Kommunikation & Recht

Betriebs-Berater für
● Medien ● Telekommunikation ● Multimedia

3
K&R

Editorial: Klarstellung zur Verantwortlichkeit im Internet: Websperren möglich · *Dr. Florian Drücke*

145 Das Anhängen an Angebote bei Amazon · *Dr. Bernd Lorenz*

150 Framing zulässig oder unzulässig? · *Christian Galetzka*

152 Softwaremängel 4.0 – Wieviel (Un-) Sicherheit können wir uns leisten? · *Dr. Florian Deusch* und *Prof. Dr. Tobias Eggendorfer*

158 Die Novellierung des Informationsweiterverwendungsgesetzes *Carl Christian Müller*

163 Anonymitätsschutz für „Pick-Up Artists"? · *Dr. Marc-Oliver Srocke*

166 Telekommunikationsrechtliche Meldepflicht eines E-Mail-Dienstes *Sebastian Telle*

168 Länderreport USA · *Clemens Kochinke*

171 BGH: Störerhaftung des Access-Providers mit Kommentar von *Franz Gernhardt*

179 BGH: Werbung in automatisch generierten Bestätigungs-E-Mails unzulässig

182 BGH: Anforderungen an Preisangabe für Telekommunikations-dienstleistung

185 BGH: Gutscheinaktion beim Buchankauf verstößt gegen Preisbindung

194 OLG Köln: Anschlussinhaber hat keinen Anspruch auf sofortige Löschung von IP-Adressen

197 OLG Frankfurt a. M.: Anforderungen an wirksame Werbe- und Cookie-Einwilligung mittels Gewinnspiel

213 LAG Mainz: Fristlose Kündigung nach Installation von Schadsoftware auf Arbeitgeber-PC mit Kommentar von *Marie Herberger*

19. Jahrgang März 2016 Seiten 145 – 216

dfv Mediengruppe · Frankfurt am Main

EuGH bei jeder Verlinkung von Material, das im Internet ohne Zustimmung des Rechteinhabers zur Verfügung stand, – also auch beim Framing – eine öffentliche Wiedergabe konsequenterweise bejahen. Denn der Rechteinhaber hatte mit Blick auf die öffentliche Wiedergabe an das Internet-Publikum der Ursprungsseite gerade nicht unlimitiert autorisiert, so dass es sich bei den Nutzern des Links um ein von einer Autorisierung des Rechteinhabers nicht abgedecktes Publikum handelt.[30]

Ob im vorliegenden Streitfall das Video tatsächlich ohne Zustimmung der Klägerin, wie diese behauptet, zugänglich gemacht worden war, muss nun das OLG München abschließend klären.

IV. Konsequenzen für die Praxis

In der Praxis verbleiben aktuell erhebliche Unsicherheiten beim Framing, da der Linksetzende zunächst prüfen muss, ob eine Zustimmung des Rechteinhabers zur Zugänglichmachung des urheberrechtlich geschützten Inhalts auf der Ursprungsseite vorliegt, bevor er den Inhalt auf seiner eigenen Internetseite als Frame einbinden kann. Sofern er diesbezüglich falsch liegt, droht mitunter eine täterschaftliche Haftung für die verlinkte öffentliche Zugänglichmachung.[31]

Eine Vorabprüfung, ob die ursprüngliche Zugänglichmachung rechtmäßig – also mit Zustimmung des Rechteinhabers – erfolgte, ist in der Praxis bezüglich der auf YouTube oder anderen Streaming-Portalen zugänglichen Inhalte schwer durchzuführen.[32] Auch eine Nachfrage beim jeweiligen Portalbetreiber dürfte wenig Erfolg versprechen, da sich dieser nicht ohne weiteres zu einer Aussage über die Rechteinhaberschaft hinreißen lassen wird (so er diese überhaupt kennt). Er müsste ansonsten auch um seine Haftungsprivilegierung als Hosting-Provider bei User Generated Content (vgl. § 10 TMG) fürchten. Viel schwieriger bis unmöglich ist die Prüfung für den Linksetzenden beim Teilen von Text-, Ton- oder Videoinhalten über soziale Netzwerke, die oftmals bereits von anderen Dritten stammen, die den Link der Ursprungsseite geteilt haben. Auch die Teilen-Funktion führt je nach Portalausgestaltung z. B. bei Facebook zu einer direkten Wiedergabe des auf der ursprünglichen Internetseite abrufbaren Inhalts, zu welchem der gesetzte Link führt.

Für die anwaltliche Beratungspraxis ist zu beachten, dass die Verletzung eines unbenanntes Rechts der öffentlichen Wiedergabe nach § 15 Abs. 2 UrhG beim Framing vorerst davon abhängen wird, ob der Fremdinhalt, auf den der Embedded Link verweist, ohne Zustimmung des Rechteinhabers zugänglich gemacht worden ist oder ob dieser seine Inhalte auf der ursprünglichen Internetseite willentlich nur einem eingeschränkten Personenkreis zugänglich machen wollte. Auch einschränkende Nutzungsbedingungen und technische Schutzmaßnahmen, wie z. B. Session-IDs,[33] könnten zukünftig im Rahmen der Vorabprüfung durch den Linksetzenden ebenfalls eine Rolle spielen.[34]

Weitere Rechtsklarheit bringt hoffentlich die noch ausstehende EuGH-Entscheidung in der Rechtssache *GS Media/Sanoma Media Netherlands u. a.*, die das OLG München nach Zurückverweisung durch den BGH wohl berücksichtigen kann. Die bisherigen höchstrichterlichen Entscheidungen zeigen jedoch, dass vollständige Rechtssicherheit nur der Gesetzgeber – auch auf internationaler Ebene – schaffen kann, um Lösungen für Internetsachverhalte zu finden, die gerichtlich immer nur im Einzelfall betrachtet werden können.[35]

30 *Leistner*, GRUR 2014, 1145, 1154.
31 *Fuchs/Farkas*, ZUM 2015, 110, 117 f.; *Jani/Leenen*, GRUR 2014, 362, 363.
32 Ebenso *Spindler*, GRUR 2016, 157, 158.
33 Hierzu BGH, 29. 4. 2010 – I ZR 39/08, K&R 2010, 802 – Session-ID.
34 Vgl. EuGH, 13. 2. 2014 – C-466/12, K&R 2014, 256 ff., Rn. 31 – Svensson/Retriever Sverige.
35 *Spindler*, GRUR 2016, 157, 160.

RA Dr. Florian Deusch, Ravensburg und Prof. Dr. Tobias Eggendorfer, Weingarten*

Softwaremängel 4.0 – Wieviel (Un-) Sicherheit können wir uns leisten?

Die Zukunft sei das „Internet der Dinge" und „Industrie 4.0". Die vielfältige Vernetzung zwischen Vorgängen und Daten scheint den Weg in eine Welt des Unmöglichen, Unbegrenzten zu eröffnen, in die „Welt 4.0". Doch fragt sich, ob alle juristischen Lösungen, die zu den herkömmlichen Problemen bei „1.0-Anwendungen" entwickelt wurden, tauglich sind für die Softwarewelt 4.0, insbesondere zur Qualitätssicherung in der Softwareentwicklung und zum Umgang mit Softwarefehlern.

I. Einleitung: „Software ist nie fehlerfrei"

Aktuelle Software enthält durchschnittlich 3 bis 10 Fehler pro 1000 Zeilen Code.[1] Viele dieser Fehler bleiben bei Softwaretests unentdeckt und fallen im Nutzungsalltag auch nicht auf. Allerdings können diese Fehler zur Angreifbarkeit des IT-Systems des Users führen und Unbe-

* Der Autor *Deusch* ist als Rechtsanwalt, Fachanwalt für Informationstechnologierecht und externer zertifizierter Datenschutzbeauftragter in der Anwaltskanzlei Dr. Gretter tätig; der Autor *Eggendorfer* ist Professor für IT-Sicherheit an der Hochschule Weingarten, freiberuflicher IT-Berater und ebenso zertifizierter externer Datenschutzbeauftragter. Der Beitrag geht auf einen Vortrag bei der DSRI-Herbstakademie 2015 zurück, der im Tagungsband von *Taeger* (Hrsg.), Internet der Dinge – Digitalisierung von Wirtschaft und Gesellschaft, 2015, dokumentiert wurde. Er ist aktualisiert zum Stand: Januar 2016. Mehr über die Autoren erfahren Sie auf S. VIII.

1 http://www.informatik.uni-oldenburg.de/~iug10/sli/index01ce.html?q=node/25, (Stand: 11. 1. 2016); ebenso bereits *Taeger*, Außervertragliche Haftung für fehlerhafte Computerprogramme, 1995, S. 40. Daher wird die Behauptung, Software könne nicht fehlerfrei sein, bisweilen der Softwareindustrie zugeschrieben, siehe zum Beispiel *Marly*, Praxishandbuch Softwarerecht, 6. Aufl. 2014, S. 608 f.

fugten Nutzungsmöglichkeiten der Software eröffnen, die deren Hersteller nicht vorgesehen hat, z. B. die Übernahme des Systems, das Ausspähen oder die Manipulation von Daten. In der IT-Sicherheit spricht man in Abgrenzung zu Softwarefehlern, die nur störend sind, von sicherheitsrelevanten Fehlern.

Rechtsprechung und Rechtsliteratur haben sich mit dieser ungenügenden Qualität von Software scheinbar abgefunden, indem der Begriff des „Softwarefehlers" im Sinne der Informatik unterschieden wird vom „Softwaremangel" im rechtlichen Sinn: Erst wenn ein „Softwarefehler" (Informatik) die Funktionalität eines Computerprogramms – für den Anwender spürbar – beeinträchtigt, wird ein Softwaremangel im rechtlichen Sinn angenommen.[2] Der vorliegende Beitrag untersucht, ob diese Betrachtungsweise für „Softwareanwendungen 4.0" noch haltbar ist. Dargestellt werden zunächst Einsatzgebiete von „Software 4.0" sowie typische Softwarefehler und die damit verbundenen Risiken (unten Abschnitte II. und III.). Anhand dieser Konstellationen werden die Schwächen des bisherigen Sachmangelbegriffs im Softwarerecht und mögliche Lösungsansätze dargestellt (Abschnitt IV.). Ein Fazit und die zentralen Thesen schließen die Untersuchung ab (Abschnitt V.).

II. Einsatzgebiete von Software und Risiken bei „4.0-Anwendungen"

Immer mehr dringen Computer in den Lebensalltag vor: Von Motorsteuergeräten, Komfort- und Sicherheitsfunktionen im Fahrzeug über Smart-Metering im Stromnetz und Gebäudeautomation bis hin zu Herzschrittmachern, die über Funk konfigurierbar sind, finden sich in fast allen Alltagsgeräten heutzutage mehr oder minder komplexe Softwaresysteme. Immer mehr dieser Systeme sind an das Internet angebunden. Dabei können Sicherheitsmängel ausgenutzt werden, die durch Programmierfehler hervorgerufen und früher nicht entdeckt bzw. nicht behoben wurden. Beispiele sind die Angriffe auf die Heizungssteuerungen in der Vaillant EcoPower[3] oder das Knacken und Lokalisieren von Fahrzeugen bei Tesla,[4] bei BMW Connected Drive[5] und das „Kapern" eines Jeep Cherokee während der Fahrt[6] über das Internet. Einem amerikanischen Hacker gelang es sogar angeblich, sich über das In-Flight-Entertainment-System einer Linienmaschine in die Cockpit-IT zu hacken; jüngst sind auch IT-Sicherheitslücken bei medizinischen Implantaten untersucht worden.[7]

Ähnliche Risiken entstehen auch für Industriesteueranlagen – der Wurm StuxNet zeigte schon vor Jahren das theoretisch Mögliche, indem er zunächst durch die Manipulation der Drehzahl der Zentrifugen zur Aufbereitung spaltbaren Urans das iranische Atomprogramm sabotierte, und später sämtliche Industriesteueranlagen von Siemens anfiel.[8] Solche Gefahren sind für den regulären Anwender im Nutzungsalltag kaum zu erkennen, häufig sind die Fehler für ihn nicht wahrnehmbar, sie würden bei herkömmlichen Funktionalitätstests, so sie überhaupt durchgeführt wurden, nicht auffallen, sogar Schäden können lange Zeit unentdeckt bleiben. Erst gezielte Sicherheitsaudits können diese Lücken sichtbar machen. Das zeigt zum Beispiel der auf der BlackHat 2012 vorgeführte Hack von Zahlterminals über manipulierte Chips auf EC- und Kreditkarten,[9] durch die Angreifer die Software der Bezahlterminals komplett austauschen konnten, was eine Vielzahl von Angriffsszenarien eröffnet: Vom Ausspähen von Kartendaten bis hin zum Zahlungsbetrug.

Sämtliche dieser Angriffe waren durch fehlerhafte Programmierung der jeweiligen Systeme möglich: Bei Tesla zum Beispiel war eine unsichere Passwortverwaltung ursächlich, bei BMW eine falsche Annahme über die Sicherheit von Handynetzen, StuxNet nutzte systemimmanente Schwächen von USB-Anschlüssen und die Zahlterminals waren durch einen Buffer Overflow angreifbar.[10] Alles Fehler, die durch eine ordentliche Qualitätssicherung vermeidbar gewesen wären.

III. Softwarefehler – nicht nur – bei 4.0 Anwendungen

1. Sicherheitsrelevante Programmierfehler

Die meisten sicherheitsrelevanten Programmierfehler sind schon von „1.0-Anwendungen" bekannt und werden auch dort nur unzureichend verhindert, da sie für Nutzer ohne einen gezielten Penetrationstest kaum zu finden sind. Nur in den seltensten Fällen ist Sicherheit bei Individualsoftware im Lastenheft zu finden, bei marktgängigen Produkten finden Kunden es durchaus normal, dass Software fehlerhaft ist, zu Abstürzen neigt oder sie wöchentlich Updates[11] installieren müssen, um schwere Sicherheitslücken zu beheben.

Zu den gängigen Lücken zählen die fehlende oder fehlerhafte Authentifizierung der Kommunikationspartner, unzureichende Eingabedatenprüfung, die zu Buffer-Overflow-, Format-String-, Off-By-One-, Boundary-Check- und Out-of-Bounds-Read-Schwachstellen[12] führen können. In die gleiche Kategorie fallen Code-Injection-Angriffe.[13]

2 *Marly* (Fn. 1), S. 608 f.; siehe auch unten Abschnitt IV.

3 http://www.bhkw-infothek.de/nachrichten/18555/2013-04-15-kritische-sicherheitslucke-ermoglicht-fremdzugriff-auf-systemregler-des-vaillant-ecopower-1-0/ (Stand: 11. 1. 2016).

4 http://www.theregister.co.uk/2014/07/21/chinese_uni_students_pop_tesla_model_s/ und http://www.dhanjani.com/blog/2014/03/curosry-evaluation-of-the-tesla-model-s-we-cant-protect-our-cars-like-we-protect-our-workstations.html (je Stand: 11. 1. 2016) – sowie zahlreiche weitere Hacks auf Tesla.

5 http://www.heise.de/newsticker/meldung/ConnectedDrive-Der-BMW-Hack-im-Detail-2540786.html (Stand: 11. 1. 2016).

6 http://www.sueddeutsche.de/digital/jeep-aus-der-ferne-gehackt-die-angst-vor-dem-autohack-kommt-jahre-zu-spaet-1.2578024 (abgerufen 11. 1. 2016).

7 Zum Cockpit-Hack: http://uk.businessinsider.com/hacker-chris-roberts-allegedly-said-he-hacked-airplanes-entertainment-system-2015-5?r=US; zur Diskussion der IT-Sicherheitslücken bei medizinischen Implantaten auf dem 32. Kongress des Chaos Computer Clubs vom 27. bis 30. 12. 2015 siehe https://events.ccc.de/congress/2015/Fahrplan/events/7273.html (Stand: jeweils 11. 1. 2016).

8 https://de.wikipedia.org/wiki/Stuxnet (Stand: 11. 1. 2016).

9 https://www.blackhat.com/docs/us-14/materials/us-14-Anderson-How_Smartcard-Payment-Systems-Fail.pdf sowie sehenswert die Demonstration des Hacks: http://www.youtube.com/watch?v=18IAjDGOdKo (je Stand: 11. 1. 2016).

10 Bei einem Buffer Overflow (Pufferüberlauf) werden zu große Datenmengen in einem dafür zu klein reservierten Zielspeicher geschrieben; dadurch werden vorhandene Daten im Zielspeicher überschrieben, was zu Programmabstürzen oder Manipulationen führen kann, siehe https://de.wikipedia.org/wiki/Puffer%C3%BCberlauf, (Stand: 11. 1. 2016); ebenso *Greg Hoglund/Gary McGraw*, Exploiting Software. How to Break Code, Addison Wesley, 2004.

11 Wie z. B. beim Microsoft Patch-Day.

12 Ein Out-Of-Bound-Read war die Ursache für den Heartbleed-Angriff, mit dem die SSL-Verschlüsselung unter Umständen ausgehebelt werden kann. Er spielt auch eine Rolle im Stagefright-Angriff, mit dem Android-Smartphones z. B. mit einem speziell präparierten Video unter die Kontrolle eines Angreifers gebracht werden können. Beim Stagefright-Angriff sind einige Integer-Overflow-Fehler mitursächlich.

13 Weiterführend: *Greg Hoglund/Gary McGraw* (Fn. 10); *Cyrus Peikari/Anton Chuvakin*, Security Warrior. Know Your Enemy, O'Reilly, Sebastopol, 2004; *Jack Koziol et al*., The Shellcoder's Handbook, Discovering and Exploiting Security Holes, 2004.

Sämtlichen diesen Fehlern ist gemein, dass sie mit minimalen Aufwand bei der Programmierung hätten verhindert werden können und durch ein vernünftiges Qualitätsmanagement entdeckt worden wären. Ihre Ursachen sind banale Programmierfehler, zum Beispiel das Verzählen um ein Byte oder die fehlende Prüfung der Größe eines Zielspeicherbereichs.[14] Sie sind allerdings regelmäßig Ursache für spektakuläre Angriffe, von denen Abschnitt II. einige aufgelistet hat.

Ein leuchtendes Beispiel für entsprechende Gegenmaßnahmen ist das OpenSource-Betriebssystem „OpenBSD", das in den letzten rund 20 Jahren in der Standardinstallation nur zwei entfernt ausnutzbare Sicherheitslücken hatte.[15] Das liegt daran, dass es bei OpenBSD für Entwickler klare Richtlinien gibt, wie zu programmieren ist, ein zweiter Entwickler oder ein Team eingereichten Programmcode gegenliest, um Fehler und Schwachstellen zu finden, zusätzlich konsequent neu erkannte Fehler durch Gegenmaßnahmen für die Zukunft verhindert werden und sämtliche bekannten oder zu erwartenden Fehler durch automatisierte Tests überprüft werden.

OpenBSD bietet dabei in der Standardinstallation einen mindestens vergleichbaren Leistungsumfang wie andere Betriebssysteme. Im Gegensatz zu Microsoft jedoch wird Sicherheit proaktiv gewährleistet.

Das zeigen der Blaster und Sasser-Wurm auf eindrucksvolle Weise: Der erste nutzte einen Buffer-Overflow im RPC-Dienst von Windows aus, der zweite einen Buffer-Overflow im LSASS-Dienst. Das sind zwei von vier Diensten, die Windows XP standardmäßig über das Netzwerk zur Verfügung stellte. Ein Test auf einen Buffer Overflow lässt sich einfach automatisieren, zu einer ordentlichen Qualitätssicherung gehört auch, bei Bekanntwerden eines Problems an einer von vier ähnlichen Funktionen, die anderen mit zu prüfen.

2. Gegenmaßnahmen

All diese Sicherheitslücken lassen sich durch geeignete defensive Programmierung frühzeitig vermeiden. Zudem kann durch Coding-Standards eine entsprechende Qualität sichergestellt werden, die über automatisierte Tests sowie manuelle Code-Reviews geprüft werden kann.

3. Qualitätssicherung im Ingenieurwesen

Entsprechende Qualitätskonzepte scheinen im Bereich der Software leider noch wenig im Einsatz zu sein: Das zeigt zum Beispiel das erneute Auftreten des aus den 90er Jahren bekannten Ping-Of-Death,[16] der in einem Update von Windows im Jahr 2013 wieder auftrat.[17] Microsoft befand sich dabei in guter Gesellschaft: 2007 hat der Netz-Hardwarehersteller Cisco in seinem Betriebssystem Cisco iOS diese Uralt-Lücke wiederbelebt, 2008 folgten Nortel und LinkSys.

Der oben erwähnte Buffer-Overflow-Angriff ist seit 1972 theoretisch bekannt,[18] führte 1988 durch den ersten Internet Wurm zu einer Infektion von 10 % des Internets,[19] bekam 1996 durch einen in der Fachwelt viel beachteten Artikel[20] wieder Bekanntheit und ist heute noch ein Standardangriffsvektor. Die Lücke ist somit seit 43 Jahren bekannt, wie man sie verhindert auch.

Ähnliche Aufzählungen lassen sich für alle gängigen Sicherheitslücken vortragen. Und sie zeigen deutlich, dass die Qualitätssicherung im Bereich der Softwareentwicklung unzureichend ist. Keine andere Industrie könnte es sich leisten, seit 43 Jahren dieselben Mängel zu liefern. In keiner anderen Branche würde sich ein Unternehmen wie Microsoft, dessen Produkte regelmäßig durch Sicherheitslücken auffallen, als Marktführer behaupten können.

In anderen Branchen ist Qualitätssicherung gängige Praxis, so zum Beispiel im Ingenieurwesen. Dort sind entsprechende Maßnahmen auch vielfältig gefordert: Mängel im Qualitätsmanagement im Bauwesen, Maschinenbau, Automobilbau etc. führen in der Regel zur Haftung des Herstellers, die sich weit über die Behebung der unmittelbaren Mängel hinaus erstrecken kann.

4. Übertragbarkeit auf Software

Die Qualitätskonzepte des Ingenieurwesens sind nach Auffassung der Autoren vollständig auf die Herstellungsprozesse von Software übertragbar, allerdings entkommen Hersteller von Software diesen Anforderungen oft, weil Kunden kaum klare Qualitätsvorgaben außer den Nutzungsanforderungen machen. Insbesondere fehlen Sicherheitsforderungen und im Auftrag festgelegte Testverfahren für Sicherheitsmängel.

Dadurch, dass das Fehlen von Sicherheitsmängeln selten vertraglich vereinbart ist, Sicherheitstests nur in den wenigsten Fällen zur Abnahme gehören, und die Rechtsprechung sich auf im Alltag störende Mängel spezialisiert (siehe unten Abschnitt IV.), ist der Anreiz für Softwarehersteller, Qualitätskonzepte umzusetzen, gering.

Aus der Praxis zeigen sich immer wieder Fälle, in denen bei Penetrationtests aufgedeckte Mängel vom Hersteller zwar zur Kenntnis genommen werden, aber die vorgeschlagenen Abhilfen völlig untauglich sind: Meist fordert dieser dann die Installation zusätzlicher Software, die schädliche Daten herausfiltert, bevor sie sein fehlerhaftes Programm erreichen. Eine – drastische – Analogie wäre ein Automobilhersteller, der es seinen Kunden verbietet, Hänge mit mehr als 0,5 % Gefälle hinabzufahren, nachdem bekannt wird, dass seine Bremsen schon bei der geringen Belastung überhitzen könnten.

Im Softwarebereich findet zudem eine merkwürdige Verschiebung der Verantwortung für sicherheitsrelevante Programmierfehler statt: Während in der Automobilindustrie ein (z. B. durch einen Crashtest aufgedeckter) konstruktiver Mangel zweifelsfrei dem Hersteller des Fahrzeuges angelastet wird, ist bei Software der Überbringer der schlechten Nachricht ein Täter: Er muss – insbesondere in den USA – mit hohem Aufwand sicherstellen, dass er keine Straftat im Rahmen seines Sicherheitstests verübt hat.

Wie gut diese Umkehr der Verantwortung funktioniert, zeigte eine von Microsoft in Zusammenarbeit mit dem FBI überführte Hackergruppe: Dazu wurde ein frisch installierter Windowsrechner in ein beobachtetes Netz gestellt, der innerhalb von weniger als 20 Minuten mit den ersten

14 Die Konsequenz dieser fehlenden Prüfung lässt sich außerhalb der IT regelmäßig vor Mitnahme-Möbelhäusern beobachten, wenn der Kofferraum des Kleinwagens für die Schrankwand nicht ausreicht. Dort sorgt derselbe Denkfehler für Erheiterung bei Umstehenden.

15 http://www.openbsd.org/security.html (Stand: 11. 1. 2016).

16 http://insecure.org/sploits/ping-o-death.html (Stand: 11. 1. 2016).

17 http://www.heise.de/security/meldung/Microsofts-August-Patches-und-die-Rueckkehr-des-Ping-of-Death-1934998.html (Stand: 11. 1. 2016).

18 *James P. Anderson*, Computer Security Technology Planning Study Volume II, 1972, S. 37.

19 Rob Morris nutzte in seinem Wurm einen Buffer Overflow im „finderd" aus. Der Wurm war ein Experiment, dessen desaströses Ergebnis auch seinen Autor überraschte.

20 *Aleph One*, Smashing the Stack for Fun and Profit, Phrack Magazine, Volume 7, Issue 49, 11. 8. 1996.

Schadprogrammen infiziert war. Statt die Angriffsvektoren zu nutzen, um die Fehler aus dem eigenen System zu beseitigen, wurden die Angreifer vom FBI verhaftet.[21]

Zum Vergleich: Einige Automobilbauer leiden darunter, dass ihre Fahrzeuge bevorzugt gestohlen werden und müssen sich deshalb bessere Schutzkonzepte überlegen. Wegfahrsperren gehören mittlerweile zum Standard. Hier sieht der Markt den Hersteller zumindest als einen an dem Problem Beteiligten und erwartet von ihm Lösungen.

IV. Vom Softwarefehler zum Sachmangel – Herausforderungen von 4.0 Anwendungen für den rechtlichen Sachmangelbegriff

Die bisherigen Beurteilungen zum Sachmangelbegriff im Softwarerecht (Ziffer 1) zeigen Defizite, die für Softwareanwendungen 4.0 unzumutbare Risiken nach sich ziehen (Ziffer 2). Softwareherstellern und -anwendern werden daher entsprechende Vorsorgemaßnahmen empfohlen (Ziffer 3).

1. Bisherige Beurteilungen zum Sachmangelbegriff bei Software

Eine Betrachtung von Sachmängeln bei Software hat zu unterscheiden zwischen dem Tatbestand des Mangels im Sinne der §§ 434 und 633 BGB und den Rechtsfolgen hieraus.

Gemäß den §§ 434 Abs. 2 und 633 Abs. 2 BGB liegt ein Sachmangel vor, wenn die Software nicht die vereinbarte Beschaffenheit aufweist, sich nicht für die vertraglich vorausgesetzte Verwendung eignet oder sich nicht für die gewöhnliche Verwendung eignet und nicht die Beschaffenheit aufweist, die bei Sachen der gleichen Art üblich sind und die der Käufer erwarten kann.

In der Informatik dagegen bedeutet „Fehler", dass das Programm nicht wie vorgesehen abläuft oder fehlerhafte Ausgabedaten bei einem fehlerhaft vorgesehenen Programmablauf entstehen. Als Sachmangel werden „Fehler" im Sinne der Informatik derzeit nur angesehen, wenn sie den Gebrauch der Software negativ beeinträchtigen. Begründet wird dies mit der sogenannten „Compiler-Entscheidung" des BGH,[22] wonach dieser die Unvermeidbarkeit von Softwarefehlern („Software ist nie fehlerfrei") anerkannt habe.[23] Ein Blick in die jüngere Rechtsprechung scheint diesen Befund zu bestätigen. Soweit ersichtlich, sind dort nur Softwarefehler als Sachmängel dokumentiert, die für den Anwender wahrnehmbare Funktionseinschränkungen der Software begründen.[24]

Sofern ein Sachmangel vorliegt, hat der Käufer bzw. Besteller einer Software das Recht, die Beseitigung des Mangels oder die Lieferung einer mangelfreien Sache zu verlangen (§§ 439, 635 BGB). Meist erhält der Softwareanwender dabei Programmdateien, die er zur Fehlerbehebung auf seinem IT-System zu installieren hat.[25] Je nach Hersteller heißen diese Aktualisierungen Patch,[26] Update,[27] Bugfix oder Hotfix. Diese Begriffe bedeuten praktisch dasselbe: Eine solche Aktualisierung tauscht in der Regel die als fehlerhaft identifizierte Programmstelle durch ein bezüglich dieses Fehlers korrigiertes Programm aus. Dabei wird bei einem Patch eine Änderungsdatei geliefert, die nur zusammen mit den Original-Programmdaten kombiniert den Fehler behebt; ein Update liefert ein vollständiges, korrigiertes Programm. Die Bezeichnung „Patch" ist nicht zufällig gewählt, so ein „Flicken" stopft das Loch, ist aber keine neue Hose.

Dabei kann offenbleiben, ob die Bereitstellung dieser Programmdateien durch den Softwarehersteller als Mangelbeseitigung oder als Lieferung einer neuen, mangelfreien Sache einzuordnen ist.[28] In beiden Fällen entscheidet allein der Softwarehersteller darüber, wie er das technische Problem löst, das dem Mangel zugrunde liegt.[29] Sofern die Software die vertraglich geschuldeten Funktionen ausführt, reicht es aus, das festgestellte Loch zu stopfen. Bestimmte Gegenmaßnahmen, etwa solche wie oben in Abschnitt III. beschrieben, kann der Softwareanwender zumindest auf der Grundlage der bisherigen Rechtsprechung ohne besondere Vereinbarungen nicht verlangen.

2. Schwächen des bisherigen Mangelbegriffs im Softwarerecht insbesondere bei 4.0-Anwendungen

Nachfolgend sollen die Schwächen des bisherigen Mangelbegriffs in Bezug auf sicherheitskritische Softwarefehler dargestellt werden. Auch in diesen Fällen verlangt ein Sachmangel entweder das Abweichen von einer Beschaffenheitsvereinbarung oder eine Funktionsbeeinträchtigung.

a) Sicherheit als vereinbarte Beschaffenheit?

Technische Möglichkeiten und Wege zur Vermeidung sicherheitskritischer Programmierfehler sind oben in Abschnitt III. dargestellt worden. Ob und inwieweit in der Praxis der Softwareherstellung bestimmte Qualitätsstandards an sicherheitsorientierter Programmierung vereinbart werden, haben die Verfasser im Rahmen dieser Untersuchung nicht erhoben.[30] Fest steht jedenfalls, dass in einzelnen Bereichen Standards für derartige Qualitätslevels bestehen und auch vereinbart werden. Die Norm IEC 61508 etwa gibt einen Standard für den sicheren, zuverlässigen und störungsfreien Betrieb elektronischer programmierbarer Systeme vor. Die Vereinbarung solcher Standards setzt indes für den Softwareanwender voraus, dass er diese Normen kennt, in den Vertragsverhandlungen mit dem Softwarehersteller durchsetzt und deren Einhaltung in technischer Hinsicht prüfen kann. Ohne eine ausdrückliche Vereinbarung jedenfalls ist unklar, ob und falls ja welches Sicherheitsniveau der Softwarehersteller schuldet (siehe unten lit. b).

---

21 In der Selbstdarstellung von Microsoft: https://news.microsoft.com/2005/10/27/stopping-zombies-before-they-attack-microsoft-teams-with-federal-trade-commission-and-consumer-action-to-promote-pc-protection/ (Stand: 11. 1. 2016).

22 BGH, 4. 11. 1987 – VIII ZR 314/86, BGHZ 102, 135 = BB 1988, 20 ff. = NJW 1988, 406, 408 – Compiler.

23 *Marly*, (Fn. 1), S. 608 f.; *Hoeren*, in: von Westphalen (Hrsg.), Vertragsrecht und AGB-Klauselwerke, IT-Verträge, 36. Aufl. 2015, Rn. 39; *ders*., ZAP 2005, 551, 554; *Schneider*, Handbuch des EDV-Rechts, 4. Aufl. 2009, Kapitel D, Rn. 928; *Conrad/Schneider*, in: Auer-Reinsdorff/Conrad (Hrsg.), Handbuch IT- und Datenschutzrecht, 2. Aufl. 2016, § 10 Rn. 88.

24 siehe zum Beispiel BGH, 5. 6. 2014 – VII ZR 276/13, K&R 2014, 601, Mängel-Darlegungslast beim Softwarevertrag, BGH, 28. 5. 2014 – VIII ZR 94/13, BB 2014, 1999 – Akustische Einparkhilfe; OLG Hamburg, 16. 8. 2013 – 9 U 41/11, BeckRS 2013, 19747, OLG Düsseldorf, 14. 3. 2014, I-22 U 134/13.

25 *Schneider*, in: Lehmann/Meents, Handbuch des Fachanwalts IT-Recht 2011, Kapitel 4 Rn. 325.

26 https://de.wikipedia.org/wiki/Patch_(Software) (Stand: 11. 1. 2016).

27 https://de.wikipedia.org/wiki/Update (Stand: 11. 1. 2016).

28 *Hoeren/Spittka*, MMR 2009, 583, 588 verstehen das Updating als eine kombinierte Nacherfüllung aus Nachlieferung und Nachbesserung.

29 Bei der Lieferung einer neuen, mangelfreien Sache entscheidet der Unternehmer ohnehin über die Herstellung. Wenn sich bei der Nachbesserung verschiedene Möglichkeiten bieten, hat der Unternehmer das Wahlrecht, siehe *Faust*, in: Bamberger/Roth (Hrsg.), BeckOK BGB, 36. Edition August 2015, § 439 Rn. 26; im Werkvertragsrecht hat der Unternehmer gemäß § 635 BGB die Wahl zwischen Mangelbeseitigung und Lieferung einer neuen, mangelfreien Sache.

30 Es gibt dagegen Anzeichen für das absichtliche „Einbauen" von Sicherheitslücken durch die Softwarehersteller, siehe *Neumann*, Vorgänge 2014, S. 82 f.

b) Sicherheit als Voraussetzung für die „vertraglich vorausgesetzte oder gewöhnliche Verwendung" von Software?

Softwarefehler – Bugs – werden nur als Mangel gewertet, wenn sie die Verwendung der Software beeinträchtigen (siehe oben Ziffer 1). Softwarefehler, die sicherheitsrelevant sind, passen nicht unter diese Definition. Sicherheit ist die Abwesenheit von Risiken.[31] Sicherheitsrelevante Bugs führen deshalb nicht zwangsläufig zu Funktionseinschränkungen, sondern in erster Linie zu Risiken, die sich verwirklichen können oder eben nicht. Die IT-Sicherheit unterscheidet dabei zwischen akzeptablen und untragbaren Risiken. Deren Einordnung richtet sich an dem Sicherheitsniveau aus, das für den Einzelfall jeweils anzuwenden ist.[32] Wer aber definiert dieses Sicherheitsniveau?

Ein Sicherheitsmangel wird von einigen Autoren angenommen, wenn der Stand der Technik zum Zeitpunkt des Gefahrübergangs der verkauften Software unterschritten wird. Wenn dem Softwarehersteller solche Sicherheitslücken bekannt werden, soll er verpflichtet sein, den Softwareanwender zu informieren und ihm entsprechende Sicherheitspatches zur Verfügung zu stellen.[33] Offen ist dabei bislang geblieben, wie genau der „Stand der Technik" bei sicherheitskritischen Fehlern bestimmt wird und welche Risiken noch zumutbar bzw. nicht mehr tragbar sind.[34]

Möglicherweise ist dieses Vakuum konkreter Sicherheitsvorgaben dafür verantwortlich, dass sich bisweilen fragwürdige Vorgehensweisen und Beurteilungen bei – längst bekannten (siehe oben Abschnitt III.) – sicherheitskritischen Bugs zeigen:

Als „Bananensoftware" werden Produkte bezeichnet, die – wie eine Banane – nach dem Verkauf beim Kunden reifen sollen. Design-Fehler werden teilweise bewusst in Kauf genommen und erst nachträglich durch Updates beim Kunden beseitigt. Dieses Vorgehen löst zwar Sachmangelansprüche beim Kunden aus, aufgrund der faktischen Reduzierung der Anwenderrechte auf Patches und Updates (siehe oben Ziffer 1) ist die Motivation zur Vermeidung solcher Fehler indes begrenzt.[35]

Veraltete Software, die nicht mehr dem aktuellen Standard entspricht, sei nicht mangelhaft, sofern dem Anwender „in geeigneten Zeitabständen" Updates zur Verfügung gestellt werden. Konzeptionelle Schwachstellen („Entwurfs- bzw. Designfehler") führen nur zu Mangelansprüchen, wenn sie die Funktion der Software beeinträchtigen.[36]

Sofern überhaupt gerichtliche Entscheidungen zu Softwaremängeln existieren, stellen diese auf Funktionsbeeinträchtigungen ab.[37] Soweit ersichtlich, sind die Sicherheitsbedrohungen durch Softwarefehler bislang nicht Gegenstand einer gerichtlichen Entscheidung gewesen. Obgleich unzutreffende Fehlermeldungen auf Malware hindeuten können,[38] hat das OLG Koblenz einen Sachmangel nicht aufgrund eines Sicherheitsrisikos angenommen, sondern wegen des Aufwands, der mit der sinnlosen Nachprüfung der Fehlermeldungen verbunden war; das LG Dortmund hat einen Sachmangel trotz ungeklärter Programmabstürze abgelehnt, weil der betroffene Nutzer nicht dargelegt hat, „welche Anzahl an Programmabstürzen dem (…) heute üblichen Standard entspricht".[39]

Der Vermieter einer Telekommunikationsanlage, der sich verpflichtet hat, diese betriebsfähig zu halten, muss die Anlage nach Auffassung des OLG Köln nicht vor Hackern bewahren oder gar kostenlos (Sicherheits-) Updates bereitstellen. Der Zugriffsschutz durch einen 4-stelligen Zahlencode gewähre ein ausreichendes Maß an Sicherheit. 4-stellige Zahlencodes seien nicht per se unsicher, da auch Geldautomaten hierdurch geschützt würden.[40]

c) Relevanz sicherheitskritischer Softwarefehler für den Sachmangelbegriff

Insbesondere im Zeitalter der Industrie 4.0 stellen sich die Fragen, ob

– die Konzentration auf spürbare Funktionsbeeinträchtigungen zu einer gefährlichen Verengung des Sachmangelbegriffs im Softwarerecht führt und
– erheblich höhere Anforderungen an das Sicherheitsniveau für sachmangelfreie Software zu stellen sind und damit die bislang rechtlich oder faktisch akzeptierten Softwarefehler (siehe oben Ziffer 3) vermieden werden müssen, obgleich sie nicht zwingend zu Funktionseinschränkungen führen.

Denn gerade für Softwareanwendungen 4.0 hat die Sicherheit hohe Relevanz:

Marginale Fehler in der Software können sich im Rahmen eines Produktionsprozesses auf das gefertigte Produkt übertragen und Produktmängel hervorrufen. Angesichts der immensen Erwartungen an die Industrie 4.0 wird die Relevanz der IT-Sicherheit unterschätzt, obgleich Sabotagefestigkeit und Widerstandsfestigkeit für die Verfügbarkeit und Unversehrtheit der IT-Systeme 4.0 von zentraler Bedeutung sind.[41] Dies könnte ein anderes Licht auf die Frage werfen, welches Maß an Sicherheit und Qualität für Software in Zukunft erforderlich ist. In diesem Zusammenhang relevant sind die Ausführungen in der „Compiler-Entscheidung" des BGH zur angeblichen Unvermeidbarkeit von Softwarefehlern:

„Dabei kann offenbleiben, ob der Vortrag (…) zutrifft, (…) Software könne nie ganz mangelfrei sein, oder ob diese Behauptung nur dazu dient, den Herstellungsaufwand in Grenzen zu halten und das Erwartungsniveau von Softwareanwendern zu dämpfen."[42]

31 *Liggesmeyer*, Softwarequalität, 2. Aufl. 2009, S. 440.
32 https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/glossar/04.html, Stichwort Risiko (Stand: 11. 1. 2016); siehe auch oben Ziffer 1 (störende bzw. sicherheitsrelevante Softwarefehler).
33 *Mankowksi*, in: Ernst (Hrsg.): Hacker, Cracker & Computerviren, 2004, Rn. 442, 462; *Spindler*, NJW 2004, 3145.
34 Aus Sicht des Produkthaftungsrechts siehe dazu *Taeger* (Fn. 1), S. 179; zumal es Gerichtsentscheidungen gibt, wonach es für Software „keinen standardisierten oder gewöhnlichen Gebrauch" gebe (OLG Stuttgart, 12. 9. 1985 – 7 U 240/84, CR 1986, 381 f.).
35 *Hoeren* (Fn. 23), Rn. 45; *Kleuker*, Qualitätssicherung durch Softwaretests, 2013, S. 1.
36 OLG Oldenburg, 24. 4. 1991 – 12 O 204/90, NJW 1992, 1771; LG Freiburg, 14. 7. 2007 – 3 S 324/06, BeckRS 2007, 14105; OLG Brandenburg, 1. 12. 1998 – 6 U 301/97, K&R 1999, 369; *Schneider* (Fn. 23), Kapitel D Rn. 840 ff.; 928.
37 Nachweise siehe oben Abschnitt IV. Ziffer 1.
38 http://www.com-magazin.de/news/sicherheit/malware-tarnt-sicherheitssoftware-191674.html (Stand: 11. 1. 2016).
39 OLG Koblenz,19. 9. 2007 – 1 U 1614/05, BeckRS 2008, 01271; LG Dortmund, 19. 3. 2014 – 5 O 235/13, BeckRS 2014, 16352.
40 OLG Köln, 16. 7. 2013 – 19 U 50/13, MMR 2014, 101. Dass dabei Äpfel mit Birnen verglichen werden, ist ersichtlich: Während bei einem Geldautomaten nach drei Fehlversuchen die Karte eingezogen werden kann, kann ein Angreifer auf eine Telefonanlage seine Angriffe beliebig oft wiederholen. Einen Roboter für ein Durchprobieren aller PINs vor einem Geldautomaten aufzubauen, verbietet sich schon durch die Auffälligkeit der Maßnahme. Im Netz lassen sich aber solche Angriffe bequem durch kleine Programme unauffällig automatisieren, wie beispielsweise http://www.linux-magazin.de/Ausgaben/2008/12/Schoene-Bescherung und http://www.sueddeutsche.de/digital/datenleck-bei-der-dhl-sendungsbewusst-1.528925 (je Stand: 11. 1. 2016) zeigt.
41 *Bräutigam/Klindt*, NJW 2015, 1137, 1140, 1141.
42 BGH, 4. 11. 1987 – VIII ZR 314/86, BGHZ 102, 135 = BB 1988, 20 ff. = NJW 1988, 406, 408 – Compiler.

Damit sind die widerstreitenden Interessen eindeutig benannt: Es geht um das „Erwartungsniveau", welches auch die Qualität und Sicherheit von Software erfasst, und die damit verbundenen Herstellungskosten. Die Datenflut, die durch die Vernetzung in der Softwarewelt 4.0 entsteht, sowie deren steigende Sensibilität und Aussagekraft verleihen der IT-Sicherheit und damit den Risiken aus Softwarefehlern eine neue Dimension.[43] Ein höheres „Erwartungsniveau" an die Sicherheit von Software ist nach Auffassung der Verfasser deshalb notwendig.

Für 4.0-Anwendungen wie z. B. intelligente Softwareagenten wird deren Architekturen maßgebliche Bedeutung zugeschrieben, insbesondere ob diese sorgfältig nach dem Stand von Wissenschaft und Technik entwickelt wurden. Diese Anforderung steht den Urteilen zur Mangelfreiheit von Software mit veralteten Architekturen (siehe oben lit. b), zweiter Spiegelstrich) diametral entgegen. Gleichwohl ist bislang unklar, welche Sicherheitsvorkehrungen der Stand der Technik verlangt.[44]

In anderen Rechtsgebieten, wie etwa dem Baurecht, ist anerkannt, dass Sicherheitsrisiken Sachmängel darstellen. Ein sicherheitsrelevanter Verstoß gegen die anerkannten Regeln der Technik (etwa DIN-Normen) stellt einen Sachmangel dar; nur wenn nachweislich kein Risiko vorliegt, sind Mangelrechte des Bestellers ausgeschlossen.[45]

Zuzugeben ist zwar, dass nicht für alle Softwareanwendungen pauschal identische Sicherheitskriterien zugrunde gelegt werden können. Gleichwohl ist zu berücksichtigen, dass die zunehmende Vernetzung im Zeitalter 4.0 Anwendungen für vermeintlich risikolose Bereiche stets das Potential verleiht, weitere Risikodimensionen zu eröffnen, die bei der ursprünglichen Entwicklung der Software nicht vorgesehen und somit auch nicht bedacht wurden.[46] Aus diesem Grund sollten zumindest sicherheitsrelevante Fehler unterlassen werden, deren Gefahren bereits längst bekannt sind (dazu siehe oben Abschnitt III.).

3. Lösungsansätze für Softwareanwender und -hersteller

Der festgestellte Befund, dass die IT-Sicherheit im Rahmen des Sachmangelbegriffs nicht ausreichend berücksichtigt wird, bringt Risiken für Softwareanwender und -hersteller mit sich. Da unklar ist, welche Vorkehrungen der Stand der Technik und mithin die Softwarequalität insbesondere für 4.0-Anwendungen verlangt, wissen Softwareanwender nicht, auf welches Sicherheitsniveau sie vertrauen dürfen; Softwarehersteller dagegen tragen Risiken, falls ein Gericht in einem Haftungsfall höhere Anforderungen stellen sollte, als die Software vorweisen kann.

Es ist daher gleichermaßen im Interesse von Softwareanwendern und -herstellern, die Standards zur Softwarequalität durch Beschaffenheitsvereinbarungen im Sinne des § 434 Abs. 1 S. 1 BGB möglichst eindeutig und präzise zu definieren.

Beschaffenheitsvereinbarungen zur Definition konkreter Qualitätsmaßstäbe sind oftmals nur mithilfe von Standards möglich. Gerade aufgrund der konkret anstehenden Herausforderungen der „Industrie 4.0" müssen daher entsprechende Standards herausgearbeitet bzw. angewendet werden.[47]

Wenn Beschaffenheitsvereinbarungen nicht möglich sind, etwa im B2C-Bereich aufgrund der wirtschaftlichen Überlegenheit des Anbieters, ist Verbrauchern zu raten, sich zumindest hinreichend über die Eigenschaften der Software zu informieren.

Auf Seiten der Softwarehersteller wäre es denkbar, durch Selbstverpflichtungen bestimmte Qualitätsniveaus vorzugeben,[48] damit können Hersteller selbst einen angemessenen und für die Softwareanwender transparenten Qualitäts- und Haftungsmaßstab definieren.

Dass die nötige Qualität erreicht werden kann, zeigen einzelne Projekte, wie z. B. OpenBSD, deutlich. Allerdings wäre auch hier zu erwägen, Kontrollinstanzen einzuführen, etwa analog dem TÜV und ADAC im Automobilbereich. Anders ist bei der Komplexität eine seriöse Prüfung nicht möglich.

Es zeigt außerdem, dass der Aufwand für eine ordentliche Qualitätssicherung überschaubar ist, denn bei OpenBSD handelt es sich um ein OpenSource-Projekt, die Software ist kostenlos verfügbar.

V. Fazit und Thesen

1. Der bisherige Sachmangelbegriff im Softwarerecht ist gekennzeichnet durch eine Konzentration auf die Funktionalität des Softwareprodukts. Softwarefehler, die die Funktionen nicht – wahrnehmbar – beeinträchtigen, gelten noch immer nicht als Mangel; die Sicherheitsrelevanz bleibt unbeachtet.

2. Diese bisherige Betrachtung führt zu einer gefährlichen Verengung des Sachmangelbegriffs, da sicherheitskritische Softwarefehler nicht ausreichend berücksichtigt werden.

3. Softwareanwendungen von morgen fordern höhere Sicherheitsniveaus. Dennoch ist bislang unklar, nach welchen Kriterien sich der Stand der Technik zur ausreichenden Vermeidung sicherheitskritischer Softwarefehler zu richten hat. Klar ist jedenfalls, dass seit über 40 Jahren bekannte Mangelursachen (zum „Buffer-Overflow" siehe oben Abschnitt III.) nicht mehr dem Stand der Technik entsprechen können.

Angesichts der unsicheren Rechtslage ist Softwareanwendern und -herstellern zu empfehlen, das Sicherheitsniveau für „ihre" Software vertraglich zu definieren; hierzu ist die Ausarbeitung bzw. Anwendung einschlägiger Standards zu fordern.

---

43 *Bräutigam/Klindt*, NJW 1137, 1140.

44 *Müller-Hengstenberg/Kirn*, MMR 2014, 307, 310, 313.

45 *Genius* in: Herberger/Martinek/Rüßmann u. a., jurisPK-BGB, 7. Aufl. 2014, § 633 BGB Rn. 30; ebenso *Voit*, in: Bamberger/Roth, BeckOK, 37. Edition, Februar 2015, § 633 Rn. 11.

46 *Bräutigam/Klindt*, NJW 2015, 1137, 1140.

47 Beispielhaft die Zusammenstellung einschlägiger Normen und Standards im Bereich Industrie 4.0 der DKE (Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE), https://www.dke.de/de/std/Industrie40/Seiten/NormeninIndustrie40.aspx, abgerufen am 11. 1. 2016.

48 Beispiel aus der Baubranche: die Gütegemeinschaft Deutscher Fertigbau e. V., http://www.guete-gemeinschaft.de/ (Stand: 11. 1. 2016). Gleichwohl hat dieser Ansatz auch Schwächen: Die Bewertung dieser Informationen ist für den Softwareanwender als Laien faktisch nicht möglich. In Hochglanzprospekten der Hersteller wird stets von maximalen Sicherheitsanforderungen gesprochen. Diese zu bewerten, deren konkrete Auswirkungen zu erkennen, dazu ist ein Laie nicht in der Lage. Ist nun das Windows „Sicherheitscenter" mit Überwachung einer Firewall und Virenscanner ein geeigneter Schutz? Das kann durchaus bestritten werden: Denn dabei handelt es sich nur um eine aufgeflanschte, nachgerüstete Schutzausrüstung, nicht um eine systemimmanente. Eine bildliche Analogie ist ein Auto, um das noch Kissen geschnallt werden, damit es bessere Crashtestwerte erreicht. Solange hier keine Transparenz geschaffen ist, kann der Softwareanwender nur durch einen neutralen Sachverständigen über „Risiken und Nebenwirkungen" angemessen aufgeklärt werden.