Philosophisches Institut, RWTH Aachen

Hausarbeit im Rahmen des Seminars
Technikethik unter besonderer Berücksichtigung der Roboterethik
im WS 2017/18

Dozentin: Carmen Kraemer

Ethische Aspekte von Sicherheitslücken in Software

Michael Krause (331069)
Studiengang: Informatik M.Sc.
michael.krause@rwth-aachen.de

Aachen, 02.03.2018

Inhaltsverzeichnis

1 Einleitung

Mit der Veröffentlichung des sogenannten „KRACK“ Angriffs im Herbst 2017 wurde bekannt, dass die Kommunikation über WPA2-verschlüsselte drahtlose Computernetzwerke[1] ohne Kenntnis der Passphrase des jeweiligen Netzwerks mitgelesen werden kann (siehe Vanhoef und Piessens 2017). Der Angriff funktioniert bei Geräten verschiedener Typen und Betriebssysteme[2], von Smartphones bis Laptops. Um den Angriff zu verhindern, müssen betroffene Anwender eine Softwareaktualisierung[3] einspielen. Solche standen jedoch bei Publikation des Angriffs nicht für alle Geräte bereit. Auch Monate später sind noch Computer anfällig, für die keine Sicherheitsaktualisierungen durch ihre Hersteller veröffentlicht wurden.

Derartige Angriffe werden durch Sicherheitslücken in der Software[4] ermöglicht, die auf einem Gerät ausgeführt wird. Eine Sicherheitslücke in einem Computerprogramm ist ein Programmierfehler mit Konsequenzen für die Sicherheitsaspekte[5] des Programms (Takanen et al. 2004, S. 93)[6]. Durch solche Lücken kann die mit dem Programm durchgeführte Kommunikation kompromittiert, sowie Daten ausgelesen, verändert oder vernichtet, aber auch das Gerät selbst außer Betrieb gesetzt werden. Nicht jede Sicherheitslücke muss alle diese Szenarien zur Folge haben, aber jedes Szenario ergibt ein ethisches Problem: Werden die Lücken nicht durch Aktualisierung der Programme behoben, kann sogar kritische Infrastruktur beeinträchtigt werden[7]. Sicherheitslücken stellen damit auch eine Angriffsmöglichkeit in Cyberkriegen dar (Dunn Cavelty 2014, S. 2). Weiterhin kann durch sie die Privatsphäre von Nutzern bedroht sein, die mit ihrem Computer sensible Daten verarbeiten. Insbesondere im Zusammenhang mit Robotern, die in private Lebensbereiche eindringen, sind diese Szenarien bedenkenswert. Schließlich tragen fehlende Aktualisierungen bei Sicherheitslücken auch dazu bei, dass erworbene Produkte früher ersetzt werden müssen, mit negativen Folgen für Nachhaltigkeit und Umwelt (siehe dazu Deutsche Umwelthilfe 2018, S. 16).

Auch andere technische Artefakte lassen sich manipulieren. Angriffe auf Software können aber besonders schwerwiegend sein, weil sie über das Internet von fast jedem Ort der Welt und mit geringer Gefahr für die Angreifer erfolgen können. Darüber hinaus ist der Einsatz von Computerprogrammen, sei es auf einem eigenen Gerät oder auf der Arbeit, in der Praxis kaum zu vermeiden. Außerdem ist es durch die Komplexität von Software besonders schwer, das

potentielle Ausmaß eines Angriffs (etwa im Hinblick auf den Umfang persönlicher Daten, die kompromittiert werden könnten) zu überblicken.

Sicherheitslücken in Software sind somit nicht nur ein technisches Problem in der Informatik, sondern bedürfen auch einer Betrachtung durch die philosophische Ethik. In dieser Arbeit sollen einige ethische Fragen untersucht werden, die sich daraus ergeben: Ist es beispielsweise Pflicht, Sicherheitslücken an Softwarehersteller zu melden? Und wenn ja, wie? Sind zu deren Auffinden auch Angriffe auf Computersysteme erlaubt? Müssen Softwarehersteller daraufhin Sicherheitsaktualisierungen für ihre Programme bereitstellen? Und sind sie generell für alle Fehler ihrer Produkte haftbar?

In der praktischen Philosophie ist die Pflichtethik Immanuel Kants besonders einflussreich (Pleger 2017, S. 94 ff.). Kant möchte Handlungen nicht auf Grund voriger Erfahrungen bewerten, sondern auf Grund eines universellen moralischen Gesetzes, des kategorischen Imperativs (ursprünglich eingeführt in: Kant 1911, S. 420-421). Eine Handlung geschieht demnach aus moralischer Pflicht, wenn die zugrundeliegende Maxime zu einem allgemein bindenden Gesetz werden kann, also wenn sie universalisierbar[8] ist.

Aus diesem kategorischen Imperativ leitet Kant beispielsweise ein absolutes Verbot unehrlicher Versprechen ab (ebd., S. 402-404): Eine Maxime wie „Ich darf ein falsches Versprechen geben“ kann nicht zu einem universellen Gesetz werden, denn sie führt zu einem logischen Widerspruch. Gälte sie für alle, so könnte niemand mehr ein gegebenes Versprechen ernst nehmen. Dann würde aber auch jedes unehrliche Versprechen erfolglos bleiben.

Kants Ethik ist nicht unumstritten (für die Positionen anderer Philosophen siehe Pleger 2017). Insbesondere stößt seine universelle Ethik auf Schwierigkeiten in einer Welt, in der nicht alle Personen gemäß ihr handeln. Viele alternative Konzeptionen sind nicht wie Kants Ethik deontologisch und bewerten die ethische Natur einer Handlung an sich, sondern konsequentialistisch und bewerten somit eine Handlung gemäß ihrer Auswirkungen. Utilitaristen betrachten dabei insbesondere den Nutzen und Schaden, den eine Handlung erzeugt. Im Rahmen dieser Arbeit wird der kategorische Imperativ akzeptiert und Situationen manchmal zusätzlich aus einer konsequentialistischen Perspektive betrachtet.

Ziel der Arbeit ist, zu den vorgestellten Fragen jeweils Positionen aus der Literatur kritisch zu diskutieren und weiterhin zu ergründen, welches Handeln in jeder der betrachteten Situationen einer universellen moralischen Pflicht gemäß des kategorischen Imperativs entspräche.

2 Müssen Sicherheitslücken an die Hersteller von Software gemeldet werden?

Wie in der Einleitung vorgestellt handelt es sich bei Sicherheitslücken letztlich um Programmierfehler. Als solche fallen diese oft zunächst dem Hersteller einer Software auf. Ebenso können jedoch auch Experten, die sich eingehend mit einem Programm beschäftigen, auf eine Sicherheitslücke aufmerksam werden. Dieser Abschnitt behandelt die Frage, ob es in einem solchen Fall eine moralische Pflicht gibt, diesen Fund an den Softwarehersteller zu melden.

Zunächst einmal scheint das wenig kontrovers zu sein: Höchstens Kriminelle könnten doch ein Interesse daran haben, Sicherheitslücken für sich zu behalten. Es sind jedoch auch andere Fälle bekannt, in denen Softwarehersteller nicht über Probleme mit ihren Produkten informiert wurden. Das sogenannte „WannaCry“ Erpressungsprogramm[9] etwa entstand, nachdem eine Reihe von bis dato unbekannten Sicherheitslücken in die Hände von Kriminellen gelangte. Ursprünglich wurden diese Lücken durch den amerikanischen Geheimdienst NSA gefunden und gesammelt, ohne dass dieser seine Erkenntnisse weitergab (heise online 2017). Als die Sammlung an Dritte geriet, nutzten diese sie für weltweite Angriffe.

Es gibt somit Akteure wie den NSA, die ihre Funde nicht melden. Sie verstoßen damit gegen ihre moralische Pflicht laut des kategorischen Imperativs, denn eine Handlungsmaxime, gemäß der jeder gefundene Sicherheitslücken für sich behält, ist nicht universalisierbar: In einer Welt, in der jeder nach ihr handelt, ist die Sicherheit von Software generell in Frage gestellt. Das Wissen über die Angreifbarkeit eines Programms wäre zwar nicht öffentlich, aber alle wüssten, dass andere ihr Wissen darüber nicht teilen. Somit muss jeder Computernutzer prinzipiell annehmen, dass alle verwendeten Systeme unsicher sind, und wird diese nicht mehr für wichtige Kommunikation etc. nutzen. Dann wird aber das ursprünglich geheimgehaltene Wissen über die Sicherheitslücken wertlos und die Handlungsmaxime letztlich widersprüchlich.

Das Verschweigen von Sicherheitslücken ist auch deshalb nicht wünschenswert, da damit die in der Einleitung genannten Probleme durch den Hersteller nicht angegangen werden könnten. Demgegenüber ist das Gebot, Sicherheitslücken an den Hersteller weiterzugeben, durchaus universalisierbar: Unter dieser Maxime

handeln alle im Interesse besserer Software. Jede Meldung an den Hersteller kann einen Beitrag zu sichererer Software leisten.

Tatsächlich haben diese Überlegungen gemäß des kategorischen Imperativs Entsprechungen in der Praxis. In einem Artikel beschreibt die Sicherheitsforscherin Myriam Dunn Cavelty, dass das beschriebene Vorgehen des NSA, welches zu „WannaCry“ führte, für staatliche Geheimdienste üblich sei (Dunn Cavelty 2014, S. 10). Deren Ziel sei es, durch Sammlungen von Sicherheitslücken Vorteile bei Cyberangriffen auf andere Staaten oder terroristische Gruppen zu erlangen. Durch ihre Nachfrage an Sicherheitslücken entstehe aber ein Markt, auf dem diese gehandelt werden können (ebd.). Die staatlichen Akteure böten damit Findern einen Grund, Sicherheitslücken nicht an den Hersteller zu melden oder gar den Herstellern eine Möglichkeit, aus bekannten Sicherheitslücken in ihren Produkten Profit zu schlagen. Diese Entwicklungen deuten darauf hin, dass auch in einer konsequentialistischen Betrachtung ein Verheimlichen von Sicherheitslücken abzulehnen ist: Statt die nationale Sicherheit dank der geheimgehaltenen Informationen zu verbessern, kann sich die Sicherheit der Computernutzer dadurch auch verringern. Wenn der zweite Effekt den ersten überlagert, dann müssen Sicherheitslücken auch aus utilitaristischer Sicht an den Hersteller gemeldet werden.

3 Sind Angriffe auf Computersysteme erlaubt, um Sicherheitslücken zu finden?

Insgesamt sollten Hersteller also über Sicherheitslücken informiert werden. Dies ist aber noch keine vollständige Antwort. Denn bisher lässt die Analyse außer acht, dass Experten, die Sicherheitslücken finden möchten, in der Regel selber Angriffe auf Computersysteme durchführen. Dadurch werden Lücken aufgedeckt oder die Wirksamkeit vermuteter Angriffswege demonstriert. Das erscheint intuitiv fragwürdig. Falls es aber unmöglich ist, Lücken zu finden ohne unmoralisch zu Handeln, so wird die Pflicht diese zu Melden ebenfalls hinfällig.

Ein Artikel von Eugene Spafford deutet darauf hin, dass dies der Fall sein könnte (Spafford 1992). Spaffords Artikel beschäftigt sich mit Angriffen auf Computersysteme im Allgemeinen und betrifft damit nicht ausschließlich die hier angesprochene Situation der Sicherheitsforscher. Er möchte deontologisch

argumentierten und bezeichnet Angriffe auf Computersysteme deshalb als grundsätzlich falsch. Seiner Meinung nach sind solche Angriffe wie Einbrüche zu bewerten. Höchstens in Fällen, wo eine akute Gefahr für Leib und Leben nur durch einen Angriff abgewendet werden könnte, sei dieser erlaubt (Spafford 1992, S. 42).

In seinem Artikel versucht Spafford darüber hinaus verschiedene Einwände, die gegen seine Position angebracht werden könnten, zu entkräften (ebd., S. 43 ff.). Einige davon sind im Zusammenhang mit Sicherheitslücken besonders interessant: Das Argument, nur durch Angriffe könnte auf Softwareprobleme aufmerksam gemacht werden, lehnt er mit Verweis auf einen gegenteiligen Beispielfall ab. Ebenso verwirft er die Absicht, durch derartige Angriffe die Veröffentlichung von Sicherheitsaktualisierungen zu provozieren und somit die Sicherheit von Computersystemen zu verbessern - es könne nicht Zweck eines Computers sein, permanent Sicherheitsaktualisierungen zu erhalten. Schließlich lehnt er es auch ab, durch Angriffe auf missbräuchliche Verwendung von Benutzerdaten aufmerksam zu machen. Dies sei dazu kein wirksames Mittel, weil Computernutzer nicht so reagieren würden, wie durch den Angreifer erhofft.

Spafford hat sich bereits sehr früh mit diesem Thema auseinandergesetzt. Vor einem aktuellen Hintergrund können aber einige seiner Antworten angezweifelt werden, denn obwohl seine Grundposition deontologisch begründet ist muss er bei seinen Repliken letztlich konsequentialistisch argumentieren. Ob Angriffe die Sicherheit eines Computersystems auch erhöhen können (indem Nutzer auf Probleme aufmerksam werden und Hersteller diese Probleme lösen), ist eine empirische Frage. Über zwanzig Jahre nach der Veröffentlichung von Spaffords Artikel lässt sich hier anbringen, dass Softwarehersteller durchaus oft mit Untätigkeit auf Sicherheitslücken reagieren (siehe CNet 2010), dass Sicherheitsaktualisierungen über das Internet ausgeliefert und vollautomatisch eingerichtet werden können, ohne den Zweck eines Computers zu beeinträchtigen, und dass Nutzer weiterhin daran interessiert sind, ob ein Softwarehersteller verantwortungsvoll mit ihren Daten umgeht (siehe Pew Research Center 2016).

Auch scheinen seine Vorstellungen über die Arbeit von Sicherheitsforschern voreingenommen zu sein. So vergleicht er deren Verhalten damit, auf eine Brandgefahr in einem Kaufhaus aufmerksam zu machen, indem man das Kaufhaus damit anzündet (Spafford 1992, S. 43 f.). Einen ähnlichen Vergleich zieht er zu Milizen, die auf schlechte Wohnungsschlösser aufmerksam machen, indem sie regelmäßig in Wohnungen einbrechen (ebd.). Zu beiden Beispielen lässt sich

einwenden, dass es natürlich falsch wäre, zum Schaden von Computernutzern zu handeln und dass Forscher auch speziell präparierte Systeme angreifen können, statt die Systeme Unbeteiligter. Die Beispiele erscheinen daher pauschalisierend und irreführend.

Spaffords grundsätzliche Kritik an Angriffen auf Computersysteme ist jedoch mit dem kategorischen Imperativ kompatibel. Angriffe sind pflichtwidrig, auch wenn sie zum Ziel haben, die Sicherheit von Computernutzern zu verbessern. Denn durch den Angriff selber wird dieses Ziel bereits negiert. Die Absicht der Handlung ist es, Wissen zu schaffen, dass die Sicherheit der entsprechenden Computersysteme reduziert. Es ist somit logisch nicht denkbar, dass in einer universellen Ethik Angriffe auf Computersysteme, wenn auch mit redlichen Motiven, erlaubt wären.

Hier widersprechen sich jedoch zwei Pflichten: Auf der einen Seite gibt es eine Pflicht, Sicherheitslücken zu melden, auf der anderen Seite sind wichtige Methoden, um diese Lücken zu finden, verboten. Der kategorische Imperativ gibt also eine zwiespältige Antwort. Sicherheitsforschern blieben unter diesem Imperativ nur andere Wege, um die Sicherheit von Software zu verbessern. Dazu gehören etwa das Erarbeiten besserer Programmiertechniken und -standards, mit Hilfe derer Softwarehersteller in ihren Programmen Sicherheitslücken vermeiden können.

Aus utilitaristischer Sicht könnten Angriffe erlaubt sein, wenn diese letztendlich zu Gunsten der Computernutzer verlaufen. Sicherheitsforscher müssten aber weiterhin sorgfältig abwägen, wann sie die Nutzer einer inakzeptablen Gefahr aussetzen. Diese Abwägung betrifft auch das Thema des nächsten Abschnitts.

4 Sollten Sicherheitslücken öffentlich oder vertraulich gemeldet werden?

Unabhängig von einer Pflicht, Sicherheitslücken zu melden oder einem Verbot, diese durch Computerangriffe zu finden, sehen sich Sicherheitsforscher in der Praxis mit der Frage konfrontiert, wie sie mit ihren Erkenntnissen umgehen. Sollen sie neue Informationen über Sicherheitslücken an den Hersteller der jeweiligen Software im Geheimen übermitteln oder sollen sie diese in einem öffentlichen Medium (beispielsweise einer wissenschaftlichen Fachzeitschrift) bereitstellen? Gegensätzliche Positionen dazu wurden in zwei Internetartikeln formuliert, die in diesem Abschnitt besprochen werden.

Scott Culp, ein Mitarbeiter des Softwarekonzerns Microsoft, verlangt ein Ende der „Informationsanarchie“, wie er das uneingeschränkte Publizieren von Sicherheitslücken bezeichnet (Culp 2001). Er fordert, dass Sicherheitslücken ausschließlich dem Softwarehersteller offengelegt werden dürfen. Ein öffentliches Bekanntwerden gefährde die Sicherheit der Nutzer in einer nicht hinnehmbaren Art und Weise.

Der Sicherheitsforscher Bruce Schneier hingegen erwartet von allen Experten eine vollständige Publikation ihrer Ergebnisse (Schneier 2007). Nur dann könnten Nutzer die Sicherheit der Software, die sie einsetzen, richtig einschätzen und im Zuge dessen die Hersteller zwingen, Sicherheitsaktualisierungen bereit zu stellen.

Die Positionen unterscheiden sich also darin, wie die Interessen der Computernutzer berücksichtigt werden sollten. Nach Culp dürften die Sicherheitsforscher die Nutzer nicht darüber in Kenntnis setzen, dass ihre Software unsicher ist; jedenfalls nicht in einer Weise, die Angreifern helfen könnte, eine Lücke auszunutzen. Schneier argumentiert jedoch, dass erst diese erweiterte Information die Hersteller zwinge, eine Aktualisierung bereit zu stellen. Denn wenn die Forscher außer dem Hersteller niemanden vollständig informieren dürften, könnte dieser letztendlich auch untätig bleiben oder die Lücke leugnen.

Der kategorische Imperativ stützt zunächst die Position von Culp: Es ist verboten, mit einer Veröffentlichung denjenigen in die Hände zu spielen, die Sicherheitslücken ausnutzen würden. Denn wäre dies universelle Praxis, so würde Computersicherheit unmöglich und Nutzer gerieten permanent in Gefahr, egal, was die Intentionen der Sicherheitsforscher sein mögen. Wenn es nicht

wünschenswert ist, dass Daten entwendet werden oder Infrastruktur beschädigt wird (wie in der Einleitung beschrieben), dann kann auch eine Beihilfe dazu nicht erlaubt sein.

Ebenso wird Computersicherheit aber unmöglich, wenn sie durch die Nutzer nicht bewertet werden kann: Müsste jede Veröffentlichung von Sicherheitslücken im Geheimen geschehen, dann könnten sich Nutzer über die Sicherheit ihrer Systeme niemals ein zuverlässiges Bild machen und würden somit als unmündig behandelt. Im schlimmsten Fall führt eine ungleiche Informationsverteilung zu einem Ausnutzen der Computeranwender. So wurden die Sicherheitslücken „Spectre“ und „Meltdown“ zunächst im Geheimen an den Prozessorhersteller Intel gemeldet. Dessen Vorstandsvorsitzender stieß in der Folge Aktien des Konzerns ab, mutmaßlich in der Erwartung eines Kursverfalls, der bei Veröffentlichung der Lücken tatsächlich eintrat (heise online 2018). Gleichzeitig stehen aber auch noch Monate danach keine Sicherheitsaktualisierungen durch Intel für die Kunden bereit. Hier konnte der Vorstandsvorsitzende von Intel aus dem Zurückhalten von Informationen und der Unwissenheit seiner Kunden Profit schlagen, ohne dass für die Kunden ein Mehr an Sicherheit entstanden ist.

Der Fall illustriert aber auch ein zweites Problem: Schneiers Argument, die Nutzer könnten erst bei einer vollständigen Veröffentlichung über ihre Systeme urteilen, kann nur dann gelten, wenn den Nutzern eine bessere Alternative bereit steht. Im Falle einer Sicherheitslücke in einem Textverarbeitungsprogramm ist es etwa leicht denkbar, dass Computernutzer auf ein Konkurrenzprodukt umsteigen. Bei Fehlern der Computerbauteile, welche die Grundlage für „Spectre“ und „Meltdown“ bilden, ist ein Wechsel nicht so einfach möglich. Wenn die Nutzer auf die Verwendung eines Programms aber nicht verzichten können, dann hilft ihnen die zusätzliche Information über ihre Unsicherheit nicht und die Gefährdung durch die Veröffentlichung einer Sicherheitslücke wiegt umso schwerer.

An dem Widerspruch der Nutzerinteressen wird klar: Eine Pflicht zur geheimen Weitergabe von Sicherheitslücken an den Hersteller ist nur dann sinnvoll, wenn durch den Hersteller tatsächlich eine Sicherheitsaktualisierung bereitgestellt wird. Die nächsten beiden Abschnitte behandeln daher die Verantwortung der Hersteller.

5 Sind Softwarehersteller moralisch verpflichtet, Sicherheitslücken aus ihren Produkten zu entfernen?

Die Verfügbarkeit einer Softwareaktualisierung ist eine notwendige Bedingung dafür, dass Computersysteme, die anfällig für eine Sicherheitslücke sind, wieder sicher gemacht werden können. Zusätzlich gibt es weitere Akteure, die Handeln müssen, nachdem ein Hersteller eine Aktualisierung bereitgestellt hat. So müssen Nutzer oder Systemadministratoren diese Aktualisierung einspielen. Unter Umständen gibt es noch Distributoren, die eine Aktualisierung für ihre jeweiligen Betriebssysteme bereitstellen müssen. Hier wird jedoch nur die Rolle des Herstellers behandelt, da diese wie erwähnt notwendig, wenn auch nicht hinreichend für die Sicherheit einer Software ist.

Nach dem kategorischen Imperativ haben Hersteller eine moralische Pflicht, Aktualisierungen anzubieten. Denn andernfalls könnten Käufer niemals davon ausgehen, eine sichere Software zu erwerben: Ist klar, dass der Hersteller nicht verpflichtet ist, seine Software zu aktualisieren, dann ist jedes Sicherheitsversprechen des Herstellers bereits beim Verkauf widersprüchlich. Ein Versprechen, dass nicht eingehalten wird, ist nach einer universellen Moral sinnlos. Fehlt diese moralische Pflicht, dann wird Software letztlich ohne jegliche Garantien verkauft und ist somit für sicherheitsrelevante Anwendungen unbrauchbar. Die Maxime, Sicherheitslücken nicht aus den eigenen Programmen zu entfernen, weil dies etwa zu viel kosten würde, ist damit nicht universalisierbar.

Es lässt sich noch hinzufügen, dass eine Sicherheitslücke zunächst nur einen Programmierfehler darstellt. Ethische Relevanz bekommt dieser Fehler erst, wenn er durch einen Angreifer ausgenutzt wird. Dies wiederum setzt voraus, dass das betroffene Programm überhaupt durch irgendeinen Nutzer im Einsatz ist. Eine Sicherheitslücke in einem alten Programm, das niemand mehr verwendet, ist für die ethische Betrachtung also irrelevant. Sonst läuft die bisher begründete Pflicht auch leicht in ein Paradox: Die ältere Fassung einer Software, bei der die Aktualisierung noch nicht eingebaut wurde, ist natürlich für immer mit der Sicherheitslücke behaftet. Die moralische Pflicht des Herstellers wird jedoch immer dann wichtig, wenn Nutzer an alte Software gebunden sind, etwa, weil sie ältere Geräte verwenden. Sicherheitsaktualisierung, die einen Nutzer dazu zwingen, neue Produkte zu kaufen, sind nicht ausreichend, um die Pflicht

des Herstellers, die mit dem Verkauf des ursprünglichen Geräts begann, zu erfüllen[10].

Die Hersteller von Software mögen nun einwenden, dass die wahren Verantwortlichen bei Sicherheitsproblemen nicht sie selbst, sondern die Angreifer der Computersysteme seien. Wieso sollte ihnen eine moralische Pflicht obliegen, wenn doch die Gefahren für die Nutzer durch Dritte ausgehen? Hierzu lässt sich zunächst erwidern, dass Dritte, die persönliche Daten eines Nutzers erhalten (und dazu kann man auch die Hersteller von Software zählen, die solche Daten verarbeitet), durchaus dafür Sorge tragen müssen, dass diese Daten nicht von Anderen missbraucht werden (siehe dazu auch Sullins 2016, Abschnitt 1.1.2). Ansonsten wäre jedes Einverständnis zur Datenverarbeitung, das einem bestimmten Hersteller gegeben wird, indirekt auch ein Einverständnis dazu, dass diese Daten in andere Hände gelangen dürfen. Die ursprüngliche Einwilligung, die nur einen bestimmten Hersteller umfasste, wird dann hinfällig. Doch auch wenn dieses Einverständnis durch den Hersteller ernst genommen wird, kann es Angreifer geben, die sich darüber hinweg setzen möchten. Wenn ein Hersteller also Vorsichtsmaßnahmen getroffen hat, reicht dies aus, um die Verantwortung für einen Angriff vollständig auf die Angreifer zu übertragen?

6 Sind Softwarehersteller für alle Fehler in ihrer Software verantwortlich?

Diese Replik der Hersteller führt zu einem generellen, sehr wichtigen Punkt: Können Hersteller für Probleme auf Grund von Softwarefehlern verantwortlich gemacht werden, auch wenn sie bei Veröffentlichung der Software gar nicht von deren Existenz ausgehen konnten? In diesem Zusammenhang wird in der Literatur die sogenannte „Gefährdungshaftung“[11] für Softwarehersteller diskutiert (etwa in Johnson 1985, S. 39 ff., Nissenbaum 1996 und Szolovits 1996). Bei Gefährdungshaftung handelt es sich um eine Art von Verantwortung für einen Vorgang, der nicht verhindert werden kann (Johnson 1985, S. 42). Wenn eine solche Gefährdungshaftung für Hersteller begründbar ist, dann sind diese auch bei Angriffen von Dritten auf ihre Software verantwortlich.

Streng genommen müsste eine solche Haftung juristisch begründet werden, da es sich dabei auch um einen juristischen Begriff handelt. In diesem Abschnitt soll aber nicht die Gesetzeslage verschiedener Länder besprochen, sondern

allgemein untersucht werden, ob eine Gefährdungshaftung ethisch begründbar ist und welche weiteren Fragen sich daraus ergäben.

Deborah Johnson argumentiert in ihrem Buch „Computer Ethics“ für die Gefährdungshaftung bei Softwareprodukten (Johnson 1985, S. 51 ff.). Insbesondere betreffen ihre Argumente keine als Dienstleistung angebotene Programme, also beispielsweise keine bei Beratungsfirmen auf Bestellung angefertigte Software. Sie gelten jedoch kommerziell verfügbarer Software, die ohne Sonderanpassungen an mehrere Kunden vertrieben wird. Laut Johnson ist Gefährdungshaftung ein angebrachtes Mittel, um für diese Programme eine hohe Qualität sicherzustellen, wenn genügende Teststandards nicht zur Verfügung stehen. Auf Grund der Komplexität und den vielen verschiedenen Typen von Software seien solche Standards jedoch schwer zu definieren, weshalb Hersteller durch eine Gefährdungshaftung dazu angehalten werden sollen, die ihnen bestmögliche Qualität anzubieten. Ihr Argument ist damit letztlich utilitaristisch, da sie Gefährdungshaftung als ein taugliches Mittel sieht, um verantwortliches Verhalten der Hersteller durch die Gefahr von finanziellen Einbußen beim Haftungsfall hervorzurufen. Diese Belastung der Hersteller sei weiterhin gerechtfertigt, da der zusätzliche Entwicklungsaufwand finanziell auf die Käufer des Softwareprodukts aufgeteilt werden könne[12].

Diese Position ist also nur dann gut begründet, wenn die Wirksamkeit von Gefährdungshaftung tatsächlich empirisch belegt werden kann. Auch müsste vor einem aktuellen Hintergrund erneut erwogen werden, ob nicht inzwischen durch eine Professionalisierung der Informatik genügende Teststandards zur Verfügung stehen. Da die Komplexität von Software aber eher noch zugenommen hat, erscheint es zweifelhaft, dass Tests die Sicherheit von Software ohne Einschränkungen belegen können.

Angenommen nun, die Wirksamkeit von Gefährdungshaftung sei belegt. Da es sich um einen juristischen Begriff handelt, würden sich Rechtsabteilungen von Konzernen oder auch Gerichte um Fragen der Haftung kümmern und Haftungsfälle würden letztendlich mit Geld gelöst. Das ist pragmatisch, doch in einer ethischen Diskussion unbefriedigend. Denn es ist noch nicht geklärt, welche Person die mit der Gefährdungshaftung verbundene Verantwortung tatsächlich zugewiesen bekommen sollte.

Hier zeigt sich ein Hauptproblem mit dem Konzept der Verantwortung bei Software: Das Problem der vielen Hände, identifiziert etwa in einem Artikel der Philosophin Helen Nissenbaum (Nissenbaum 1996, S. 28 f.). Wenn, wie bei

großen Softwareprojekten üblich, viele dutzende Programmierer beteiligt sind, dann wird es laut Nissenbaum sehr schwer, die kausale Schuld an einem konkreten Problem bei bestimmten Entwicklern festzumachen. So ergebe sich aber für Programmierer die Möglichkeit, Verantwortung an andere abzuschieben, bis sich letztlich alle einer moralischen Verantwortung entziehen können. Nissenbaum unterstützt eine Gefährdungshaftung für Hersteller, da diese zumindest teilweise Verantwortlichkeit etabliere (Nissenbaum 1996, S. 39 f.).

Damit hat Nissenbaum Recht: Indem eine einzelne Person ein volles Risiko übernimmt, wird durch sie sichergestellt, dass jemand mit Sorgfalt für die Sicherheit der Nutzer vorsorgt. Das ist auch notwendig für eine moralische Pflicht zur Veröffentlichung von Softwareaktualisierungen, weil ansonsten durch die Vielzahl an potentiell Verantwortlichen die Gefahr besteht, dass immer gute Gründe vorgebracht werden können, aus denen jemand anderes diese Aktualisierung bereitstellen sollte. Die Gefährdungshaftung erscheint also nötig, um diese Pflicht tatsächlich umzusetzen. Das Problem der vielen Hände ist damit aber noch nicht gelöst. Denn welcher Person wird das Risiko angelastet? Trägt bei einer Gefährdungshaftung nur noch der Verkäufer einer Software die Verantwortung für ihre Qualität? Oder die Vorstandsvorsitzende des Herstellers? Wie verhält es sich mit der Verantwortung individueller Entwickler? Ist nur der Autor einer bestimmten Codezeile für ihre Effekte verantwortlich, wenn sich Fehler auch aus einer Wechselwirkung der Arbeit von vielen verschiedenen Programmierern ergeben können?

In einer Replik auf Nissenbaums Artikel argumentiert Peter Szolovits, dass eine Gefährdungshaftung keinen positiven Beitrag zur Softwaresicherheit leisten könne (Szolovits 1996, S. 45). Insbesondere bestehe Software in der Regel auch aus Komponenten, die von anderen Herstellern stammen und integriert werden. Bei Softwareprodukten, die aus vielen Komponenten bestehen, lasse sich somit noch viel schwerer eine persönliche Verantwortung begründen (ebd., S. 44).

Hiermit wirft Szolovits ein weiteres, sehr wichtiges Problem auf: Wenn durch eine Gefährdungshaftung immer nur der Hersteller einer von Endkunden verwendeten Software verantwortlich gemacht wird, dann haben Hersteller von einzelnen Softwarekomponenten (die also nicht an Endkunden verkauft, sondern von anderen Entwicklern in deren Programme eingebaut werden) keine Verantwortung mehr zu tragen. Denn etwaige Probleme (wie etwa Angriffe auf Grund von Sicherheitslücken) betreffen schließlich die zusammengesetzten Programme und bei diesen haftet deren Hersteller. Dieser kann aber schwer

nachweisen, dass ein Hersteller einer bestimmten Komponente verantwortlich ist, denn er hat dessen Code verarbeitet und eigenen hinzugefügt.

Als einen Ausweg schlägt Szolovits wie schon Johnson das Befolgen von Standards vor. Weiterhin solle bei der Ausbildung von Entwicklern an ihre Verantwortung appelliert werden (Szolovits 1996, S. 45). Dieser Vorschlag ist sicherlich zu unterstützen, doch er überzeugt nur dann, wenn einzelnen Entwicklern wirklich Verantwortung für Programmfehler zugeschrieben werden kann. Intuitiv scheint das wünschenswert zu sein. Wenn ein Hersteller eine Sicherheitslücke beheben muss, sollten die dortigen Entwickler die Aufgabe nicht einfach ablehnen, weil sie lieber etwas anderes programmieren würden. Im Rahmen einer Gefährdungshaftung hätten sie aber keine Verantwortung zu tragen, schließlich haftet eine andere Person ihres jeweiligen Konzerns. Andererseits handelt es sich bei den Entwicklern um jene Gruppe, die den Programmcode am besten kennt und somit am ehesten die Möglichkeit hat, das Problem zu beheben. Dies muss bei der Zuschreibung einer moralischen Pflicht, die Sicherheitslücke zu beheben, berücksichtigt werden.

Ein Eintrag in der Stanford Encyclopedia of Philosophy nennt drei Bedingungen, die gemeinsam hinreichend seien, um eine Person für ein Ereignis verantwortlich zu machen (Noorman 2016, Abschnitt 1). Zunächst müsse es einen kausalen Zusammenhang zwischen einem Ereignis und der Person geben. Diese müsse weiterhin genug informiert und in der Lage sein, die Konsequenzen ihrer Handlung abzuschätzen. Sie müsse sich außerdem frei für eine bestimmte Handlung entscheiden können. Die Glaubwürdigkeit dieser Kriterien wird dadurch unterstrichen, dass sie sich mit ähnlichen Konzeptionen in anderen Quellen decken, etwa jener bei Takanen et al. 2004, S. 95. Auch Nissenbaum nennt Kriterien, welche sich explizit auf Schuld, also Verantwortlichkeit für einen Schaden, beziehen. Eine Person habe Schuld, wenn ihr Handeln in kausalem Zusammenhang mit einem Schaden steht und dieser entweder beabsichtigt oder nicht sorgfältig verhindert wurde (Nissenbaum 1996, S. 28). Sowohl bei Nissenbaum als auch im Enzyklopädieeintrag ist also neben einem Kausalzusammenhang auch eine Abwägung der zu erwartenden Konsequenzen relevant.

Akzeptiert man die Voraussetzungen für die Zuschreibung von Verantwortung aus (Noorman 2016), dann erscheint es sinnvoll, Entwickler für Programmierfehler - und damit Sicherheitslücken im Besonderen - verantwortlich zu machen. Denn auch wenn bei komplexen Programmen ein Kausalzusammenhang zwischen Entwickler und Programmierfehler schwer zu entdecken ist, heiß

das nicht, das keiner existiert. Bei ausgebildeten Experten lässt sich außerdem davon ausgehen, dass diese über die potentiellen Folgen von Programmierfehlern und Sicherheitslücken Bescheid wissen. Außerdem haben Entwickler in der Regel auch die Freiheit, bessere Software zu schreiben, solange sie nicht beispielsweise unter Zeitdruck stehen. Das Problem der korrekten Zuweisung von Verantwortung bleibt jedoch bestehen, da die Codeteile der Entwickler miteinander interagieren.

Aus Sicht der Entwickler ließe sich nun einwenden, dass Programmierfehler erst relevant werden, wenn eine Software verwendet wird. Da die Entwickler ihre Programme aber nicht selbst an Andere zur Verwendung weitergeben, könnten sie versuchen, sich von diesbezüglichen Konsequenzen zu distanzieren; schließlich haben sie das Produkt nicht selber verkauft. So eine Verteidigung kann deshalb nicht überzeugen, da die Entwickler wissen, dass ihr Programm verkauft werden soll und sie im Gegensatz zum Verkäufer in der Lage sind, die Sicherheitseigenschaften ihres Programms zu beurteilen.

Die Programmierer können aber auch nicht die einzigen Personen sein, denen Verantwortung zugewiesen wird, etwa wenn sie gemäß eines Auftrags handeln. Verlangt etwa der Endkunde bewusst ein Produkt mit schlechten Sicherheitseigenschaften (zu Gunsten beispielsweise einer höheren Programmgeschwindigkeit), dann muss er gemäß der oben genannten Kriterien für die aus Sicherheitslücken resultierenden Angriffe auf dieses Produkt mitverantwortlich sein. Ähnlich lässt sich argumentieren, dass auch der Vorstand eines Konzerns Schuld trägt, wenn er den Entwicklern zu wenig Zeit zur Fertigstellung eines Programms gibt oder nicht ausreichend Wert auf sichere Programmiertechniken legt.

Insgesamt läuft also die im letzten Abschnitt begründete Pflicht zum Bereitstellen von Softwareaktualisierungen auf eine Gefährdungshaftung für Softwarehersteller hinaus. Gleichzeitig muss aber einzelnen Personen, die für diesen Hersteller arbeiten, sinnvoll Verantwortung zugewiesen werden. Zu diesem Zweck wäre eine Handlungsanweisung wünschenswert, die für alle mit einer Software im Zusammenhang Stehenden gilt, also sowohl für die Entwickler untereinander, als auch für Vorstandsmitglieder, Verkäufer und andere.

Mit Hilfe des kategorischen Imperativs lässt sich so eine Anweisung herleiten: Jeder muss ehrlich bezüglich der erwartbaren Sicherheitseigenschaften des Produkts, der Softwarekomponente oder des Codeteils sein, den er weitergibt

oder an einen Kunden verkauft. Das beinhaltet wahrheitsgemäße Auskunft über durchgeführte sowie fehlende Tests oder über potentielle Sicherheitsrisiken.

Angenommen, jeder handele nach dieser Maxime. Dann kann von allen Betroffenen Ehrlichkeit erwartet werden. Von einem einzelnen Entwickler bis zum Endkunden ergibt sich eine Kette von zuverlässigen Sicherheitsgarantien. Eine solche Maxime wäre also universalisierbar. Verantwortung lässt sich immer dort zuschreiben, wo jemand über die Sicherheitsaspekte eines Teils der Software lügt oder aus Fahrlässigkeit ungenügende Sicherheitsgarantien (wie etwa mangelnde Tests oder fehlende Kontrollen durch weitere Entwickler) akzeptiert. Universell angewandt kann diese Maxime zur Sicherheit von Computersystemen beitragen und das Vertrauen der Nutzer stärken, die tatsächlich jene Softwarequalität erhalten würden, die sie erwarten.

Zusammenfassend lässt sich mit dem kategorischen Imperativ eine Pflicht für Softwarehersteller herleiten, für ihre Produkte Sicherheitsaktualisierungen bereit zu stellen, solange diese noch durch jemanden genutzt werden. Dies gilt auch, wenn Angriffe für den Hersteller nicht absehbar waren. Eine Zuweisung von Verantwortung unter den Mitarbeitern eines Herstellers oder zwischen verschiedenen Herstellern kann geschehen, indem sich alle an eine allgemeine Pflicht halten, die Sicherheitseigenschaften ihres Codes offenzulegen.

7 Schluss

In dieser Arbeit wurden ethische Fragen im Zusammenhang mit Sicherheitslücken in Software behandelt. Gemäß des kategorischen Imperativs nach Kant wurde dabei zunächst eine moralische Pflicht begründet, nach der Sicherheitslücken an Softwarehersteller gemeldet werden müssen. Um diese Lücken zu finden, sehen sich Sicherheitsforscher jedoch mit dem Dilemma konfrontiert, selber in Computersysteme eindringen zu müssen. Gemäß des kategorischen Imperativs ist ihnen das verboten. Hingegen wurde hergeleitet, dass die Hersteller über vertrauliche Kanäle kontaktiert werden und Information über die Sicherheitslücken nicht der Öffentlichkeit zugespielt werden sollten. Anschließend sind die Lücken durch den Hersteller mit Softwareaktualisierungen zu beheben.

Diese Positionen lassen sich kritisieren: Zunächst einmal stellt der kategorische Imperativ nach Kant nur eine mögliche Ethikkonzeption dar und wird von verschiedenen anderen Autoren abgelehnt (siehe dazu Pleger 2017). Weiterhin ist ein Ausnutzen von Sicherheitslücken wie in der Einleitung beschrieben (also zur Entwendung von Daten oder zum Angriff auf wichtige Infrastruktur) ohnehin kategorisch verboten. Da eine Sicherheitslücke nur gefährlich wird, sobald ein Angreifer sie ausnutzt, wäre unter universeller Befolgung des Imperativs das Melden und Beheben von Sicherheitslücken letztlich egal. Natürlich ist aber eine solche universelle Anwendung des kategorischen Imperativs nicht der Fall. Weiterhin stellen sich viele der in dieser Arbeit behandelten Fragen auch im Kontext von allgemeinen Programmierfehlern, welche auch ohne einen Angreifer zu Schäden führen können. Für solche Fehler wäre es beispielsweise auch unter universeller Anwendung des Imperativs relevant, eine Pflicht zur Bereitstellung von Softwareaktualisierungen zu begründen.

Die verschiedenen in dieser Arbeit genannten Praxisbeispiele legen dar, dass es sich bei Sicherheitslücken in Computerprogrammen um ein dringendes Problem mit sehr weitreichenden ethischen Konsequenzen handelt. Es kann nicht davon ausgegangen werden, dass Entwickler und Händler von Software, sowie Sicherheitsforscher oder Angreifer von Computersystemen über eine Ausbildung in ethischen Fragen verfügen. Es erscheint fraglich, ob sich diese Gruppen in allen Fällen über die ethischen Konsequenzen ihres Handelns bewusst sind. Eine philosophische Auseinandersetzung mit diesem Phänomen ist daher sehr wünschenswert.

Die in dieser Arbeit behandelten Fragen bilden nur einen Teil des Themenkomplexes ab. Zum Ende sollen noch beispielhaft einige weitere Fragestellungen genannt werden.

In der Arbeit wurden grundsätzlich alle Sicherheitslücken als gleichwertig behandelt, unabhängig davon, welche von den in der Einleitung genannten Folgen durch sie ermöglicht werden. Könnten oder sollten diese Folgen hierarchisiert werden? Ist eine Lücke, die Änderungen an privaten Daten erlaubt, anders zu behandeln als eine Lücke, durch die Daten nur ausgelesen aber nicht geändert werden können?

Wie im Abschnitt 3 erläutert ist es für Sicherheitsforscher schwer, Lücken zu finden ohne selbst zum Angreifer zu werden. Sollten Hersteller deshalb verpflichtet sein, ihre Programme regelmäßig durch unabhängige Experten auf Schwachstellen überprüfen zu lassen?

Im Abschnitt 4 wurde angemerkt, dass eine öffentliche Publikation von Sicherheitslücken besonders schädlich ist, wenn Nutzer keine Alternativen haben, auf die sie umsteigen können. Sollte es aus diesem Grund eine Pflicht für Hersteller geben, einen solchen Umstieg möglichst einfach zu machen (zum Beispiel durch standardisierte Dateiformate, Programmversionen für verschiedene Betriebssysteme etc.)?

Im Abschnitt 5 wurde für eine Pflicht zum Bereitstellen von Aktualisierungen plädiert. Wie soll jedoch mit Herstellern umgegangen werden, die sich nicht an diese Pflicht halten? Sollten sie zum Beispiel für Kosten aufkommen, die den Nutzern durch einen Wechsel zu Konkurrenzprodukten entstehen? Gibt es auch eine Pflicht für Nutzer, Sicherheitsaktualisierungen einzuspielen?

Das in Abschnitt 6 vorgestellte Konzept der Gefährdungshaftung wurde nur im Zusammenhang mit kommerzieller Software diskutiert. Wie verhält es sich mit quelloffener Software[13], die maßgeblich von Ehrenamtlichen entwickelt wird? Diese Frage ist auch deshalb besonders wichtig, da von einigen Seiten (etwa in Hoepman und Jacobs 2007) gefordert wird, besonders sicherheitskritische Software ausschließlich als quelloffene Software zu entwickeln. Ist diese Forderung berechtigt?

All diese Fragen betreffen - wenn auch indirekt und vielleicht sogar unbewusst - den Alltag vieler Menschen. Für Philosophen ergibt sich die Möglichkeit, durch die Diskussion dieser Fragen zu einem verantwortungsvollen und menschlichen Umgang in einer digitalisierten Gesellschaft beizutragen.

Endnoten

[1]Drahtlose Netzwerke (WLAN) werden häufig benutzt, um Laptops oder Smartphones mit dem Internet oder untereinander zu verbinden. WPA2 ist ein Standard zum Schutz drahtloser Netzwerke vor dem Zugriff unberechtigter Dritter.

[2] Betriebssysteme stellen auf jedem Gerät die grundlegende Funktionalität der Bauteile (wie etwa Ein- und Ausgabe) zur Verfügung. Software wird meistens für ein bestimmtes Betriebssystem entwickelt und kann dank diesem bis zu einem gewissen Grad von den technischen Besonderheiten eines bestimmten Geräts abstrahieren. Bekannte Betriebssysteme sind etwa Microsoft Windows, Apple Mac OS und GNU/Linux.

[3]Eine Softwareaktualisierung ist eine neue Fassung eines Computerprogramms, die Programmierfehler behebt oder neue Funktionen hinzufügt.

[4]Die Begriffe „Software“ und „Computerprogramm“ werden in dieser Arbeit synonym verwendet. Software besteht aus Anweisungen, welche von Programmierern geschrieben und durch das Betriebssystem von den Bauteilen eines Computers ausgeführt werden. Sie sorgt dafür, dass ein Computer Eingaben über Tastatur, Maus, Touchscreen etc. aufnimmt und Ausgaben über Lautsprecher und Bildschirm produziert. Beispiele für Software sind Textverarbeitungsprogramme, Internetbrowser und Videospiele.

[5]Takanen et al. beschreiben die für die Sicherheit von Software relevanten Aspekte nicht weiter. Für den Zweck dieser Arbeit reicht es, Sicherheitslücken als all jene Programmierfehler zu sehen, durch welche sich die in dieser Einleitung beschriebenen Gefahren ergeben.

[6]Mit den Anfang des Jahres 2018 veröffentlichten Angriffen „Spectre“ und „Meltdown“ wird diese Definition fragwürdig. Bei jenen handelt es sich um Sicherheitsprobleme, die nicht aus einer Software, sondern aus den Bauteilen eines Computers resultieren. Die daraus entstehenden ethischen Probleme (bei „Spectre“ und „Meltdown“ das Auslesen privater Daten) ähneln jedoch denen von Sicherheitslücken in Software. Die Definition muss also unter Umständen verändert werden.

[7]Zur Beeinträchtigung britischer Krankenhäuser durch „WannaCry“, das auf einer Sicherheitslücke im Betriebssystem Windows basiert, siehe (BBC 2017). „WannaCry“ wird auch im Abschnitt 2 behandelt.

[8]Wenn Kant fordert, dass eine Maxime zum allgemeinen Gesetz werden können muss, dann bedeutet das zum Einen, dass sie überall, also unabhängig von Vorlieben, Kulturen etc. zu gelten habe. Weiterhin muss die Maxime als Gesetz widerspruchsfrei sein, also zu keinen logischen Problemen führen, wenn sie allgemein angewandt wird.

Siehe zu diesen Kriterien (Kant 1911) und (Pleger 2017, S. 96). In dieser Arbeit wird eine Maxime, welche diese Anforderungen erfüllt, als universalisierbar bezeichnet.

[9]Erpressungsprogramme wie „WannaCry“ (im Englischen auch als „Ransomware“ bezeichnet) zwingen einen Computernutzer dazu, gegen Geld ein Zerstören oder Veröffentlichen privater Daten zu verhindern. Im Fall von „WannaCry“ wurden beispielsweise private Daten von Opfern verschlüsselt. Um diese zurückzuerhalten, mussten Betroffene eine Geldsumme an die Erpresser überweisen.

[10]Diese moralische Pflicht trifft jedoch in der Praxis auf erhebliche Probleme. Wenn auch sehr alte Produkte aktualisiert werden müssen, nur weil noch mindestens ein Käufer sie verwendet, so kann dies unter Umständen den technischen Fortschritt blockieren. Da neuere Geräte auch rein durch ihre Bauart bessere Sicherheitseigenschaften haben können, ginge das zu Lasten der Nutzer. Weiterhin kann insbesondere für kleinere Hersteller eine solche Pflicht eine enorme Bürde sein und ihre Konkurrenzfähigkeit reduzieren, was die Entstehung von Monopolen großer Hersteller zu Ungunsten der Nutzer begünstigt. Doch es gibt aus einer praktischen Perspektive auch Argumente, die für eine Pflicht zu Aktualisierungen sprechen: Zum Einen geht mit fehlenden Softwareaktualisierungen ein enormer Wertverlust für existierende Geräte einher. Zum Anderen hat ein faktischer Zwang zum Kauf immer neuer Geräte auch die bereits in der Einleitung genannten negativen ökologischen Folgen.

[11]Im Englischen wird dies als „strict liability“ bezeichnet. Wörtlich bedeutet das „strikte Haftung“.

[12]Aus praktischer Sicht lässt sich hier wieder einwenden, dass eine Gefährdungshaftung größere Hersteller unfair begünstigen könnte. Nicht nur, da diese mehr finanzielle Mittel haben, um einen Haftungsfall zu stemmen, während dieser einen kleineren Hersteller in den Ruin treiben könnte. Auch haben große Konzerne eine höhere Kundenzahl, sodass es ihnen einfacher fallen dürfte, das finanzielle Risiko der Gefährdungshaftung auf ihre Kunden aufzuteilen und dabei konkurrenzfähige Preise zu halten.

[13]Quelloffene Software wird in der Öffentlichkeit entwickelt. Dies bedeutet, dass der Programmcode von jedem einsehbar und verwendbar ist. Grundsätzlich ist quelloffene Software damit für alle kostenlos verfügbar und hat selten einen einzigen Hersteller. Das Betriebssystem GNU/Linux oder der Internetbrowser Firefox gehören zu diesem Typ Software.

Literatur

BBC. *NHS cyber-attack: GPs and hospitals hit by ransomware*. 2017. URL: http://www.bbc.com/news/health-39899646 (besucht am 04. 02. 2018).

CNet. *Forcing vendors to fix bugs under deadline*. 2010. URL: https://www.cnet.com/news/forcing-vendors-to-fix-bugs-under-deadline/ (besucht am 04. 02. 2018).

Culp, Scott. *It's Time to End Information Anarchy*. 2001. URL: https://web.archive.org/web/20011109045330if_/http://www.microsoft.com:80/technet/treeview/default.asp?url=/technet/columns/security/noarch.asp (besucht am 19. 11. 2017).

Deutsche Umwelthilfe. *Nachhaltigkeit von Geschäftsmodellen in der Informations- und Kommunikationstechnik. Analyse und Empfehlungen am Beispiel von Smartphone, Telefon und Router*. 2018. URL: http://www.duh.de/fileadmin/user_upload/download/Projektinformation/Kreislaufwirtschaft/Elektroger%C3%A4te/180115_DUH-Studie_Nachhaltigkeit-IKT-Industrie.pdf (besucht am 04. 02. 2018).

Dunn Cavelty, Myriam. „Breaking the Cyber-Security Dilemma: Aligning Security Needs and Removing Vulnerabilities". In: *Science and Engineering Ethics* 20.3 (2014), S. 701–715.

heise online. *Prozessor-Bug: Intel-Chef stieß hunderttausende Aktien ab, Börsenkurs sackt ab*. 2018. URL: https://www.heise.de/newsticker/meldung/Prozessor-Bug-Intel-Chef-stiess-hunderttausende-Aktien-ab-Boersenkurs-sackt-ab-3932649.html (besucht am 04. 02. 2018).

— *WannaCry: Was wir bisher über die Ransomware-Attacke wissen*. 2017. URL: https://www.heise.de/newsticker/meldung/WannaCry-Was-wir-bisher-ueber-die-Ransomware-Attacke-wissen-3713502.html (besucht am 04. 02. 2018).

Hoepman, Jaap-Henk und Bart Jacobs. „Increased Security Through Open Source". In: *Communications of the ACM* 50.1 (2007), S. 79–83.

Johnson, Deborah G. *Computer Ethics*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

Kant, Immanuel. *Grundlegung zur Metaphysik der Sitten*. Akademieausgabe, 1911.

Nissenbaum, Helen. „Accountability in a computerized society". In: *Science and Engineering Ethics* 2.1 (1996), S. 25–42.

Noorman, Merel. „Computing and Moral Responsibility“. In: *The Stanford Encyclopedia of Philosophy*. Hrsg. von Edward N. Zalta. Winter 2016 Edition. 2016. URL: https://plato.stanford.edu/archives/win2016/entries/computing-responsibility/.

Pew Research Center. *The state of privacy in post-Snowden America*. 2016. URL: http://www.pewresearch.org/fact-tank/2016/09/21/the-state-of-privacy-in-america/ (besucht am 04. 02. 2018).

Pleger, Wolfgang. *Das gute Leben - Eine Einführung in die Ethik*. Stuttgart: J. B. Metzler, 2017.

Schneier, Bruce. *Full Disclosure of Security Vulnerabilities a 'Damned Good Idea'*. 2007. URL: https://www.schneier.com/essays/archives/2007/01/schneier_full_disclo.html (besucht am 19. 11. 2017).

Spafford, Eugene H. „Are computer hacker break-ins ethical?“ In: *Journal of Systems and Software* 17.1 (1992), S. 41–47.

Sullins, John. „Information Technology and Moral Values“. In: *The Stanford Encyclopedia of Philosophy*. Hrsg. von Edward N. Zalta. Spring 2016 Edition. 2016. URL: https://plato.stanford.edu/archives/spr2016/entries/it-moral-values/.

Szolovits, Peter. „Sources of error and accountability in computer systems: Comments on “accountability in a computerized society”“. In: *Science and Engineering Ethics* 2.1 (1996), S. 43–46.

Takanen, Ari et al. „Agents of responsibility in software vulnerability processes“. In: *Ethics and Information Technology* 6.2 (2004), S. 93–110.

Vanhoef, Mathy und Frank Piessens. „Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2“. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17. New York, NY, USA: ACM, 2017, S. 1313–1328.