten Zwischenstand oder einer fertigen Lösung. Hierbei werden die dazu notwendigen Phasen wie Anforderungsanalyse, Design, Implementierung und Test für

[3] Meilenstein, der das Ende der Designphase und den Beginn der Implementierung kennzeichnet s.a. http://people.ucalgary.ca/~design/engg251/First%20Year%20Files/design_reviews.pdf

jede Erweiterung und Optimierung erneut durchlaufen. Vorteil sind das Auflösen der Komplexität eines geplanten Systems, die Fokussierung auf Transparenz durch Dokumentation und gegenseitige Kontrolle sowie die fortlaufende Überprüfung der Inkremente.

Innerhalb einer Iteration, dem sog. „Sprint", werden neben der Entwicklung und Dokumentation auch die erforderlichen Tests durchlaufen. Die iterative Vorgehensweise begünstigt einen toolgestützten Erstellungs- und Testprozess mit (teil-)automatisierten Prozessschritten. Abhängige FOSS kann so identifiziert und mögliche Schwachstellen mittels forensischer Analysewerkzeuge ermittelt werden.

Vor Auslieferung der Software (Definition of Done) wird in einem abschließenden (manuellen) Sicherheits- und Obsoleszenz Check die Integrität bestätigt.

Regelmäßig durchgeführte Reviews durch die beitragenden Entwickler unterstützen den IT-Sicherheitsprozess durch gegenseitige Kontrolle der eigenen Implementierungen. Somit werden Abweichungen von den vereinbarten Coding Rules und mögliche Schwachstellen oder Fehler im Code frühzeitig erkannt.

3 Juristische Bewertung

Der in zahlreichen FOSS-Lizenzen vereinbarte (vollständige) Haftungsausschluss darf einen hier nicht in falscher Sicherheit wägen. Er verleitet schnell zu der Annahme, dass die Verwendung nicht nur kostenlos ist, sondern auch für Fehler nicht eingestanden werden muss. Nach deutschem Recht ist ein so weitgehender **Haftungsausschluss regelmäßig unwirksam.** Bei Individualverträgen ist eine Begrenzung bis zur Grenze des Vorsatzes zulässig (§ 276 Abs. 3 BGB). Bei Allgemeinen Geschäftsbedingungen ist ein Ausschluss der Haftung für grobes Verschulden unzulässig (§ 309 Nr. 7 lit. b BGB). Besonders misslich ist die Lage, wenn der Rechteinhaber seinen gewöhnlichen Aufenthalt in den USA hat und im Rechtsverhältnis zu dem Verwender **US-amerikanisches Recht** anwendbar ist. Denn nach US-Recht dürfte ein derart umfangreicher Haftungsausschluss zumeist wirksam sein.[4]

Nach der **ROM-I-Verordnung** kann das anwendbare nationale Recht grundsätzlich von den Parteien selbst bestimmt werden [ScLo17: Vor § 120 ff; Scha15: Rn. 1285].[5] Fehlt es an einer expliziten Rechtswahl[6] durch die Parteien, insbesondere wenn die betroffene FOSS-Lizenz hierzu keine Regelung enthält,[7] ist bei FOSS-Lizenzverträgen das Recht desjenigen Staates an-

[4] Zusammenfassung der unterschiedlichen Behandlung der einzelnen Staaten der USA: https://www.pillsburylaw.com/images/content/1/8/v2/1875/0C62DFD605F0471619ADF0E2E5576E98.pdf. Die meisten Staaten billigen entsprechende Disclaimer, solange sie auffällig gestaltet sind und keine Übervorteilung darstellen. Letzteres dürfte gerade bei FOSS nicht vorliegen, da diese kostenlos angeboten wird.

[5] Eine explizite Rechtswahl taucht in einigen FOSS-Lizenzen auf, so z.B. in der Mozilla Public License (MPL) v1.1, der Eclipse Public License (EPL) v1.0. Auch die neuere MPLv2.0 enthält eine Rechtwahl- sowie Gerichtsstandsklausel, wobei sich hier das anwendbare Recht flexibel nach dem Wohnsitz bzw. Geschäftssitz des Beklagten richtet. Bei Verbraucherverträgen ist Art. 6 ROM-I-VO zu beachten.

[6] Der BGH hat eine entsprechende Rechtswahlklausel zugunsten des US-Rechts, die mit einer Gerichtsstandsklausel verbunden war, für unwirksam erachtet, da dadurch zwingende nationale bzw. europäische Schutzvorschriften umgangen würden, s. Beschluss vom 05.09.2012, Az. VII ZR 25/12.

[7] Aus der Tatsache, dass die meisten FOSS-Lizenzverträge in englischer Sprache abgefasst sind, kann allein noch nicht von der Anwendbarkeit des US-amerikanischen Rechts ausgegangen werden [JaMe16: Rn. 362; Deik03: 11; Schi03: 109; Sest00: 802; Spin04: 141].

wendbar, in dem Vertragspartner mit der charakteristischen Leistung seinen gewöhnlichen Aufenthalt hat. Bei FOSS-Lizenzverträgen wird diese charakteristische Leistung in der (kostenlosen) Einräumung von Nutzungsrechten zu sehen sein, so dass bei Rechtsstreitigkeiten das Recht des Staates einschlägig ist, in dem der Lizenzgeber seinen Aufenthalt hat. Deutsches Recht kommt i.d.R. nur dann zur Anwendung, wenn der Rechteinhaber in Deutschland seinen gewöhnlichen Aufenthalt hat.

Die Frage des anwendbaren Rechts hat unmittelbare Auswirkungen auf die Gewährleistungs- und Haftungsansprüche des Lizenznehmers bei FOSS. Während die Haftung damit zwischen Rechteinhaber und Verwender, z.B. aufgrund der Anwendbarkeit US-amerikanischen Rechts, tatsächlich ausgeschlossen sein kann, wäre ein solcher Ausschluss im Verhältnis zwischen dem Verwender und dem Nutzer regelmäßig unwirksam; dies gilt insbesondere bei Verträgen mit Verbrauchern. Denn FOSS-Lizenzen sind nach dt. Recht als Allgemeine Geschäftsbedingungen einzuordnen [Deik03: 13; Koch00: 339; JaMe16: 179].[8]

Ob und in welcher Form dem Verwender bzw. dem Nutzer Gewährleistungs- und weitere Haftungsansprüche zustehen, bemisst sich zunächst nach den vertraglichen Vereinbarungen zwischen den Parteien. Allerdings führt v.a. die Bestimmungen des BGB für Allgemeine Geschäftsbestimmungen (AGB) dazu, dass zu weitreichende Begrenzungen der gesetzlich vorgesehenen Ansprüche unwirksam sind. Die entsprechenden Vorschriften sind in den §§ 305 ff. BGB zu finden. Sie sind grundsätzlich sowohl auf Verbrauchergeschäfte als auch Geschäfte unter Unternehmern anwendbar.

Die charakteristische Vertragsleistung bei FOSS liegt in der kostenlosen Überlassung von Software und der gleichzeitigen Einräumung von umfangreichen einfachen Nutzungsrechten. Der FOSS-(Lizenz-)Vertrag wird demnach von der herrschenden Meinung in der Literatur als **Schenkungsvertrag verbunden mit einem Lizenzvertrag mit schenkungsrechtlichen Elementen** angesehen [MeJa99: 847; Deik03: 14; a.A.: Koch00: 335; Witz16: 163; JaMe16: Rn 205 Fn 923]. Der Vertrieb einer proprietären Software, die (auch) FOSS des Rechteinhabers beinhaltet, ist bei einer kostenpflichtigen Überlassung dem **Kaufrecht** und bei der Überlassung auf Zeit dem **Mietrecht** zuzuordnen, bei Überlassung z.B. im Rahmen von Implementierungsprojekten liegt ggf. ein Werkvertrag vor [Witz16: 163; Rede12: Rn 595a ff.].

Bei einer **Schenkung** ist die Haftung des Schenkers aufgrund der Uneigennützigkeit und Großzügigkeit des Schenkers stark eingegrenzt. ***Grundsätzlich*** haftet der Schenker nur für grobe Fahrlässigkeit und Vorsatz. Bei ***Mängeln an dem Gegenstand der Schenkung*** – hier der Software – bestehen Ansprüche nur bei Arglist. Auch bei Schäden an anderen Sachen als der Software, die aufgrund des Mangels eingetreten sind, bestehen Haftungsansprüche – abgesehen von sonstigen Nebenpflichtverletzungen – nur bei Arglist des Schenkers.[9] Eine weitere Begrenzung der Haftung des Schenkers in Individualverträgen ist bis zur Grenze des Vorsatzes möglich, bei AGB ist jedoch ein vollständiger oder auf einzelne Teile (z.B. Sicherheitslücken) bezogener Ausschluss von Gewährleistungsrechten sowie ein Ausschluss der Haftung für grob fahrlässiges Verhalten unwirksam.

Bei einer Schenkung von Software, die mangelbehaftete FOSS-Komponenten beinhaltet, haften der Rechteinhaber als auch der Verwender nur im Falle der Arglist, soweit es vertragliche Ansprüche betrifft. Ein Fall der Arglist könnte vorliegen, wenn der Rechteinhaber/Verwender

[8] vgl. zur GPLv2.0 die Entscheidung des LG München I vom 19.05.2004, Az. 21 O 6123/04, MMR 2004, 693.

[9] BGH, Urteil vom 20.11.1984, Az. IVa ZR 104/83 – Kartoffelpülpe-Fall.

in bewusster Kenntnis von (schwerwiegenden) Sicherheitslücken seine Software weiter unter Verwendung dieser Komponente entwickelt oder aber die Software vertreibt und den Nutzer nicht über diesen Umstand informiert.

Wird die FOSS-Komponente von einem Software-Hersteller in eine proprietäre Software eingebunden und gemeinsam kostenpflichtig Kunden überlassen (etwa im Rahmen eines ERP-Implementierungsprojekts), findet wohl Werk- bzw. Kauf- oder Mietrecht Anwendung (ggf. auch in Kombination). Wird die Software **verkauft**, so muss diese in einem mangelfreien Zustand geliefert werden. Ein Mangel liegt danach vor, wenn Beschaffenheit einer Kaufsache von der vereinbarten Beschaffenheit[10] abweicht bzw. in Ermangelung einer solchen Vereinbarung sich die Sache nicht für die nach dem Vertrag vorausgesetzte Verwendung[11] oder aber für die gewöhnliche Verwendung[12] eignet. Der BGH legt den Begriff des Mangels weit aus: *„Als Beschaffenheit einer Kaufsache im Sinne von § 434 Abs. 1 BGB sind sowohl alle Faktoren anzusehen, die der Sache selbst anhaften, als auch alle Beziehungen der Sache zur Umwelt, die nach der Verkehrsauffassung Einfluss auf die Wertschätzung der Sache haben."*[13] Die Besonderheit bei Software ist, dass sie als komplexes Werk fast nie absolut fehlerfrei im technischen Sinne sein wird [Lehm92: 1725; Taeg96; Mart93: 18 f]. Allerdings bedeutet eine kleine, technische Fehlfunktion (z.B. die Tatsache, dass eine Software z.B. nicht manipulationssicher ist) nicht zugleich die Annahme eines Sachmangels im juristischen Sinne.[14] Eine bestehende Sicherheitslücke wird in der Regel aber einen Sachmangel darstellen, da hier eine nicht unerhebliche Abweichung von der Ist- zu der Soll-Beschaffenheit vorliegt.[15] Die Eignung zur vertraglich vorausgesetzten Verwendung sowie die Eignung zur gewöhnlichen Verwendung vor dem Erwartungshorizont des Kunden sind bei bestehenden Sicherheitslücken in jedem Fall zweifelhaft. Dies gilt insbesondere dann, wenn der Kunde einen Telemediendienst anbietet (vgl. § 13 Abs.

[10] Es bedarf hier einer ausdrücklichen oder zumindest konkludenten Abrede zwischen den Vertragsparteien. Ein ausdrücklicher Ausschluss von Sicherheitslücken als Mängel dürfte in der Praxis jedoch nicht vorkommen. Anknüpfungspunkte für eine Vereinbarung über Sicherheitslücken bieten u.a. § 13 Abs. 7 TMG und § 8a BSIG (letztere derzeit noch nicht relevant aufgrund einer zweijährigen Übergangsregelung).

[11] An einer Eignung zur vertraglich vorausgesetzten Verwendung wird es insb. fehlen, wenn die Software geltenden gesetzlichen Anforderungen nicht entspricht (siehe z.B. OLG Hamm, NJW-RR 1995, 941: fehlende Berücksichtigung der Bilanzrichtlinien). Dies gilt v.a. hinsichtlich solcher Rechtsvorschriften, die unmittelbar für den Anbieter bzw. sein Produkt gelten (vgl. BGH NJW 1985, 1769 zum GerätesicherheitsG; OLG München, NJW-RR 1992, 1523 zur Nichteinhaltung von DIN- und VDA-Bestimmungen; LG Aachen, BeckRS 2010, 16836 zum Geräte- und ProduktsicherheitsG). Gleiches gilt auch, wenn der Anbieter Kenntnis davon hat, dass der Kunde im Rahmen der Nutzung der Software bestimmte rechtliche Vorgaben einzuhalten hat.

[12] Die übliche Beschaffenheit von Software kann sich aus der Natur des Produkts als auch aus den rechtlichen Anforderungen an die Software ergeben.

[13] BGH, Urteil vom 15. Juni 2016 - VIII ZR 134/15.

[14] OLG München, Urteil vom 17.03.1987 – Az. 5 U 3879/86, CR 1989, 295: „Eine wahlweise zuschaltbare Mehrfunktion, bei der das korrekte Programmergebnis leichter als bei anderen Systemen manipuliert werden kann, stellt grundsätzlich keinen Fehler dar, es sei denn, die Parteien hätten gerade Manipulationssicherheit vereinbart."

[15] Wenn der Software-Hersteller sein Software-Produkt im virenfreien Zustand ausliefern muss, so lässt sich dies auch auf vorliegende Sicherheitslücken übertragen, vgl. LG Regensburg, Urt. v. 17.6.1997 – 2 S 168/96, CR 1997, 686; LG Kleve, Urt. v. 29.6.1995 – 7 O 17/95, CR 1996, 292. Es gibt Stimmen, die davon ausgehen, eine fehlerfreie Software könne von vornherein nicht erwartet werden: [Baue89: 38; Hons95: 212]; im Ergebnis aber kritisch zu sehen. Insbesondere muss der vereinbarte Leistungsumfang eingehalten werden; differenziert auch bei Sachmängeln zu betrachten. Kritisch vor dem Hintergrund der möglichen starken Beeinträchtigung des Integritätsinteresses.

7 TMG[16]), eine sog. kritische Infrastruktur betreibt (vgl. § 8a Abs. 1 BSIG) oder in Zukunft als „Anbieter digitaler Dienste“[17] anzusehen ist (vgl. § 8c BSIG-E[18]) und dies dem Softwareanbieter bekannt ist.

Sofern die Software beim Kauf einen Sachmangel in Form einer Sicherheitslücke aufweist, stehen dem Nutzer Gewährleistungsrechte zu. Der Kunde des Verwenders hat bei Vorliegen eines Sachmangels zunächst einen Nachbesserungsanspruch auf Beseitigung der Sicherheitslücke. In Betracht kommt der Rückgriff auf eine neuere (oder aber auch ältere) Version der Komponente ohne den besagten Fehler sowie die Verwendung einer alternativen Komponente mit ähnlichen Eigenschaften, die ebenfalls den Ablauf der Software ermöglicht. Ggf. ist auch ein vollständiger Verzicht auf die problematische Komponente (technisch) möglich. In diesem Zusammenhang stellt sich dann die Frage, ob durch die Nachbesserung des FOSS-Verwenders ein neuer Mangel an der Software entsteht, etwa in Form einer Funktionseinschränkung oder anderer negativer Auswirkungen [Horn17: 289]. Der Anspruch auf Nachbesserung ist verschuldensunabhängig. Der Hersteller kann eine Nacherfüllung nur ablehnen, wenn diese nur mit unverhältnismäßigen Kosten möglich ist. Ob dies der Fall ist, hängt nach dem Gesetz von dem Wert der Software in mangelfreien Zustand sowie der Schwere und der (möglichen) Folgen des Mangels ab.

Daneben stehen ihm Schadensersatzansprüche für solche Schäden zu, die im Zusammenhang mit dem Mangel eingetreten sind. Anzuführen ist hier insbesondere der Betriebsausfallschaden. Voraussetzung für Schadensersatzansprüche ist, dass der Verwender die Lieferung der mit Mängel behafteten Software zu vertreten hat. Grundsätzlich hat der Verwender Vorsatz und (jede Form der) Fahrlässigkeit zu vertreten. Fahrlässig handelt gemäß § 276 Abs. 2 BGB, wer die im Verkehr erforderliche Sorgfalt außer Acht lässt. Das Vertretenmüssen wird vom Gesetz vermutet.[19] Der Verwender muss daher aktiv den Nachweis erbringen, dass er für den Mangel nicht einzustehen hat. Ein möglicher Nachweis des Verwenders kann dadurch erfolgen, dass er im Streitfall darlegen kann, die Software unter Einsatz von anerkannten Tools bzw. Methoden auf (bekannte) Schwachstellen hin untersucht zu haben. In Hinblick auf die Virenfreiheit wurde eine solche Untersuchungspflicht bereits von der Rechtsprechung angenommen (s. LG Regensburg, Urt. v. 17.6.1997 – 2 S 168/96, CR 1997, 686; LG Kleve, Urt. v. 29.6.1995 – 7 O 17/95,

---

[16] Dort heißt es u.a.: *„Diensteanbieter haben (…) für geschäftsmäßig angebotene Telemedien durch technische und organisatorische Vorkehrungen sicherzustellen, dass (1.) kein unerlaubter Zugriff auf die für ihre Telemedienangebote genutzten technischen Einrichtungen möglich ist und (2.) diese (…) b) gegen Störungen, auch soweit sie durch äußere Angriffe bedingt sind, gesichert sind. (…)“*.

[17] Also solche zählen Anbieter von Online-Marktplätzen, Online-Suchmaschinen und Cloud-Computing-Diensten. Nach Schätzung des Bundesministeriums des Innern sind gegenwärtig zwischen 500 und 1.500 Unternehmen in Deutschland hiervon betroffen (siehe http://hoganlovells-blog.de/2017/01/27/umsetzung-der-nis-richtlinie-neue-pflichten-fuer-anbieter-digitaler-dienste/#).

[18] In dem Entwurf eines Gesetzes zur Umsetzung der Richtlinie (EU) 2016/1148 des Europäischen Parlaments und des Rates vom 6. Juli 2016 über Maßnahmen zur Gewährleistung eines hohen gemeinsamen Sicherheitsniveaus von Netz- und Informationssystemen in der Union vom 25.01.2017 heißt es in § 8c Abs. 1 BSIG-E: *„Anbieter digitaler Dienste haben geeignete und verhältnismäßige technische und organisatorische Maßnahmen zu treffen, um Risiken für die Sicherheit der Netz- und Informationssysteme, die sie zur Bereitstellung der digitalen Dienste innerhalb der Europäischen Union nutzen, zu bewältigen.* ***Sie haben Maßnahmen zu treffen, um den Auswirkungen von Sicherheitsvorfällen*** *auf innerhalb der Europäischen Union erbrachte digitale Dienste* ***vorzubeugen oder die Auswirkungen so gering wie möglich zu halten.****“*.

[19] Die Vermutung ergibt sich aus der Formulierung in § 280 Abs. 2 Satz 2 BGB: „Dies [Verpflichtung zum Ersatz des Schadens] gilt nicht, wenn der Schuldner die Pflichtverletzung nicht zu vertreten hat.“.

CR 1996, 292). Die genannten Ansprüche aus dem Gewährleistungsrecht können i.d.R. bis zwei Jahre ab Ablieferung der Sache geltend gemacht werden.

Die entscheidende Frage ist nun, wie der FOSS-Urheber sowie der FOSS-Verwender auf vorhandene Sicherheitslücken in FOSS-Komponenten reagieren muss. Es ist dabei in zweierlei *Hinsicht zu **differenzieren***. Zunächst ist **(1.)** abzugrenzen zwischen der Zeit bis zur erfolgten Verbreitung der Software (an den Nutzer) und der Phase der sich anschließenden Verwendung der Software durch den Nutzer. Weiter ist **(2.)** zu unterscheiden, ob der FOSS-Rechteinhaber bzw. der FOSS-Verwender positive Kenntnis von der Sicherheitslücke hat, diese ihm in jedem Fall bekannt gewesen sein müsste (Fahrlässigkeit) oder aber sie nur wenigen Eingeweihten bekannt war (ggf. nicht verantwortlich). Je nach Sachverhalt ergeben sich für den FOSS-Urheber/-Verwender unterschiedliche Gewährleistungs- und Haftungsverpflichtungen.

Wenn das Leib, Leben, Gesundheit, Eigentum oder ein sonstiges absolutes Rechtsgut durch die mit Sicherheitslücken belastete Software verletzt wird, besteht ein Schadensersatzanspruch des Geschädigten nach dem sog. **„Deliktsrecht“**. Auch hier reicht – wie bei den vertraglichen Ansprüchen – eine fahrlässige Verletzung aus. Auch der Datenverlust beim Nutzer kann als Eigentumsverletzung angesehen werden [MeWe98: 1588 f.].

Den Hersteller treffen sog. Verkehrssicherungspflichten. Sowohl bei Konstruktion, Produktion als auch Instruktion muss sich der Hersteller nach dem erkennbaren und ermittelbaren Stand von Wissenschaft und Technik richten. In diesem Zusammenhang bestehen für den Hersteller **Organisationspflichten**, **Instruktionspflichten**, **Produktbeobachtungspflichten** sowie **Gefahrenabwendungspflichten**. Letztere teilt sich auf in die Pflicht zur Warnung, zum Rückruf sowie zur Beseitigung der Gefährdung. Hat der Hersteller einer Software erkannt, dass die Verwendung dieser, z.B. aufgrund vorhandener Sicherheitslücken, zu einer Gefährdung von Leib, Leben, Gesundheit, Eigentum oder eines sonstigen absoluten Rechtsguts führen kann, so müssen entsprechende Schritte zur Gefahrenabwendung eingeleitet werden. Aber auch bereits zum Zeitpunkt der Programmierung der Software ist darauf zu achten, dass bekannte Sicherheitslücken vermieden werden. Zudem ist in diesem Zusammenhang das Geräte- und Produktsicherheitsgesetzt[20] zu beachten, dass auch auf Software Anwendung findet. Hieraus ergeben sich Verpflichtungen des Herstellers zur vorbeugenden Installation eines Risikomanagements [Litt05: 457; ZsLu05: 499].

Im Rahmen einer Schenkung von FOSS-Software gilt die Haftungsbeschränkung des Schenkungsrecht nach §§ 521 ff. BGB auch für deliktische Ansprüche.[21] Von daher hat der Rechteinhaber/Verwender in solchen Fällen in der Regel nur für Arglist bei Schäden aufgrund eines Sachmangels, ansonsten für grobe Fahrlässigkeit einzustehen. Im Kaufrecht gibt es keine vergleichbare Haftungserleichterung.

Im Rahmen des **Produkthaftungsgesetzes** (kurz: ProdHaftG) ist der Hersteller eines fehlerhaften Produktes zum Schadensersatz verpflichtet, wenn durch den Fehler jemand getötet, sein Körper oder seine Gesundheit verletzt wird oder eine Sache beschädigt wird. Relevant kann dies v.a. beim Einsatz von FOSS in *„embedded systems“* werden, z.B. im Flugzeug, im Auto etc. Auch Standardsoftware in isolierter Form unterfällt nach der Rechtsprechung dem Produktbegriff, da sie auf einem Datenträger verkörpert werden kann [Bart89: 142; Tasc86: 84; Junk03: Rn. 478 ff.; Hoer88: 119; Hoer89: 30 f.; Junk88]. Bei individuell erstellter Software ist die

[20] Das Geräte- und Produktsicherheitsgesetz gilt als Schutzgesetz i.S.d. § 823 Abs. 2 BGB.

[21] BGH, Urteil vom 20.11.1984, Az. IVa ZR 104/83 – Kartoffelpülpe, Leitsatz 2, VersR 1985, 278.

Sacheigenschaft von Software umstritten [Junk03: Rn. 480]. Damit ein Anspruch besteht, muss ein Fehler des Produkts vorliegen. Die Definition des Fehlers nach dem ProdHaftG weicht von der des Mangelbegriffs ab und kann nicht gleichgesetzt werden. Aber eine Sicherheitslücke kann in jedem Fall die Sicherheit des Produkts beeinträchtigen.

Das ProdHaftG führt jedoch bei Schäden *an anderen Sachen als dem Produkt* des Herstellers nur dann zu einer Haftung, wenn die andere Sache gewöhnlicher Weise privat genutzt wird. Es ist damit bei Sachschäden nur im privaten Bereich von Relevanz und findet daher im Softwarebereich nur wenig Anwendung. Zudem muss der Geschädigte bei einer Sachbeschädigung eine Selbstbeteiligung in Höhe von EUR 500,- tragen. Die Ersatzpflicht des Herstellers aus dem ProdHaftG darf allerdings vertraglich nicht ausgeschlossen werden.

Neben dem allgemeinen ProdHaftG gibt es noch spezielle Gesetze, die bestimmten Personen besondere Pflichten im Bereich der IT-Sicherheit auferlegen. Anzuführen ist hier insbesondere das **Gesetz über das Bundesamt für Sicherheit in der Informationstechnik** (BSIG). Nach § 8a Abs. 1 BSIG müssen Betreiber *Kritischer Infrastrukturen „angemessene organisatorische und technische Vorkehrungen zur Vermeidung von Störungen der Verfügbarkeit, Integrität, Authentizität und Vertraulichkeit ihrer informationstechnischen Systeme, Komponenten oder Prozesse zu treffen, die für die Funktionsfähigkeit der von ihnen betriebenen Kritischen Infrastrukturen maßgeblich sind."* Unter den Begriff der *„Kritischen Infrastruktur"* fallen nach § 2 Abs. 10 BSIG v.a. die Sektoren Energie, Informationstechnik und Telekommunikation, Transport und Verkehr, Gesundheit, Wasser, Ernährung sowie Finanz- und Versicherungswesen sowie solche Infrastrukturen von *„von hoher Bedeutung für das Funktionieren des Gemeinwesens"*. Darunter können auch Telemedien- und Cloud-Anbieter fallen. Die Verpflichtungen nach dem BSIG richten sich zwar nicht direkt an die Software-Hersteller, jedoch kann das Bundesamt für Sicherheit in der Informationstechnik (BSI) von den Herstellern bei Sicherheitslücken Abhilfe verlangen (vgl. § 8b Abs. 6 BSIG).

4 Ergebnis

Der Einsatz von Open Source Software in kommerziellen (Software-)Produkten ist sowohl in technischer als auch juristischer Hinsicht für den Hersteller eine Herausforderung. Obschon eine Vielzahl von Herstellern auf FOSS setzen, sind vielen von Ihnen die bestehenden Risiken nicht (hinreichend) bekannt. Dies gilt insbesondere auch in Hinblick auf Sicherheitslücken.

Unternehmen, deren Software-Produkte Open Source Software enthalten, sollten bereits frühzeitig das IT-Security-Monitoring zum festen Bestandteil ihrer Wertschöpfungs- und Beschaffungskette machen. Das Monitoring in Eigenentwicklungen wird Teil des Erstellungsprozesses, wobei Meilensteine im linearen Verfahren und Iterationsschleifen im agilen Prozess sinnvolle Aufsetzpunkte für die IT Sicherheitschecks bilden. Eine aktuelle Dokumentation und gegenseitige Kontrollen des Quellcodes unterstützen hierbei.

Für Produktbestandteile kommerzieller Dritter, die ebenfalls FOSS enthalten, gilt: Transparenz und Support sind hier ebenso wichtig, wie für proprietäre SW mit FOSS. Daher ist das Procurement gehalten, hier die notwendigen Informationen von den Lieferanten einzuholen.

Nach Auslieferung und Überführung in die Nutzung ist es sinnvoll, im Rahmen von Supportvereinbarungen zwischen Hersteller/Lieferant und Kunde regelmäßige Updates von schwachstellen-bereinigter FOSS einzuspielen. Das Monitoring von bereits im Einsatz befindlicher

FOSS kann z. Bsp. im Rahmen eines ISO 27035 basierenden Incident-Management-Prozesses[22] erfolgen, welches hier aber aus Platzgründen nicht weiter betrachtet werden konnte.

In juristischer Hinsicht sind Hersteller von kommerzieller Software dazu angehalten, sich über die möglichen Gewährleistungs- und Haftungsansprüche ihrer Kunden klarzuwerden. Von besonderer Bedeutung ist hier der in den meisten FOSS-Lizenzen vereinbarte vollständige Gewährleistungs- und Haftungsausschluss. Während ein derart umfangreicher Ausschluss von Ansprüchen nach US-amerikanischen Recht zulässig sein dürfte, lässt sich eine solche Einschränkung nach dt. Recht nicht wirksam an den Kunden weiterreichen. In Individualverträgen lässt sich die Haftung bis zur Grenze des Vorsatzes bzw. der Arglist ausschließen. Bei Allgemeinen Geschäftsbedingungen hingegen kann eine Haftung für grobe Fahrlässigkeit nicht abbedungen werden. Ebenso kann die Haftung wegen (leichter) Fahrlässigkeit nicht ausgeschlossen werden, soweit es sich um die Verletzung von wesentlichen Vertragspflichten handelt. Möglich ist jedoch eine summenmäßige Begrenzung auf den vorhersehbaren, vertragstypischen Schaden.[23] Zudem müssen Hersteller Verkehrssicherungspflichten beachten. Auf der anderen Seite werden sich die Verwender kaum gegenüber den Urhebern von FOSS schadlos halten können. Wenn nicht bereits der Haftungsausschluss greift, dürfte es in meisten Fällen auch an deren Liquidität fehlen.

Herstellern von Software empfiehlt es sich daher aus juristischer Sicht, entsprechende technische Vorkehrungen zu treffen, wie etwa das angesprochen IT-Security-Monitoring, und Pläne für einen Incident-Management-Prozess bereitzuhalten. Nur auf diese Weise kann im Ernstfall nachgewiesen werden, dass man nicht grob fahrlässig gehandelt und auch ansonsten alles Zumutbare unternommen hat, (weitere) Schäden bei seinen Kunden soweit möglich zu verhindern.

Literatur

[Bart89] Harald Bartl: Produkthaftung nach neuem EG-Recht (ProdHaftG), Mi-Wirtschaftsbuch (1989).

[Baue89] Axel Bauer: Produkthaftung für Software nach geltendem und künftigem deutschen Recht, In: PHI 1989, VVW-Verlag (1989) 33-48.

[Deik03] Thies Deike: Open Source Software: IPR-Fragen und Einordnung ins deutsche Rechtssystem, In: CR 2003, Otto-Schmidt-Verlag, 9-18.

[Hoer88] Thomas Hoeren: Softwareüberlassung als Sachkauf - Konsequenzen aus dem Urteil des BGH vom 4- November 1987, In: RDV 1988, Datakontext Verlag (1988) 115-120.

[Hoer89] Thomas Hoeren: Softwarehaftung innerhalb der Europäischen Gemeinschaft, In: Handbuch der modernen Datenverarbeitung, Heft 146/1989, Forkel-Verlag (1989) 22.

[Hons95] Heinrich Honsell: Produkthaftungsgesetz und allgemeine Deliktshaftung, In: JuS 1995, C.H.Beck (1995) 211-215.

[22] ISO/IEC 27035:2016 Information technology — Security techniques — Information security incident management: http://iso27001security.com/html/27035.html

[23] Siehe BGH, Beschluss vom 17.10.2013, Az. I ZR 226/12 sowie BGH, Urteil vom 19.02.1998, Az. I ZR 233/95.

[Horn17] Jakob Horn: Der kaufrechtliche Ausbesserungsanspruch – Nachbesserung um den Preis eines neuen Mangels, In: NJW 2017, C.H.Beck (2017) 289-294.

[JaMe16] Till Jaeger, Axel Metzger: Open Source Software, 4. Auflage, C.H.Beck (2016).

[Junk88] Abbo Junker: Ist Software Ware?, In: WM 1988, WM Gruppe (1988) 1217-1221 und 1249-1255.

[Junk03] Abbo Junker: Computerrecht, 3. Auflage, Nomos-Verlag (2003).

[Koch00] Frank Koch: Urheber- und kartellrechtliche Aspekte der Nutzung von Open-Source-Software (II), In: CR 2000, Otto-Schmidt-Verlag (2000) 333-344.

[Lehm92] Michael Lehmann: Produkt- und Produzentenhaftung für Software, In: NJW 1992, C.H.Beck (1992) 1721-1725.

[Litt05] Sigurd Littbarski: Das neue Geräte- und Produktsicherheitsgesetz: Grundzüge und Auswirkungen auf die Haftungslandschaft, In: VersR 2005, VVW-Verlag (2005) 448.

[Mart93] Michael Martinek: Moderne Vertragstypen – Computerverträge, Kreditkartenverträge sowie sonstige moderne Vertragstypen (Band 3), C.H.Beck (1993).

[MeJa99] Axel Metzger, Till Jaeger: Open Source Software und deutsches Urheberrecht, In: GRUR Int. 1999, C.H.Beck (1999) 839-848.

[MeWe98] Klaus Meier, Andreas Wehlau: Die zivilrechtliche Haftung für Datenlöschung, Datenverlust und Datenzerstörung, In: NJW 1998, C.H.Beck (1998) 1585-1591.

[Rede12] Helmut Redeker: IT-Recht, 5. Auflage, C.H.Beck (2012).

[Scha15] Haimo Schack: Urheber- und Urhebervertragsrecht, 7. Auflage, Mohr Siebeck (2015).

[Schi03] Thomas Schiffner: Open Source Software – Freie Software im deutschen Urheber- und Vertragsrecht, VVF (2003).

[Schn17] Jochen Schneider: Handbuch EDV-Recht, 5. Auflage, Otto-Schmidt-Verlag (2017).

[ScLo17] Gerhard Schricker, Ulrich Loewenheim: Urheberrecht, 5. Auflage, C.H.Beck (2017).

[Sest00] Peter Sester: Open-Source-Software: Vertragsrecht, Haftungsrisiken und IPR-Fragen, In: CR 2000, Otto-Schmidt-Verlag, 797-807.

[Spin04] Gerald Spindler: Rechtsfragen bei Open Source, Otto-Schmidt-Verlag (2004).

[Taeg96] Jürgen Taeger: Produkt- und Produzentenhaftung bei Schäden durch fehlerhafte Computerprogramme, In: CR 1996, Otto-Schmidt-Verlag (1996) 257-271.

[Tasc86] Hans Claudius Taschner: Produkthaftung, C.H.Beck (1986).

[Witz16] Michaela Witzel: Beschaffung von Open Source Software, In: ITRB 2016, Otto-Schmidt-Verlag (2016) 160-164.

[ZsLu05] Kerstin A. Zscherpe, Hoger Lutz: Geräte- und Produktsicherheitsgesetz: Anwendbarkeit auf Hard- und Software, In: K&R 2005, Deutscher Fachverlag (2005) 499-502.