Urheberrecht ./. Sicherheitsanalyse

Zivilrechtliche Aspekte der Sicherheitsforschung

Nico Müller, Tilo Müller, Felix Freiling[1]

Abstract: 2007 hat die Bundesregierung im Rahmen des 41. Strafrechtsänderungsgesetzes zur Bekämpfung von Computerkriminalität den umstrittenen Tatbestand des Ausspähens und Abfangens von Daten, inklusive deren Vorbereitung, unter Strafe gestellt. Durch das schon damals äußerst umstrittene Gesetz drohte die Tätigkeit von Sicherheitsforschern kriminalisiert zu werden, weswegen eine Klarstellung der Gesetzeslage durch eine Verfassungsbeschwerde erzwungen wurde. Jedoch wird neben der strafrechtlichen Relevanz von Sicherheitsanalysen oftmals die zivilrechtliche Seite des Urheberrechts übersehen, die eine beinahe ebenso große Bedrohung für Forscher darstellt, die sich kritisch mit der Sicherheit von Software auseinandersetzen. Im Folgenden soll daher die rechtliche Lage zu Sicherheitsanalysen kommerzieller Software für Forschungszwecke erörtert und die existierenden Grauzonen rechtlicher Unsicherheit ausgeleuchtet werden.

Keywords: Urheberrecht, Sicherheitsanalyse, Reverse Engineering

1 Einführung

Seit Anbeginn der menschlichen Zivilisation wurden im Laufe der Zeit immer neuere Technologien entwickelt. Während sich in den ostasiatischen Kulturkreisen der Grundsatz, dass Ideen nicht schützbar sind, durchgesetzt hat, bildete sich in den westlichen Ländern das Patentwesen und Urheberrecht. Das Urheberrecht garantiert dem Urheber für seine Werke eine gewisse Exklusivfrist, um ihm eine Vergütung seiner Arbeit zu gewähren. In Deutschland hat der Gesetzgeber erst im Laufe der 1990er Jahre mit dem WIPO-Urheberrechtsvertrag die Grundlagen für die geistigen Eigentumsrechte an digitalen Medien gelegt. Erste Implementierungen fanden 2001/2003 durch die EG-Softwarerichtlinie und die Novellierung des Urheberrechtsgesetzes statt.

Nicht jede neue Technologie ist von Anfang an sicher, weswegen sich insbesondere Firmen im Laufe der letzten Jahre die Frage gestellt haben, ob verwendete Computerprogramme Sicherheitslücken enthalten, die wie im Falle von STUXNET [FMC12] und dem WPA2-Key-Reinstallation-Angriff [VP17] von Dritten ausgenutzt werden können, um schwere Schäden an Firmennetzwerken anzurichten. Damit Sicherheitsforscher proaktiv solche Gefahren abwenden können, stehen sie im Konflikt mit den Interessen der Softwareentwickler, die

[1] Friedrich-Alexander-Universität Erlangen-Nürnberg, Department Informatik, Martensstr. 3, 91058 Erlangen, nico.mueller, tilo.mueller, felix.freiling@fau.de

doi:10.18420/sicherheit2020_08

oft explizit das *Reverse Engineering* der zu untersuchenden Software untersagen. Im Fall gekaufter Software hat der ursprüngliche Entwickler keine über die Urheberpersönlichkeitsrechte hinausgehenden Ansprüche mehr. Da die meisten Entwickler aber üblicherweise ein Lizenzmodell verwenden, behält sich eine Entwicklungsfirma als Lizenzgeber alle Rechte vor. Dennoch sieht der Gesetzgeber Ausnahmen vor, die durch Lizenzvereinbarungen nicht berührt werden dürfen. Aufgrund dieser Tatsache gestaltet sich die rechtliche Situation des Interessenkonflikts zwischen Lizenzgeber (Softwareentwickler) und Lizenznehmer (Sicherheitsforscher) als schwierig. Zur praktischen Orientierung leuchtet dieser Beitrag die Grauzonen rechtlicher Unsicherheit in diesem Kontext aus.

2 Rechtliche Natur eines Computerprogramms

Um weitere Aussagen über die rechtliche Situation zur Sicherheitsanalyse von Computerprogrammen zu treffen, wird zuerst die Natur eines Computerprogramms untersucht. Ein Computerprogramm besteht prinzipiell aus zwei juristisch relevanten Teilen: Schnittstellen, die eine Ein- und Ausgabe zu Menschen oder anderen Computerprogrammen ermöglichen und meist einem Standard folgen, sowie der eigentlichen Leistung des Entwicklers, den verarbeitenden Algorithmen, welche in Kombination miteinander die Funktionalität des Programms gewährleisten. Beide Teile und deren Unterkomponenten müssen sicher sein, um das Programm als Ganzes sicher zu machen. Als *Computerprogramm* wird lediglich der auszuführende Code gesehen, im Falle des Vorhandenseins von Begleitmaterialien, wie Grafiken und Musik, wird von *Computersoftware* gesprochen.

2.1 Schnittstellen

In der Computerprogramm-Richtlinie 2009/24/EG der Europäischen Union wird eine Schnittstelle im 10. Erwägungsgrund wie folgt definiert „Die Teile des Programms, die eine [...] Verbindung und Interaktion zwischen den Elementen von Software und Hardware ermöglichen sollen, sind allgemein als Schnittstellen bekannt". Diese Schnittstellen sind oft standardisiert und technisch notwendig, was gegen eine urheberrechtliche Schutzfähigkeit spricht. In der Tat wird die Schutzfähigkeit von technisch notwendigen „Designentscheidungen" in der Rechtswissenschaft verneint (vgl. BGH, 12.05.2011 - I ZR 53/10 - Seilzirkus). Zu dieser Einschätzung kommt auch die Fachliteratur [Ma18].

Dennoch könnten Abweichungen von dem Standard, einen urheberrechtlichen Schutz des Teilwerks nach sich ziehen, da diese eine eigene geistige Leistung darstellen. In der Rechtswissenschaft wird dabei der Begriff der sogenannten *kleinen Münze* verwendet, bei dem es sich um das kleinstmögliche, aber dennoch schutzfähige Werk handelt. Triviale Änderungen spielen bei der Anwendbarkeit dieser Regelung allerdings keine Rolle.

2.2 Algorithmen

Bei der Datenverarbeitung werden zwar oft Bibliotheksfunktionen verwendet, allerdings werden diese in ihrem Zusammenspiel ein schützenswerter Ausdruck der dahinterliegenden Idee. So ist ein hinreichend kleiner Teil eines jeden urheberrechtlich geschützten Werks nicht mehr urheberrechtlich geschützt, da ab einem gewissen Punkt keine persönliche Leistung mehr erkennbar ist. Dieser Punkt wird als, wie bereits oben genannt, kleine Münze bezeichnet. Für die verschiedenen vom Urheberrecht geschützten Werksarten galten verschiedene Maßstäbe, welche in den letzten Jahren auf ein einheitliches Niveau abgesenkt wurden.Die genauen Grenzen sind in den Gesetzestexten nicht ersichtlich, da meist für die Interpretation offene Formulierungen genutzt werden. So hat Bisges [Bi14, S. 224ff.] in seinem Buch konkrete Fälle der kleinen Münze ausgewertet und ist zu dem Schluss gekommen, dass für die *Vermutung der Schutzfähigkeit* vielmehr der investierte Aufwand und die wirtschaftliche Signifikanz der Werke für eine Entscheidung ausschlaggebend sind. Daraus ableitend kann man sagen, dass die meisten zu untersuchenden Computerprogramme in einem Gerichtsverfahren als schützenswert gelten werden.

3 Schutzbereich des Urheberrechts

Zuallererst stellt sich die Frage, was genau vom Urheberrecht erfasst ist. Die entsprechenden Vorschriften finden sich dabei im Urheberrechtsgesetz §69 a-g, wobei diese ein sogenanntes „lex specialis" zum normalen Urheberrecht bilden, d.h. die dort verankerten Grundsätze haben Vorrang vor anderen Definitionen, die beispielsweise in §2 UrhG getroffen werden.

So findet man in §69a die Bestimmungen zum Schutzgegenstand, die sich wesentlich von den in §2 festgelegten Kriterien unterscheiden. Dort ist von „individuelle[n] Werken in dem Sinne [...], dass sie das Ergebnis der eigenen geistigen Schöpfung des Urhebers sind" die Rede, was jedoch mit der sonst geforderten „persönliche[n] geistige[n] Schöpfung" nicht deckungsgleich ist, wie Marly beschreibt [Ma18]. Zielsetzung dieser Gesetzgebung ist es, dass möglichst alle Programme vom Urheberrechtsschutz abgedeckt sind, sofern sie nicht äußerst einfach strukturiert sind.

Eine Festlegung auf Programmcode ist dabei nicht erforderlich, sofern der Schöpfungsgedanke hinreichend fixiert worden ist. Dies umfasst beispielsweise Diagramme und technische Grafiken, die mit Zeichnungen eines Architektengebäudes vergleichbar sind, welche ebenfalls Urheberrechtsschutz genießen.

Um den komplexeren Sachverhalt der Schutzfähigkeit zu erörtern, wird nun für die Fälle eines kleinen Programms und eines komplexeren Programms eine Prüfung der Schutzfähigkeit anschaulich vorgenommen.

3.1 Beispiel Hallo-Welt-Programm

Das Hallo-Welt-Programm ist das seit jeher implementierte Programm, mit dem eine Programmiersprache erlernt wird. Bei dieser Art von Programm ist es die Aufgabe für Anfänger beispielsweise eine einfache Botschaft auf der Kommandozeile auszugeben. Dabei lässt sich eine nicht unerhebliche Banalität des Programms feststellen die, neben der quasi-Standardisierung, ein individuelles Werk nicht gestatten. So ist insbesondere bei Kleinstprogrammen, wie dem Hallo-Welt-Beispiel, der technische Hintergrundgedanke überwiegend, nicht jedoch der persönliche Schöpfungsgedanke, was einen urheberrechtlichen Schutz ausschließt. Marly knüpft die Schutzvermutung an die tatsächliche Komplexität des Programms, wobei jedoch darauf verwiesen wird, dass es sich lediglich um eine Vermutung, nicht aber um die tatsächliche Schutzfähigkeit handelt [Ma18].

3.2 Beispiel Professionelles Programm

Falls der Programmumfang größer sein sollte, insbesondere wenn das Programm von einer Firma entwickelt wurde, ist davon auszugehen, dass das Programm in den Schutzbereich des Urheberrechts fällt. Jedoch ist, wie bereits oben erwähnt, meist nur ein Indiz für die Individualität eines Programms vorhanden. Ob und inwiefern der persönliche Charakter eines Programms durch Übersetzungen und Optimierungen, wie sie von einem Compiler vorgenommen werden, zerstört wird, wurde jedoch bisher in der Fachliteratur nicht erörtert. Dennoch kann man sagen, dass wenn zur Lösung eines konkreten Problems, beispielsweise der symmetrischen Verschlüsselung von Daten, es aus technischer Sicht nur eine oder wenige Wahlmöglichkeiten gibt, hier AES oder 3DES, eine Schutzfähigkeit nicht gegeben ist. Die Rechtssprechung sagt dazu, dass es „an der nötigen Individualität [fehlen kann], wenn betriebswirtschaftliche, technische und funktionale Zwänge, etwa solche der Effizienz oder solche, die auf externen Faktoren (z.B. Kompatibilitätsanforderungen) oder auf dem jeweiligen Sachgebiet beruhen, den Gestaltungsspielraum einschränken" [Wa19]. Konkret war die GUI eines Programms, das zur Verwaltung durch den Betriebsrat entwickelt wurde, nach Ansicht des OLG Karlsruhe - 13.06.1994 6 U 5294 - „Bildschirmmasken" nicht schutzfähig, da diese „durch Gesetz oder durch Betriebsvereinbarungen und betriebliche Übung festgelegt" war.

4 Eingriff in den Schutzbereich

Um eine Beurteilung der während einer Sicherheitsanalyse typischen Untersuchungsmöglichkeiten vorzunehmen, werden diese im Folgenden vorgestellt. Da eine Vervielfältigung des Programms bei allen vorgestellten Analysemethoden einen wesentlichen Bestandteil darstellt, muss zwingend mindestens eine Nutzungsberechtigung der Software vorliegen, damit die Analyse erlaubt ist.

4.1 Blackboxing

Beim sogenannten *Blackboxing* [BZ17] wird das Programm geladen und anhand verschiedener Eingaben getestet, wobei die Ausgaben des Programms betrachtet werden. Da dadurch ein möglicher Anwendungsfall simuliert wird, handelt es sich um eine erlaubte Handlung nach §69d Abs. 3 UrhG. Da keine darüberhinausgehende Interaktion mit dem Programm stattfindet, bleibt der Quellcode unberührt und damit ist, falls eine Benutzungsberechtigung vorliegt, kein Eingriff in das Urheberrecht gegeben. In der Praxis eignet sich Blackboxing meist nur für die oberflächliche Suche nach Fehlern, insbesondere eignet es sich nicht dazu die Ursache eines Fehlers zu identifizieren. So können unübliche Eingaben, wie sie beispielsweise durch *Fuzzing* generiert werden, Fehlerzustände verursachen. Die technischen Gründe, warum ein Fehler auftrat, können aber nicht geklärt werden.

4.2 System Monitoring

Das *System Monitoring* stellt eine Sonderform des Blackboxing dar. Dabei wird die Kommunikation des Programms mit anderen Programmen und dem Betriebssystem überwacht, wobei die internen Vorgänge des Programms nicht genauer untersucht werden. Zwar sind Systemaufrufe und gesendete Pakete ein signifikanter Bestandteil eines jeden Programms, allerdings kann ähnlich wie oben argumentiert werden, dass der eigentliche Quellcode nicht betrachtet wird, sondern nur die Auswirkungen einer Programmausführung. Dadurch lassen sich Informationen über den Programmablauf auslesen, so dass sich System Monitoring wie Blackboxing zur initialen Fehlersuche eignet. Allerdings stellt auch das System Monitoring kein ausreichendes Werkzeug dar, um technische Ursache eines Fehlers zu identifizieren. Aus juristischer Sicht sind entstehende Ausgaben wie Pakete und Systemaufrufe durch Veranlassung des Bedieners entstanden, wenn auch unter Zuhilfename des zu untersuchenden Programms. Damit sind die so erzeugten Daten nicht durch das Recht des Programmurhebers geschützt.

4.3 Statische Analyse

Bei der statischen Analyse wird der zu untersuchende Code in seiner Gesamtheit von Maschinencode in Assemblercode und anschließend, je nach Analysesoftware bzw. Interesse seitens der Analysten, in eine höhere Programmiersprache umgewandelt.

Auf den ersten Blick betrachtet setzt also die statische Analyse eine Übersetzung von Maschinen- in Assemblercode und bei weiteren Analysen eine Übersetzung in eine höhere Programmiersprache voraus. Somit stellt die statische Analyse eine genehmigungspflichtige

Im weitesten Sinne stellt die Umwandlung ebenfalls eine Vervielfältigung dar, wobei nach Ansicht der Autoren diese Vervielfältigung nach §69 d Abs. 2 f UrhG erlaubt ist.

Handlung nach §69c UrhG dar und ist reglementiert nach §69e UrhG. Jedoch stellen sich essenzielle Fragen aus der Sicht des Untersuchenden, darunter

- Stellt eine Disassemblierung eine Umarbeitung oder eine Übersetzung im Sinne des §69c UrhG dar?
- In welchem Umfang ist die statische Analyse eines Programms erlaubt?
- Ist eine (Teil-)Nachprogrammierung/Dekompilierung des ursprünglichen Quellcodes noch durch das Urheberrecht geschützt?

Zum ersten Punkt kann mit ziemlicher Sicherheit gesagt werden, dass durch Disassemblierung keine Übersetzungshandlung im Sinne von §3 UrhG gegeben ist. Grund dafür ist, dass um ein urheberrechtlich geschütztes Werk zu erzeugen eine „persönliche geistige Schöpfung" nach §2 UrhG erforderlich ist. Da in den meisten Fällen die Disassemblierung eines Programms nach fest definierten Regeln durchgeführt wird, kann nicht von einer persönlichen geistigen Schöpfung die Rede sein. Dies sieht auch Schweyer [Sc14, S. 97 f.] ähnlich.

Der Punkt der Umarbeitung ist jedoch bedeutend interessanter, da bei einer Be- beziehungsweise Umarbeitung kein eigenständig schutzfähiges Programm entstehen muss. Die Umarbeitung ist dabei als Erweiterung des Bearbeitungsbegriffs zu verstehen, welche auch die Übersetzung, das Arrangement und andere Formen der Umarbeitung erfasst [Ma18]. Dabei hat §69c Abs. 2 Vorrang vor der in §23 definierten Bearbeitung, die das Kriterium der Eigenständigkeit verlangt. Darüber hinaus ist schon die Vornahme der Umarbeitung eine in §69c gelistete Handlung, und damit genehmigungspflichtig.

Zum zweiten Punkt gibt es nach Kenntnis der Autoren noch keine gerichtliche Entscheidung. Im 2. Gesetz zur Änderung des Urheberrechtsgesetzes [Bu93] findet sich „§69 d Abs. 3 ist nicht nur auf einzelne Programmelemente, sondern auch auf vollständige Programme anzuwenden." Damit impliziert der Gesetzgeber, dass eine Betrachtung des Assemblercodes für das gesamte Programm zulässig ist, da eine teilweise Betrachtung nur durch Betrachten der Teile des Assemblercodes möglich ist, wobei eine Dekompilierung des Programms in seiner Ganzheit wie unten beschrieben trotzdem nicht erlaubt ist. Weitere Bestätigungen dieser Sichtweise lassen sich in der Fachliteratur finden [Ka18].

4.4 Dekompilierung

Bei der *Dekompilierung* wird der Objektcode automatisiert in eine höhere Programmiersprache umgewandelt. Eine originalgetreue Rekonstruktion des ursprünglichen Quellcodes ist in der Praxis unmöglich. Automatisierung der Rückübersetzung wurde zwar versucht, allerdings mit begrenztem Erfolg, beispielsweise mittels „boomerang" [Pr06]. Eine Dekompilierung von beschränkten Programmteilen kann dagegen ohne Weiteres möglich sein.

Dennoch ist es aus praktischer Sicht fast unmöglich, den exakten Programmcode aus dem gegebenen Assemblercode zu rekonstruieren. Wo die Grenzen zwischen Nachahmung und Übernahme der Funktionalität liegen, kann im Allgemeinen auch nicht beantwortet werden.

Etwas schwieriger verhält es sich mit der Nachprogrammierung in einer höheren Sprache. Die Idee hinter einer jeden Software ist durch das Urheberrecht zwar nicht geschützt, allerdings die konkrete Ausdrucksweise, also das Programm an sich, ist es. Bei einem Plagiat, ob 1:1 übernommen oder nachgeahmt, das zum Ziel hat, den vorliegenden Assemblercode in einer höheren Programmiersprache nachzubilden, ist von einer Umarbeitung auszugehen [Ma18].

Rechtlich betrachtet ist anzunehmen, dass wenn es sich um ein vom Urheberrecht geschütztes Programm handelt, die Dekompilierung untersagt ist. Nicht nur stellt die Dekompilierung eine Umarbeitung dar, sondern auch eine Art Übersetzung, da Computerprogrammen den Sprachwerken zugeordnet werden. Die Dekompilierung wird durch die gesetzliche Schranke in §69e nur dann erlaubt, wenn eine Bereitstellung des Quellcodes durch den Urheber nicht erfolgt ist und Interoperabilität zu einem anderen Programm hergestellt werden muss. Wenngleich eine Teildekompilierung zwar wahrscheinlich erlaubt ist, insbesondere mit Hinblick auf die fehlende Schutzfähigkeit von Kleinstprogramm(-abschnitten) oder Schnittstellen, ist bei der vollständigen Dekompilierung von Illegalität auszugehen.

4.5 Dynamische Analyse

Die dynamische Analyse basiert darauf, dass das Programm im laufenden Zustand betrachtet wird. Dabei wird der Inhalt des Arbeitsspeichers und der Register ausgelesen und dargestellt.

Ein Beispiel für ein solches Programm ist der *GNU Enhanced Debugger* (gdb) [Fo19]. Dargestellt in der Benutzeroberfläche wird konkret der Registerinhalt, der Stack und der noch auszuführende Code, allerdings lassen sich durch Werkzeuge dieser Art auch alle Daten, die durch die Ausführung des Programms im Arbeitsspeicher vorhanden sind, ohne Weiteres auslesen. Ähnlich wie bei der statischen Analyse wird hierbei der Bytecode in Assemblercode umgewandelt. Hier ist jedoch der große Unterschied, dass nur ein kleiner Teil des Programmcodes ausgewertet wird, ohne das Gesamtprogramm zu übersetzen.

Dies ermöglicht eine genaue Betrachtung der internen Programmablaufe unter Realbedingungen und ist ein gängiges Mittel beim Finden von Sicherheitslücken dar. In Verbindung mit der statischen Analyse stellt sie das faktisch notwendige Werkzeug dar, um eine gründliche Analyse durchführen zu können. Jedoch gibt es auch hier rechtliche Bedenken, wenn es um die Kontrollflussveränderung von Programmen geht. So hat das OLG Hamburg, 23.04.2012 - 5 U 11/11, entgegen der Vorinstanz entschieden, dass eine Umarbeitung schon gegeben ist, wenn diese nicht manifestiert wurde, also nur im Arbeitsspeicher stattfindet. Entgegen dieser Entscheidung steht jedoch die Ansicht des LG München I, 27.05.2015 - 37 O 11673/14, ebenso die Ansicht des OLG München, die in dem als Adblock-Prozess bekannten Verfahren

In diesem Fall fällt das Teilprogramm aus der sog. kleinen Münze heraus.

einen Eingriff in den Kontrollfluss, sofern dieser nur zur Laufzeit vorgenommen wird, nicht ausreichend manifestiert sei, um eine Umarbeitung zu begründen. So heißt es in dem Urteil „Dieser Schutz des Rechteinhabers vor der unberechtigten Herstellung einer Bearbeitung eines Computerprogramms macht nach Ansicht der Kammer dann Sinn, wenn in das Computerprogramm als solches eingegriffen wird, insbesondere in den Quellcode, nicht jedoch, wenn im Rahmen einer Vervielfältigung im Arbeitsspeicher lediglich ein Teil eines Programms unterdrückt bzw. nicht aufgerufen wird." Jedoch konnte nicht abschließend geklärt werden, inwiefern sogenanntes „Patching" eine Umarbeitung darstellt, da es meist einen wesentlichen Bestandteil der Kontrollflussveränderung darstellt.

5 Gesetzliche Lizenzen

In fast jedem Lizenztext findet man Vertragsklauseln, die explizit eine Dekompilierung oder Disassemblierung des Programmcodes verbieten. Diese sind jedoch ausschließlich für den amerikanischen Rechtsraum relevant, da der deutsche Gesetzgeber in §69g Abs. 2 UrhG solche Klauseln für nichtig erklärt, so es sich um erlaubte Testhandlungen handelt.

Der deutsche Gesetzgeber sieht ebenfalls eine gesonderte Regelung für nicht kommerziell forschende Stellen vor. So darf man nach §60c UrhG bis zu 15% eines Werkes in einer nicht öffentlichen, klar abgegrenzten Umgebung verbreitet werden, bei der eigenen Forschung dürfen bis zu 75% eines urheberrechtlich geschützten Werkes kopiert werden, sofern das Werk der Hauptgegenstand der Forschung ist.Eine solche nicht öffentliche Umgebung stellt zum Beispiel eine Konferenz mit weiteren Wissenschaftlern dar, bei der das entsprechende Material nur für Besucher der Konferenz zugänglich gemacht wird.

5.1 Sonderfall Schadsoftware

Schadsoftware stellt eine spezielle Herausforderung an den Gesetzgeber und an den Untersuchenden dar, da eine uneingeschränkte Sicherheitsanalyse wünschenswert ist. Um eine dynamische Analyse zu verhindern setzt manche Schadsoftware Maßnahmen ein, um zum Beispiel virtuelle Arbeitsumgebungen zu erkennen [WF12]. Falls diese von Analysten umgangen werden, könnten sich Analysten theoretisch nach §95a UrhG schuldig machen, wobei es sich bei dem untersuchten Virus um Computersoftware, nicht aber um ein Computerprogramm handeln muss. Allerdings kann dahingehend argumentiert werden, dass durch die Verbreitung einer Schadsoftware der Entwickler derselben die Ansprüche an der exklusiven Verbreitung seiner Schöpfung aufgibt. Da sich Aufgrund der selbstpropagierenden Natur von Computerviren so gut wie jeder eine legitime Kopie des

Die Ansicht wird von der Kammer für Computerprogramme im Allgemeinen geteilt, ohne auf die Schutzfähigkeit von Webseiten näher einzugehen.
Die Sichtweise des OLG Hamburg wird im Rahmen dieses Urteils zwar erwähnt, jedoch wird die Gültigkeit des Urteils explizit als ebenfalls valide Sichtweise dargestellt.

Virus aneignen kann, ohne einer Lizenzvereinbarung zuzustimmen, darf auch jeder den Virus untersuchen. Oder anders ausgedrückt, weil ein Virus üblicherweise ohne allgemeine Geschäftsbedingungen daherkommt, kann man davon ausgehen, dass ein Entwickler keine Einwände gegen eine Untersuchung seines Werkes hat. In der Tat lasst sich eine Konstruktion der Einwilligung vollziehen.

Abschließend könnte man bei einem vorliegenden Virus ebenfalls mit §904 BGB beziehungsweise §34 StGB argumentieren, dass bei einer Abwägung der Rechtsgüter der Selbstschutz durch Analyse überwiegt.

5.2 Beauftragen eines Dritten mit der Untersuchung

Einer der ersten Schritte bei Sicherheitsanalysen ist meist die Beauftragung einer Drittfirma oder eines Forschers, da beide das nötige Fachwissen besitzt, um Sicherheitslücken zu erkennen. Allerdings ist dazu Zugang zur Software nötig, welcher meist durch Vervielfältigung und Weitergabe an die untersuchende Firma realisiert wird. Dies stellt allerdings nach §69c Abs. 1 UrhG eine genehmigungspflichtige Handlung dar, die der Nehmer in der Regel nicht tun darf. Zudem ist die kommerzielle Nutzung, die im Rahmen einer beauftragten Untersuchung stattfindet, von für Privatanwender bestimmter Software meist auch durch vertragliche Vereinbarungen untersagt. Dazu hat der I. Zivilsenat des Bundesgerichtshofs im Grundurteil vom 6.10.2016 - I ZR 25/15, bekanntgegeben: „[Der dazu berechtigte Lizenznehmer darf genehmigungspflichtige Handlungen nach §69c Abs. 1] auch dann ohne Zustimmung des Rechtsinhabers vornehmen, um das Funktionieren dieses Programms zu beobachten, zu untersuchen oder zu testen und die einem Programmelement zugrundeliegenden Ideen und Grundsatze zu ermitteln, wenn er dabei gewerbliche oder berufliche Zwecke verfolgt und der Lizenzvertrag lediglich eine Nutzung des Programms zu privaten Zwecken gestattet.”. Da zudem eine im Auftrag der Lizenznehmerfirma arbeitende Sicherheitsfirma oder Sicherheitsforscher eine handlungsbevollmächtigte Partei ist, darf, so der ursprüngliche Lizenznehmer dazu berechtigt ist, die beauftragte Firma die gleichen Handlungen vornehmen wie der Auftraggeber. Eine Außnahme bilden hierbei sogenannte „höchstpersönliche” Lizenzen, die explizit eine Person als Lizenznehmer bestimmen. In diesem Fall darf nur die im Vertrag benannte Person die Software verwenden und damit testen.

6 Fazit

Eine Sicherheitsanalyse ist prinzipiell erlaubt, wenngleich es viele verschiedene Arten gibt, auf die diese Analyse stattfinden kann. Zusammenfassend scheint in der Literatur der Konsens zu bestehen, dass alle Blackbox-Verfahren in Deutschland erlaubt sind, Whitebox-Verfahren bedürfen jedoch einer gesonderten Rechtfertigung. Dekompilierung ist im Rahmen einer Sicherheitsanalyse neben der technischen Problematik auch rechtlich

bedenklich, wohingegen die Dekompilierung von kleinsten Programmabschnitten geringer Komplexität im Sinne der kleinen Münze nach aktuellem Recht akzeptabel ist, muss bei der Dekompilierung ganzer Programme von Illegalität ausgegangen werden. Diese Grauzone ist aus Sicht der Sicherheitsforschung bedauerlich, da sich Software-Schwachstellen häufig gerade in komplexeren Code-Teilen verbergen. Ein Ausnahmetatbestand, sei es in Form einer gesetzlichen Schranke oder der Erweiterung des Testbegriffs aus §69d Abs. 3, ähnlich dem zur Herstellung von Interoperabilität wäre hier wünschenswert.

Abschließend ist zu sagen, dass das Urheberrecht bei weitem nicht das einzige Recht ist, das bei einer Sicherheitsanalyse beachtet werden muss. Vertragsrecht, Patentrecht, Gebrauchsmusterschutz und das Geschäftsgeheimnisgesetz sind oftmals ebenso wichtig wie das Urheberrecht. Dennoch stellt das deutsche Urheberrecht oftmals den ersten und oftmals einzigen Ansatzpunkt für Zivilklagen dar, da durch ungenaue Formulierungen sowie der Interpretationen mancher Autoren juristischer Fachliteratur, in beliebige Richtungen argumentiert werden kann. So hat selbst das BSI bereits Sicherheitsforschern mit einer Unterlassungsklage gedroht [Bö13]. In Abhängigkeit der Sichtweise der Kammer und der Qualität der Klage und Verteidigung kann keine verlässliche Voraussage über Prozesse getroffen werden.

Ferner kann auch die urheberrechtliche Schutzfähigkeit von Programmen in Frage gestellt werden, da Computercode oft nur Mittel zum Zweck ist, um Software zu verwirklichen. Oft sind Programme, die im professionellen Umfeld eingesetzt werden, auf Leistung optimiert und demzufolge die dahinterliegenden Entwicklungsentscheidungen technisch bedingt. Auf der anderen Seite wurde bis zum Urteil „Geburtstagszug" des BGH, 13.11.2013 - I ZR 143/12, unterschieden zwischen sogenannter angewandter Kunst und zweckfreier Kunst. Erstere hatte „[...] seit jeher höhere Anforderungen [an das Mindestmaßder Schöpfungshöhe] gestellt", wie der BGH in seinem bis dahin nicht aufgegebenen Urteil „Silberdistel", 22.06.1995 - I ZR 119/93, vermutet hat. Inwiefern sich ein doch so umfangreicher Schutz aufrecht erhalten lässt, kann jedoch im Rahmen dieser Arbeit nicht geklärt werden.

Die Leidtragenden dieser Unsicherheit sind die Sicherheitsforscher, die durch Klageandrohung und Prozesse unter Druck gesetzt werden können, da Urheberrechtsverletzungen mit hohen Geldstrafen oder Freiheitsstrafen von bis zu drei Jahren nach UrhG bedroht werden. Juristisch müsste Sicherheitsforschern heute empfohlen werden dem Grundsatz „In dubio mitius" (Im Zweifel für das Mildere) zu folgen und lieber nichts anstelle etwas falschem zu tun, also nach Schwachstellen gar nicht erst zu suchen. Unabhängige Sicherheitsanalysen aber erhöhen das Sicherheitsniveau unserer digitalen Infrastruktur und dienen damit dem Gemeinwohl. Das Recht und die Rechtsprechung sollte diesem Interesse Vorrang einräumen vor der Möglichkeit, die Aufdeckung von Sicherheitsschwachstellen in kommerzieller Software durch Anwendung des Urheberrechts zu unterdrücken.

Acknowledgements

Besonderer Dank gilt Prof. Franz Hofmann für seine Vorlesung über Grundlagenwissen zum geistigen Eigentum und seine Gesprächs- und Hilfsbereitschaft, sowie Dr. Till Jaeger für seine fachliche Beratung zu diesem Thema.

Literaturverzeichnis

[Bi14] Bisges, Marcel: Die Kleine Münze im Urheberrecht - Analyse des ökonomischen Aspekts des Werkbegriffs. Nomos, 2014.

[Bö13] Böck, Hanno: , GSTool: BSI bedroht Sicherheitsforscher, September 2013. https://www.golem.de/news/gstool-bsi-bedroht-sicherheitsforscher-1309-101531.html.

[Bu93] Bundestag: , Zweites Gesetz zur Änderung des Urheberrechtsgesetzes, June 1993. eingesehen unter http://www.urheberrecht.org/law/normen/urhg/1993-06-09/materialien/bgbl_I_910.php.

[BZ17] Bertoglio, Daniel Dalalana; Zorzo, Avelino Francisco: , Overview and open issues on penetration test, February 2017. https://journal-bcs.springeropen.com/articles/10.1186/s13173-017-0051-1.

[FMC12] Falliere, Nicolas; Murchu, Liam O; Chien, Eric: W32.Stuxnet Dossier. Technical report, Symantec, February 2012.

[Fo19] Foundation, Free Software: , GDB: The GNU Project Debugger, September 2019. https://www.gnu.org/software/gdb/.

[Ka18] Kaboth/Spies: , Beck Online Kommentar Urheberrecht, June 2018. BeckOK UrhR/Kaboth/Spies UrhG §69d Rn. 15-17.

[Ma18] Marly, Jochen: Praxishandbuch Softwarerecht, 7. Auflage. C.H. Beck, 2018.

[Pr06] Project, The Boomerang Decompiler: , Boomerang Decompiler, June 2006. http://boomerang.sourceforge.net/FAQ.php.

[Sc14] Schweyer, Florian: Die rechtliche Bewertung des Reverse Engineering in Deutschland und den USA. Mohr Siebeck, 2014.

[VP17] Vanhoef, Mathy; Piessens, Frank: Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2. In: Proceedings of the 24th ACM Conference on Computer and Communications Security (CCS). ACM, 2017.

[Wa19] Wandtke/Bullinger/Grützmacher: UrhG § 69a Rn. 38, 5. Aufl. C.H. Beck, 2019.

[WF12] Willems, Carsten; Freiling, Felix C.: Reverse Code Engineering - State of the Art and Countermeasures. it - Information Technology, 54:53–63, 2012.