DER NACHWEIS VON URHEBERRECHTSVERLETZUNGEN IN COMPUTERPROGRAMMEN

Michael Sonntag

Assoz.-Professor, Johannes Kepler Universität Linz, Institut für Netzwerke und Sicherheit
Altenbergerstr. 69, 4040 Linz, AT
michael.sonntag@ins.jku.at; https://ins.jku.at/

Schlagworte: ***Computerprogramme, Urheberrecht, Kopieren***

Abstract: *Übernimmt eine Firma zB Open-Source Quellcode in ihr Produkt und erfüllt dabei nicht die Bedingungen der entsprechenden Open-Source Lizenz, so liegt eine Urheberrechtsverletzung vor. Aus früheren Gerichtsentscheidungen ist bekannt, dass diese auch erfolgreich verfolgt werden können. Praktisch stellen sich dabei jedoch vielfach Probleme, insb wenn die Open-Source Software nicht identisch verteilt, sondern in ein eigenes Programm eingebaut wird. Der Grund hierfür ist, dass der Quellcode bei proprietärer Software fast nie zugänglich ist und in den Lizenzbedingungen jede Dekompilierung (und evtl weitere andere Analysemöglichkeiten) explizit verboten wird. Es stehen daher kaum legale Möglichkeiten zur Verfügung, eine Urheberrechtsverletzung vor einem Gerichtsverfahren festzustellen bzw nachzuweisen. Dieser Beitrag arbeiten heraus, welche Analysemethoden noch offenstehen, wie technisch erfolgversprechend diese sind, und wie es um deren rechtliche Zulässigkeit steht.*

1. Einleitung

Ein praktisches «Problem» der Open-Source Software ist, dass ihr Quellcode weit verbreitet und leicht zugänglich ist – eines der expliziten Ziele. Dies führt jedoch dazu, dass proprietäre Software jederzeit diesen Quellcode übernehmen und in das eigene Programm einbauen kann. Während dies je nach Lizenz legal sein kann, bzw unter bestimmten Zusatzbedingungen, ist dies bei der weit verbreiteten GPL-Lizenz nicht möglich. Wird entsprechender Quellcode eingebaut, muss das gesamte Programm unter die GPL gestellt werden. Was passiert aber, wenn das Unternehmen dies nicht macht? Dann liegt eine Urheberrechtsverletzung vor, sofern es sich bei der Open-Source Software um ein Werk handelt – was (abgesehen von winzigen Skripts oder kleinen Teilen) praktisch immer vorliegen wird. Eine Urheberrechtsverletzung ist jedoch sogar rechtlich völlig folgenlos, sofern sie nicht entdeckt und nachgewiesen werden kann. Denn nur dann ist (notfalls) ein Gerichtsverfahren möglich, bzw kann das Unternehmen, welches die proprietäre Software mit den Open-Source-Teilen darin erzeugt/lizenziert/vertreibt/…, zu Verhandlungen über korrekte Handlungsweisen «überredet» werden.

Allerdings legt das Urheberrechtsgesetz dem Entdecken bzw dem Nachweis von Verletzungen bei Computerprogrammen einige Steine in den Weg. Während es bei Software gleich wie bei Bildern, Büchern, Musikstücken etc trivial ist, eine vollständig identische Kopie nachzuweisen (zB unerlaubte Kopie des Installationsmediums), so kann eine Teil-Kopie, Neukompilierung illegal erhaltenen Quellcodes bzw eine Integration von Ausschnitten nur sehr schwer bewiesen werden. Wird zB ein Bild oder ein Ausschnitt desselben unerlaubt in ein Buch übernommen, so reicht es aus, ein Exemplar dieses Buches zu kaufen. Der Nachweis des Kopierens ist dann ohne Probleme durch Vergleich möglich. Gleiches gilt für Musikstücke (Tonaufnahme oder Noten) sowie zB Gebäude oder Statuen (Fotografie). Datenbanken sind schon etwas schwieriger, doch durch die Integration erfundener Datensätze oder bewusster minimaler Verfälschungen ist eine Kopie, bei entsprechender Vorsorge, ebenfalls leicht nachweisbar. Bei einem Computerprogramm erfolgt das hier relevante Kopieren

jedoch praktisch immer auf der Ebene des Quellcodes, für einen Verletzungsnachweis verfügbar ist jedoch ausschließlich der übersetzte Maschinencode (dh die direkt ausführbare Form), oder sogar nur die Ausgabe (zB bei Online-Diensten). Mit der Übersetzung liegt daher ein Zwischenschritt vor, der nicht ohne Probleme rückgängig gemacht werden kann, bzw darf, und einen einfachen Vergleich unmöglich macht[1].

1.1. Dekompilierungsverbot

In § 40e UrhG ist festgelegt, wann eine Dekompilierung, dh die Übersetzung eines Computerprogramms von der ausführbaren Form zurück in den Quellcode, erlaubt ist. Dies gilt ausschließlich (unter weiteren Bedingungen) für einen einzigen Zweck: um die erforderlichen Informationen zur Herstellung der Interoperabilität eines unabhängig geschaffenen Computerprogramms mit anderen Programmen zu erhalten.

Judikatur der obersten Gerichte zu diesem Paragraphen scheint in Österreich komplett zu fehlen: Im RIS konnte hierzu keine einzige Entscheidung gefunden werden[2]. Der EuGH hat hierzu einmal sehr indirekt Stellung genommen in der Entscheidung SAS (EuGH C-406/10 vom 2. Mai 2012; siehe auch Staudegger, JusIT 2012/45, 97), wobei festgestellt wurde, das im konkreten Fall gerade keine Dekompilierung erfolgte, sondern das Programm nur «von außen» untersucht wurde (sogenanntes «Black-Box Testing»).

Entscheidungen zum Dekompilieren aus Deutschland sind:

- OLG Frankfurt, 27. Januar 2015, 11 U 94/13: Eine Dekompilierung verletzt das Recht auf Bearbeitung (§ 69c Abs 1 Z 1 dUrhG). Der Datenschutz (Zweck: Entfernung des Google-Analytics-Tracking-Codes) rechtfertigt keine Dekompilierung. Allerdings war im konkreten Fall schon die Weitergabe der Software an neue Kunden widerrechtlich.
- LG Frankfurt 26. Juli 2006, 2–6 O 224/06: Es wurden Zeichenketten des eigenen Quellcodes im Maschinencode des Verletzers gefunden. Dies war ohne Urheberrechtsverletzung feststellbar, sodass es zu keinem Beweisverwertungsverbot kam. Konkret wurde die Firmware von der Webseite heruntergeladen, entpackt und dann die Strings extrahiert. Dies stellt kein unzulässiges Dekompilieren dar.

2. Relevante aktuelle Urteile

2.1. Aktueller Fall: LG Hamburg, 8. Juli 2016, 310 O 89/1

Dies betrifft eine Entscheidung des LG Hamburg, welche derzeit am OLG anhängig ist: Der Kläger bot uA folgende Beweismittel an:

- Im Source-Code-Verwaltungs-System von Linux seien seine Beiträge detailliert nachvollziehbar. Weiters legte er den gesamten Quellcode des Linux-Kernels vor, wobei in einer weiteren Datei angegeben sei, welche Änderungen daran von ihm stammten.
- In zwei von der Beklagten offengelegten Header-Dateien fänden sich Hinweise auf sein Urheberrecht.
- Der Verletzer verwendet die Funktionalität «SCSI-Hotplug», welche im Linux-Kernel von ihm (mit-) implementiert wurde.

Als problematisch sah das Gericht hierbei an, dass der Kläger nicht auf konkrete Codezeilen hinwies, welche von ihm stammten und urheberrechtlichen Schutz genießen sollten. Weitere Teile des Quellcodes stammten

[1] Hintergrund: Ein und derselbe Quellcode kann durch verschiedene Compiler, mit unterschiedlichen Einstellungen, Optimierungen etc in eine enorm große Anzahl unterschiedlicher Programme übersetzt werden. Funktionell sind diese zwar identisch (= gleiche Ausgabe bei gleicher Eingabe), doch ein direkter Vergleich ist nicht mehr möglich.

[2] Keine Entscheidung zu §40e UrhG. Zu «dekompilier» werden nur zwei relevante Ergebnisse geliefert: Ein Bescheid des Bundesvergabeamtes (3. Januar 2011, N/0075-BVA/14/2010-56; Eine Dekompilierungsmöglichkeit reicht nicht aus, um Software zu warten/weiterzuentwickeln) sowie eine Empfehlung der Datenschutzkommission (28. November 2003, K211.507/024-DSK/2003; Kryptographische Schlüssel zur Verschlüsselung einer Datenbank können per Dekompilierung ausgelesen werden. Dies ist jedoch nur Spezialisten möglich, sodass eine Sicherung dagegen nicht erforderlich ist).

nicht von ihm, sondern wurden von ihm nur an andere Stellen verschoben. Das Gericht verwies zudem darauf, dass der Quellcode des angeblich verletzenden Programms verfügbar ist, sodass der Kläger genau bezeichnen könnte, welcher von ihm stammende Code wo eingebaut wurde, dies jedoch nicht tat. Es wurde zwar ein Vergleich vorgelegt (identisch, hinzugefügt, sowie gelöscht), doch betrifft dieser nur pauschal ganze Dateien und gibt wiederum nicht an, welche Teile davon vom Kläger stammten (dies wäre zwar aus anderen Dateien feststellbar, doch ist dies nicht hinreichend genau). Dass der angebliche Verletzer die identische Funktionalität implementiert, sei noch kein Hinweis auf eine Urheberrechtsverletzung. Bezüglich der Header-Dateien vermisste das Gericht einen Vortrag darüber, inwieweit dadurch urheberrechtlich geschützter Code des Klägers übernommen worden sei. Hintergrund für letzteres dürfte sein, dass Header iA keinen Programmcode enthalten sondern nur Schnittstellen beschreiben (deren urheberrechtliche Schutzfähigkeit umstritten bzw jedenfalls stark eingeschränkt ist). Die Kopie eines Headers bzw von Schnittstellendefinitionen beweist weiters nicht, dass die Implementierung derselben kopiert wurde – sie könnte auch vollständig selbst erfolgt sein. Die Angebote, weitere Entwickler anzuhören, wer wovon Urheber sei, oder ein Sachverständigengutachten einzuholen, wurden als Ausforschungsbeweise abgelehnt.

Nach dieser Entscheidung sind daher folgende Elemente für den Nachweis einer Urheberrechtsverletzung an einem Computerprogramm notwendig:

- Es ist detailliert (Datei+Zeilenangabe; Version) anzuführen, welcher Code vom Kläger stammt, indem sein eigener Quellcode präsentiert wird.
- Es ist auszuführen, warum dieser Code urheberrechtlich geschützt ist (= ein Werk ist). Dies ist relevant, wenn es sich nur um kleine Teile oder lediglich um Bearbeitungen von Werken Dritter handelt und nicht ohnehin offensichtlich ein Computerprogrammwerk vorliegt.
- Es ist anzugeben, in welches Programm welche dieser Zeilen übernommen wurden, und warum dies vermutet wird (bzw wo genau mit Datei+Zeilenangabe und Version, sofern Quellcode vorhanden ist).

Die vom Gericht gestellten Anforderungen sind hoch, aber nicht ungewöhnlich: Auch bei Verletzung von Urheberrechten an Texten oder Bildern ist genau anzugeben, welche Textstellen bzw welche Bilder wo «wiederverwendet» wurden. Ein pauschaler Hinweis «Teile dieses Buches wurden in das Buch des Beklagten übernommen» reicht nicht aus. Problematisch ist hier wiederum der letzte Punkt: Wenn der Quellcode des angeblich verletzenden Produktes nicht verfügbar ist, wie soll dann nachgewiesen, oder auch nur wahrscheinlich gemacht werden, dass eine Urheberrechtsverletzung tatsächlich vorliegt? Der Beklagte könnte einfach anführen «Wir haben nicht kopiert. Beweisen Sie etwas anderes (aber ohne unseren Quellcode, und ohne unser Programm zu dekompilieren!).»

2.2. EuGH 18. Oktober 2018, C-149/17 [Hörbuch]

Die RZ 41 dieser Entscheidung lautet: «Daher ist Art. 6 Abs. 1 der Richtlinie 2004/48 in Verbindung mit ihrem 20. Erwägungsgrund dahin auszulegen, dass die Mitgliedstaaten es dem Geschädigten tatsächlich ermöglichen müssen, die zur Begründung seiner Ansprüche erforderlichen Beweismittel zu erlangen, die sich in der Verfügungsgewalt der gegnerischen Partei befinden, sofern bei der Vorlage dieser Beweismittel der Schutz vertraulicher Informationen gewährleistet wird.» Da die RL 2004/48[3] zur Durchsetzung der Rechte geistigen Eigentums dienen soll, gilt sie (neben im gegenständlichen Fall Hörbüchern) auch für Computerprogramme (siehe Art 2 Abs 2). Dieses Recht ist zwar mit dem Recht auf Achtung des Privat- und Familienlebens abzuwägen, darf aber nicht zu einer vollständigen Verhinderung jeden Nachweises führen.

Daraus kann mE nach nicht argumentiert werden, dass Dekompilierung zum Verletzungsnachweis zulässig wäre, da dies ein schwerer Eingriff ist, und die Informationen zB dann auch zu anderen Zwecken verwendet

[3] Richtlinie 2004/48/EG des Europäischen Parlaments und des Rates vom 29. April 2004 zur Durchsetzung der Rechte des geistigen Eigentums, ABl L 195/16 vom 2. Juli 2004 (Berichtigung).

werden könnten. Gleichwohl kann dies jedoch als Grundlage dafür dienen, dass zB Vervielfältigungen des Programms zur Untersuchung vorgenommen werden dürfen. Beispielsweise kann die Lizenz eines Programms nur die Ausführung zur tatsächlichen Nutzung erlauben. Ein Laden in den Speicher (ohne Ausführung des Programms!) ist jedoch bereits eine Vervielfältigung (siehe unten) und von diesem Zweck (bzw dem UrhG) uU nicht gedeckt. Dies jedoch zu verbieten würde es vollkommen unmöglich machen, jemals irgendeine Art von Beweis zu erlangen, sodass dies ein Widerspruch zu diesem Urteil wäre.

Gleichzeitig lässt sich daraus (bzw direkt aus RL 2004/48 Art 6) ablesen, dass eine Untersuchung durch Dritte (= Gewährleistung des Schutzes vertraulicher Informationen) nicht folgenlos verweigert werden kann. Daraus ist weiters zu schließen, dass alles (jedoch nur das absolute Minimum) was nötig ist, um dies vor Gericht erreichen zu können, zulässig sein muss, denn ansonsten würde dieses explizit vorgesehene Recht leerlaufen.

2.3. BGH, 6. Oktober 2016, I ZR 25/15 [World of Warcraft]

Diese Entscheidung stellt für die Suche nach Urheberrechtsverletzungen ein weiteres Problem dar, da in ihr die Ausnahme des §69d Abs 3 dUrhG (Black-Box Testen) ausschließlich auf Computerprogramme bezogen wird, nicht aber auf in diesem enthaltene andere Werke. Da jedes Laden in den Speicher (unabhängig vom Ausführen des Programms) eine Vervielfältigung darstellt, ist daher evtl jedwede Untersuchung eines Computerprogramms widerrechtlich, sofern dieses ein weiteres Werk enthält. Denn für ein solches Werk kann uU nicht einmal die Argumentation aus dem vorigen Abschnitt gelten, sofern es nicht ebenfalls unter dem Verdacht einer Urheberrechtsverletzung steht. Der einzige Ausweg ist, dass alle darin enthaltene Werke illegal wären. Dies wird nur bei vollständigen 1:1 Kopien vorliegen (dann ist jedoch eine detaillierte Untersuchung unnötig), aber nicht bei der Übernahme bloßer Teile.

3. Nachweismöglichkeiten

Aus technischer Sicht stehen folgende Möglichkeiten zur Verfügung: Dekompilierung, Extraktion von Zeichenketten, Binärvergleich, sowie statische bzw dynamische Aufrufstruktur.

3.1. Extraktion von Zeichenketten

Zeichenketten können aus ausführbaren Dateien extrahiert werden, da diese neben dem eigentlichen Programmcode auch die zugehörigen Daten enthalten. Eine mögliche Gegenmaßnahme dagegen ist, die ausführbare Datei zu komprimieren. Dann ist lediglich ein winziger Programmteil zur Dekomprimierung direkt sichtbar (der keine Zeichenketten enthält), welcher beim Programmstart den eigentlichen Programmcode zusammen mit den Daten entpackt. In diesem Fall müsste das Programm ausgeführt werden, um anschließend den Speicher des Programms zu kopieren und auszuwerten. Dies stellt keine Dekompilierung dar, sondern nur eine (weitere) Vervielfältigung und erschwert die Analyse daher nur, aber verhindert sie nicht. Da eine Datenkomprimierung nur der Effizienz dient, wird hiermit auch keine Schutzmaßnahme umgangen. Allerdings kann in die Komprimierung auch ein Passwort eingebaut werden, was zu einer anderen Beurteilung führen könnte. Informationen zu sammeln um ein eigenes Extraktions-Programm zu schreiben ist jedoch von der Dekompilierungs-Ausnahme gedeckt.

Bei der Extraktion der Zeichenketten wird uA klar, welche Betriebssystem-Funktionen (zB «Öffnen einer Datei») aufgerufen werden, da deren Namen enthalten sind. Dies ist jedoch funktionsbezogen (was macht das Programm), und ist daher zum Nachweis von Urheberrechtsverletzung ungeeignet. Denn ein komplett anderer Programmcode würde immer noch die gleichen Funktionen verwenden (müssen).

Allerdings können Zeichenketten auch sehr individuell für die Software sein, zB Copyright-Strings (Beispiel: «(c) Michael Sonntag, 2018»). Diese werden oft in einer «About»-Funktion des Programms ausgegeben. Wird daher der gesamte Quellcode kopiert, so ist auch diese Funktion dabei. Problematisch hierbei ist jedoch, dass Compiler/Linker nicht verwendete Funktionen (sowie ausschließlich die zu diesen Funktionen gehörende Daten) entfernen. Derartige Zeichenketten bleiben daher nur dann erhalten, wenn die Funktion zumindest auf

irgendeine Weise weiterhin ausgeführt werden könnte. Wird also nur der «nützliche» Programmcode übernommen, so fallen derartige Zeichenketten bei der Übersetzung weg – sofern sie nicht manuell entfernt wurden.

Erfolgversprechender sind daher zB Debug-Zeichenketten, Muster für Logeinträge oder sonstige Ausgaben des Programms. Diese werden für die Funktionalität benötigt und müssen daher enthalten bleiben. Nachteilig ist bei ihnen, dass sie stark auf die Aufgabe des Programms zurückgehen, sodass eine individuelle Prägung (oder sonstige Einmaligkeit) weniger wahrscheinlich ist. Dennoch sind sie als Nachweis gut geeignet, denn eine einzelne Zeichenkette mag zufällig gleich formuliert werden, Duzende oder gar Hunderte jedoch kaum. Weitere Anhaltspunkte können Fehler in den Texten sein: Tippfehler, grammatikalische Fehler, inhaltliche Unkorrektheiten etc. Für die Beurteilung ist wichtig, dass die Zeichenketten nicht unbedingt selbst für sich alleine urheberrechtlich geschützt sein müssen. Denn wurden diese unverändert übernommen, so ist es auch höchst wahrscheinlich, dass der restliche Quellcode ebenso übernommen wurde – und dieser ist als Ganzes praktisch immer geschützt.

Dies wurde auch im Urteil LG Frankfurt 26. Juli 2006, 2-6 O 224/06 bestätigt: «[Die Kägerin] hat lediglich geltend gemacht, dass der Kläger nicht ermittelt habe, dass der Code tatsächlich übereinstimme, sondern lediglich, dass der Kläger Anhaltspunkte hierfür ermittelt habe. Entgegen dieser Auffassung hat der Kläger durch diesen Vortrag aber die Urheberrechtsverletzung hinreichend konkret vorgetragen. Generell kann aus dem Umstand, dass Zeichenketten vorhanden sind, die mit an Sicherheit grenzender Wahrscheinlichkeit dem Quellcode entstammen, der Schluss gezogen werden, dass der Quellcode verwandt wurde. Wie sonst die von dem Kläger im Einzelnen aufgezählten Zeichenketten zu erklären sind, die unstreitig auf die Verwendung der Quellcodes hinweisen, wurde von der Beklagten nicht erläutert.»

Urheberrechtlich wird bei der Extraktion von Zeichenketten nichts bearbeitet und nicht dekompiliert, es wird jedoch vervielfältigt, da die ausführbare Datei in den Speicher geladen und durchsucht werden muss (= vollständige Kopie), sowie die gefundenen Teile ausgegeben/gespeichert/etc werden (minimale Kopie, evtl nicht einmal urheberrechtlich relevant). Da es sich hierbei nicht um eine flüchtige und vorübergehende Vervielfältigung handelt (die dort erforderliche Zweckeinschränkung wird nicht erfüllt) ist es daher notwendig, dass dies durch einen Berechtigten und an einem legalen Vervielfältigungsstück erfolgt. Problematisch könnte sein, dass die Extraktion der Zeichenketten kaum im Lizenzvertrag vorgesehen ist und auch nicht zur «bestimmungsgemäßen Benutzung» des § 40d Abs 2 UrhG gehört. Hierbei wird auch nicht das Funktionieren des Programms (§ 40d Abs 3 Z 2 UrhG) untersucht. Ebenso wenig handelt es sich um eine Privatkopie, da es sich wohl meist auch beim angeblich in seinen Rechten Verletzten um eine gewerbliche Tätigkeit handeln wird (und § 42 für Computerprogramme ohnehin ausgeschlossen ist). Entsprechend dem EuGH Urteil «Hörbuch» (siehe oben) ist dies jedoch mM nach trotzdem zulässig.

3.2. Binärvergleich

Ein Vergleich von Quellcode kann indirekt durch den Vergleich des daraus erzeugten Binärcodes durchgeführt werden. Das Problem ist, dass Quellcode in eine Vielzahl an Binärcodes übersetzt werden kann, je nach Version des Übersetzungsprogramms (Compiler), verwendeter Optionen, Optimierungen etc. Der Vorteil dieses Ansatzes ist, dass der eigene Quellcode beliebig verarbeitet werden darf und mit dem vermutlich verletzenden Code lediglich in Binärform verglichen wird - es erfolgt keine Dekompilierung sondern nur minimale Vervielfältigung.

Die Herangehensweise hierbei ist wie folgt: Der eigene Quellcode wird in eine enorme Vielzahl an Versionen übersetzt (mehrere Compiler, Kombinationen von Einstellungen davon etc, verschiedene Versionen externer Bibliotheken – und alle möglichen Kombinationen dieser Elemente!). Anschließend werden die allgemeinen Teile (für das Betriebssystem zum Laden des Programms, Daten, externe Bibliotheken etc) entfernt. Das Ergebnis kann dann binär mit dem fremden Programm verglichen werden. Wurde eine identische Übereinstimmung

gefunden, so ist exakt der gleiche Quellcode enthalten. Denn dass ein anderer Quellcode den absolut identischen Maschinencode erzeugt, ist mit an Sicherheit grenzender Wahrscheinlichkeit auszuschließen. Allerdings ist dies eine sehr aufwendige Methode und kann keine Garantie geben: Schon kleinste Änderungen des Quellcodes beim Kopieren führen dazu, dass eine identische Übereinstimmung nicht mehr gegeben ist; aufgrund von Optimierungen kann der Maschinencode sogar praktisch komplett anders sein.

Ein weiteres technisches Problem ist, dass eine zusätzliche «Maskierung» erforderlich ist, wenn kein Positionsunabhängiger Programmcode erzeugt wird (ob dies so ist, richtet sich nach dem vermutlich verletzenden Programm, kann bei der Überprüfung also nicht selbst entschieden werden). Wird zB eine Funktion vom Anfang des Programms ans Ende verschoben, so wandern auch die (absoluten) Ziele von Sprüngen nach hinten («Springe zu Zeile 12» → «Springe zu Zeile 217») – und ändern damit den Binärcode (die Zahl 12 wird zu 217). Positions-unabhängiger Code hingegen enthält nur Anweisungen der Art «Überspringe 5 Zeilen», was unabhängig von der Position im Programm ist. Eine solche Maskierung müsste sehr aufwändig händisch für jede Variante erfolgen, da kein Programm dafür bekannt ist (aber zumindest theoretisch möglich wäre).

Weiters ist zu berücksichtigen, dass ein größerer identischer Bereich zu finden ist: Ansonsten kann nur festgestellt werden, dass «eine einzelne Zeile des Quellcodes» (die Dutzenden Maschinencode-Befehlen entsprechen kann) identisch übernommen wurde. Dies ist jedoch urheberrechtlich meist irrelevant. Dh nur wenn zumindest ein Teilwerk gefunden wurde, kann eine Verletzung nachgewiesen werden.

Im Ergebnis ist dies daher zwar eine Möglichkeit, aber mit sehr vielen Unwägbarkeiten verbunden und trotz hohem Aufwand ist es uU unmöglich eine tatsächlich stattgefundene Urheberrechtsverletzung nachzuweisen. Praktisch scheint diese Methode daher nach Kenntnis des Autors derzeit nicht eingesetzt zu werden.

3.3. Dekompilierung

Dieser oft verwendete Ansatz erfordert, einen praktischen Aspekt zu berücksichtigen: Eine Dekompilierung kann «im Geheimen» immer erfolgen, um eine Urheberrechtsverletzung zu erkennen. Denn sie kann ohne Unterstützung von außen auf einem einzelnen Computer (wenn auch sinnvoll nur durch Experten) erfolgen. Es bleiben kaum, außer auf diesem einen Computer und dort nur wenige, Spuren zurück, sodass ein Nachweis in der Praxis fast unmöglich ist. Im Endeffekt handelt es sich daher um eine Art Beweisverwertungsverbot: Eine Rechtsverletzung konnte (technisch) nachgewiesen werden, aber die gewonnenen Beweise hierfür sind nicht zulässig (oder besser: sind selbst nur durch Rechtsverletzung entstanden und führen daher uU zu einer Gegenklage).

Die Praxis sieht daher oft so aus: Durch Dekompilierung wird festgestellt, dass eine Verletzung vorliegt. Anschließend wird versucht, mittels anderer Wege (siehe die anderen Unterabschnitte) genügend Hinweise zu sammeln, mit denen ein Richter überzeugt werden kann, dass ausreichender Verdacht vorliegt. Dies kann schwierig sein (siehe das Beispiel aus Abschnitt 2.1; ob dort vorher dekompiliert wurde, ist jedoch unbekannt). Anschließend kann dann ein Dritter, zB ein Gerichts-Sachverständiger, beauftragt werden, die Quellcodes der beiden Programme zu vergleichen. Weigert sich eine Partei, diesen an den Gutachter herauszugeben, so kann dies als Beweisvereitelung angesehen werden, da kein vernünftiger Grund hierfür besteht, sofern es sich um einen echten Dritten handelt der zu Verschwiegenheit verpflichtet ist.

Praktische Probleme bei der Dekompilierung sind, dass diese zwar technisch vielfach einfach ist, die notwendige Interpretation durch einen Menschen jedoch enorm aufwendig (finden des gleichen bzw entsprechenden Programmcodes). Leichter ist dies in Spezialfällen, zB wenn eine Rückübersetzung nicht nur nach Assembler erfolgt, sondern zu einer höheren Programmiersprache, zB C oder Java. Dies ist nur möglich, wenn die Programmiersprache des Programms bekannt ist (trivial), entsprechende Software zur Verfügung steht (nur für wenige, aber dafür die am weitesten verbreiteten Sprachen gegeben), sowie keine absichtliche Erschwerung durch Verschleierung («Obfuskation») vorliegt. Letztere kann für sich kein Hinweis auf eine Urheberrechtsverletzung sein, da es auch andere Gründe hierfür gibt, zB Erschwerung (legaler) Analyse, Sicherheit (Erzeugung

von Variationen um das Ausnützen von Fehlern zu erschweren) oder Nachverfolgbarkeit (erzeugen individueller Versionen für jeden einzelnen Kunden).

Rechtlich ist Dekompilierung für diesen Zweck auf eine Zustimmung des Inhabers der Nutzungsrechte an dem Programm gebunden, da § 40e UrhG dies verbietet. Da dies eine explizite bewusste Regelung ist, kann hier mM nach nicht mit EuGH-Urteil «Hörbuch» argumentiert werden.

3.4. Aufrufstruktur

Bei der Aufrufstruktur handelt es sich darum, in welcher Reihenfolge (oder Art) Programmteile aufeinander zugreifen. Dies kann einerseits statisch analysiert werden (wo befinden sich Schleifen, welche Funktion ruft welche andere auf), andererseits dynamisch zur Laufzeit (welche Ablauffolge ergibt sich bei bestimmter Eingabe). Im ersteren Fall ist es erforderlich, den Programmcode zu dekompilieren, daher stellt dies nur eine Variante des vorigen Abschnitts dar (einfachere Analyse). Im zweiten Fall ist entweder ebenfalls Dekompilierung nötig, oder eine Instrumentation/Debuggen des Programms. Hierbei ist es wiederum erforderlich, zumindest einzelne Anweisungen zu dekompilieren (ist der nächste Schritt ein Sprungbefehl oder nicht), sodass daraus das gleiche folgt. Vorteilhaft ist, dass kein Mensch den dekompilierten Code zu Gesicht bekommen muss, da diese Analyse vollautomatisch möglich ist. Eine Geheimhaltung von Interna ist daher besser gewährleistet. Rechtlich ist dies jedoch kein Unterschied, da eine Dekompilierung (und Vervielfältigung) dennoch erfolgt. Allenfalls kann man argumentieren, dass das Programm zu dem vorgesehenen Zweck ausgeführt wird (wenn auch mit zusätzlicher Beobachtung), sowie dass die minimalen Teile die jeweils dekompiliert werden, keine Werke darstellen. Da jedoch in Summe schrittweise wieder ein Großteil des Programms dekompiliert wird, ist letztere Argumentation kaum haltbar.

Während ein Vergleich zweier Programme über die Aufrufstruktur deutlich einfacher ist muss beachtet werden, dass dies nur bedingt auf identischen Programmcode hinweist. Denn es wird nur der Algorithmus sowie die Gliederung und Aufteilung des Codes dargestellt, sodass sich auch nur diese Aspekte vergleichen lassen. Und diese sind urheberechtlich als bloße Konzepte nicht geschützt. Dies darf nicht darüber hinwegtäuschen, dass eine identische Struktur ein sehr starker Anhaltspunkt dafür ist, dass auch der Quellcode identisch ist. Problematisch ist mehr der Fall, dass sich die Strukturen nur stark ähneln: Dann ist unklar, in welchem Maße Quellcode übernommen wurde oder zufällig eine ähnliche Struktur gewählt wurde. Umgekehrt ist hilfreich, dass die Aufrufstruktur sehr unabhängig von Compiler und Optionen (ausgenommen zB Optimierungen) ist, und dadurch entsprechende Probleme des Binärvergleichs vermieden werden.

Dieser Ansatz erleichtert daher den Vergleich von Programmen ohne Quellcode, kann aber nur eingeschränkt Hinweise auf eine Übernahme von Quellcode geben und ist rechtlich eine Variante der Dekompilierung.

4. Zusammenfassung

Rechtlich gesehen ist es aufgrund der bisherigen Rechtsprechung bei strenger Auslegung praktisch unmöglich, eine Urheberrechtsverletzung an einem Computerprogramm auf rechtmäßige Art nachzuweisen. Denn schon das bloße Laden des Programms in den Speicher stellt eine Vervielfältigung des Computerprogramms (sowie aller sonstigen darin enthaltenen Werke!) dar. Hierfür besteht jedoch weder eine gesetzliche Ausnahme noch regelmäßig eine vertragliche Erlaubnis. Das Urteil des EuGH zu Hörbüchern ist daher in dieser Hinsicht besonders zu begrüßen und ist auch auf Computerprogramme anzuwenden, inklusive aller in diesen enthaltenen weiteren Werke. Ansonsten verhindert das Urheberrecht selbst jeden Nachweis von Verletzungen an durch ihm geschützten Werken einer Kategorie – im Gegensatz zu anderen Arten von Werken.

Hinsichtlich des praktischen Vorgehens ist daher eine mehrstufige Vorgehensweise empfehlenswert:

1. Untersuchung «von außen»: Ergeben sich aus den Dateinamen (zB Bibliotheken), angezeigten Texten (Benutzeroberfläche, Log-Dateien, Ausgabedaten) etc Hinweise auf eine Übernahme von Quellcode?

2. Analyse von Strings: Extraktion aller Zeichenketten aus den Dateien und Vergleich mit Zeichenketten des eigenen Quellcodes. Bei entsprechenden Anhaltspunkten sind die erforderlichen Vervielfältigungen legal.
3. Aufnahme von Verhandlungen bzw Einleitung eines Gerichtsverfahrens.
4. Vergleich des Quellcodes durch Dritte unter Geheimhaltung Nicht-übernommener Teile.

Wie man sieht, hat die zuverlässigste Form des Nachweises, die Dekompilierung hier leider keinen Platz: Wenn sie hilfreich wäre (vor einem Verfahren) ist sie verboten, und während eines Verfahrens steht ohnehin der Quellcode zur Verfügung. Dieses Ergebnis ist nicht optimal und es ist davon auszugehen, dass im Geheimen Dekompilierung sehr wohl stattfindet, aber für den «offiziellen» Teil man dann darauf angewiesen ist, andere Ansätze als Begründung heranzuziehen. Bei interaktiven Programmen (= viele Zeichenketten) ist dies möglich, doch zB bei Steuerungssystemen, die fast gar keine für Menschen gedachte Ausgabe besitzen, kann dies uU unmöglich sein, sodass dort eine echte Rechtsschutzlücke entsteht. Es sollte daher angedacht werden, eine entsprechende Ausnahme im Gesetz zu verankern. Das Argument, dass dann die Übernahme von Geschäftsgeheimnissen trivial ist, kann nicht gelten, da verbotene Dekompilierung auch jetzt problemlos im Geheimen stattfinden kann. Es ist sogar noch eher ein Nachweis möglich, da dann das Nachbau-Programm ebenfalls analysiert werden darf.