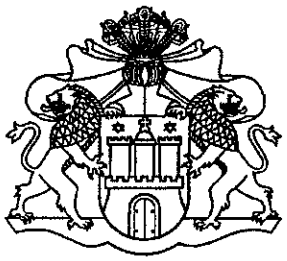
Abschrift

Landgericht Hamburg
Az.: 310 O 89/15

Verkündet am 08.07.2016

Heinelt, JAng
Urkundsbeamtin der Geschäftsstelle



Urteil

IM NAMEN DES VOLKES

In der Sache

Christoph Hellwig, Schidlachstraße 11, 6020 Innsbruck, Österreich

- Kläger -

Prozessbevollmächtigte:
Rechtsanwälte **Jaschinski, Biere, Brexl**, Christinenstraße 18/19, 10119 Berlin, Gz.: 14-1640

gegen

VMware Global, Inc., Zweigniederlassung Deutschland, vertr. durch Patrick Kevan Krysler, Freisinger Straße 3, 85716 Unterschleißheim

- Beklagte -

Prozessbevollmächtigte:
Rechtsanwälte **Freshfields, Bruckhaus, Deringer**, Im Zollhafen 24, 50678 Köln, Gz.: DAC17877145

erkennt das Landgericht Hamburg - Zivilkammer 10 - durch den Vorsitzenden Richter am Landgericht Hartmann, den Richter am Landgericht Harders und die Richterin Dr. Frantzen auf Grund der mündlichen Verhandlung vom 25.02.2016 für Recht:

1.

Die Klage wird abgewiesen.

2.

Der Kläger trägt die Kosten des Rechtsstreits.

3.

Das Urteil ist vorläufig vollstreckbar gegen Sicherheitsleistung in Höhe von 110% des jeweils zu vollstreckenden Betrages.

T a t b e s t a n d

Der Kläger ist Software-Entwickler. Er beansprucht, am sog. Kernel des bekannten Linux-Betriebssystems mitgearbeitet und auf diese Weise Bearbeiterurheberrechte erworben zu haben. Die Beklagte habe für ein eigenes, zum Download angebotenes Software-Produkt Teile des Linux-Codes des Klägers verwendet. Sie könne sich dafür nicht auf die Open-Source-Lizenz, unter der Linux veröffentlicht sei, berufen, da sie deren Lizenzbedingungen nicht eingehalten habe, deshalb die auflösend bedingte Lizenzierung entfallen sei, somit für die Beklagte derzeit kein Nutzungsrecht mehr bzgl. der Linux-Code-Anteile bestehe und der Kläger als Mitbearbeiter von Linux der Beklagten die Nutzung ihrer unter Verwendung von Linux entwickelten Software untersagen könne.

Im Einzelnen:

Nach dem Vortrag der Parteien im vorliegenden Verfahren ist das **Linux-Betriebssystem** lange vor dem Jahr 2000 in seiner Ursprungsversion von dem Programmierer Linus Torvalds initiiert und seither von einer Vielzahl von Programmierern unabhängig voneinander weiterentwickelt worden. Seit 2006 wird das **Versionskontrollsystem „git"** für das Linux-System eingesetzt, über welches nachvollziehbar ist, welche Beiträge von welchem Programmierer beigesteuert worden sind (vgl. Auszug Website Anlage K 1).

Dabei ist die Software Linux fortwährend als sog. Freie Software lizenziert worden und zwar unter den Bedingungen der **GNU General Public License, Version 2, auch GPL-2.0** (Text Anlage K 6). Grundgedanke dieser Lizenzierung ist, dass jeder Softwareentwickler, der einen Beitrag zu Linux leistet, es generell jedermann gestattet, Linux einschließlich der jeweils beigesteuerten Beiträge nicht nur zu nutzen, sondern auch weiterzuentwickeln, jedoch unter der auflösenden Bedingung, dass auch die Weiterentwicklung wiederum unter der GPL lizenziert wird, d.h. jedermann zur freien Nutzung und Weiterentwicklung zur Verfügung gestellt wird. Die Nutzer sollen also umfassende Nutzungsrechte dafür erhalten, dass sie im Gegenzug selbst auch allen anderen Nutzern wieder umfassende Nutzungsrechte einräumen, wenn sie die Software weitergeben. Dazu gehört u.a. die Offenlegung des Quellcodes der veränderten Version.

Der **Kläger** ist selbstständiger Software-Entwickler und arbeitet an einer Vielzahl von Open Source Software-Projekten mit, so auch seit dem Jahre 2000 an der Fortentwicklung von Linux. Welche Anteile er genau an Linux bearbeitet hat, ist streitig. Nicht bestritten ist, dass der Kläger, soweit er Bearbeitungen der Linux-Software vorgenommen hat, diese nach dem vorstehenden Gedanken unter der GPL-2.0 lizenziert hat.

Der Kläger macht im vorliegenden Rechtsstreit **Rechte am sog. Kernel des Linux-Betriebssystems** auch und gerade im Zusammenhang mit dem System zur Ansprache externer Geräte über sog. Gerätetreiber geltend. Zu den generellen Begrifflichkeiten „Betriebssystem", „Gerätetreiber" und

„Schnittstelle" – unabhängig von den vorliegend streitgegenständlichen Programmen - ist dabei unstreitig:

- Unter dem sog. **Kernel eines Betriebssystems** versteht man nach Darstellung der Parteien im vorliegenden Rechtsstreit den Kern eines Betriebssystems, wobei dieser Kern für die Verwaltung und das Ansprechen von Geräten zur Datenspeicherung (Festplatte, optische Laufwerke, Flash-Speicher) sowie für die Verwaltung von Gerätetreibern zuständig ist (Kläger, Klageschrift S. 8 und 9) bzw. der Kernel neben anderen Aufsichtsfunktionen die Aufgabe hat, die Input/Output-Anfragen anderer Softwarekomponenten und Anwendungen zu verwalten, zu überprüfen, zu organisieren und zu priorisieren (Beklagte Ss 01.07.2015 S. 6 = Bl. 68 d.A.).

- Zur Kommunikation eines Betriebssystem-Kernels mit einem an den Computer angeschlossenen externen Gerät ist ein sog. **Gerätetreiber** erforderlich ist. Die Beklagte hat hierzu unbestritten vorgetragen, dass grundsätzlich ein Gerätetreiber eine Software ist, die einen besonderen Typ von Geräten, welche mit dem Computer verbunden sind, bedient oder kontrolliert, indem sie die Anfragen von Anwendungen und vom Betriebssystem für die besonderen Anforderungen des jeweiligen Typs, der Marke und des Modells des Gerätes übersetzt. Möchte eine Software mit einem Gerät kommunizieren, muss der Treiber einen Befehl oder eine Serie von Befehlen an das Gerät erteilen (Ss 01.07.2015 S. 7 = Bl. 61 d.A.).

- Das Zusammenwirken zwischen Kernel und Gerätetreibern kann durch einen unterschiedlichen Aufbau des Betriebssystems erreicht werden. Der Kläger hat dazu unbestritten ausgeführt, dass beim Aufbau eines Betriebssystems die Gerätetreiber entweder mit dem übrigen Kernel des Betriebssystems zu einer einheitlichen Binärdatei verbunden werden können oder als sog. **Kernel-Module** ausgelagert werden können und dann vom System dynamisch nachgeladen werden müssen (Klageschrift S. 9).

- Als **Schnittstelle** bezeichnet man eine Komponente, die die Kommunikation anderer Komponenten ermöglicht; im Hardware-Bereich mag das eine bestimmte Steckverbindung sein, im Softwarebereich mögen dies bestimmte Programmierungsanforderungen sein. Als eine **abstrakte oder stabile Schnittstelle** kann dabei eine solche bezeichnet werden, deren Konfiguration bzw. Anforderungen sich auch bei einer Weiterentwicklung der kommunizierenden Komponenten nicht ändert; so stellt z.B. Microsoft für das Betriebssystem Windows spezielle öffentliche Schnittstellen für Gerätetreiber zur Verfügung, und diese Schnittstellen bleiben über längere Zeit unverändert (Klageschrift S. 17). Unter dieser Voraussetzung werden üblicherweise die Hardeware-Unternehmen, die die anzusprechenden externen Geräte herstellen, in Kenntnis der Spezifikation der abstrakten Schnittstelle den für ihr jeweiliges Gerät zu dieser Schnittstelle passenden Gerätetreiber entwickeln und zur Verfügung stellen (vgl. Beklagten-Ss 01.07.2015 S. 7 = Bl. 61).

Im Gegensatz dazu sind **Linux-Kernelmodule** keine klassisch dynamisch ladbaren Treiber und andere Module, die anhand einer abstrakten Schnittstellendefinition entwickelt werden, sondern voll integrierte Teile des Linux-Gesamtprogramms, die vollen Zugriff auf die Interna des Linux-Betriebssystems haben und allenfalls zur Reduzierung des Ressourcenverbrauchs ausgelagert werden (Klageschrift S. 11). Anders als bei anderen Betriebssystemen sind bei Linux **die Treiber und der Kern** des Betriebssystems grundsätzlich untrennbar; bei Linux stammen die internen Schnittstellen und die Treiber regelmäßig aus der gleichen Hand und werden zusammen erstellt. Die **Schnittstellen bei Linux** sind nicht stabil, also weder gleichbleibend noch rückwärtskompatibel. Vielmehr werden Schnittstellen und Treiber ständig aneinander angepasst. Der Betriebssystemkern und die Treiber sind voneinander abhängig und entwickeln sich gemeinsam und gegenseitig (Klageschrift S. 17).

Die **Beklagte** ist Herstellerin der **Software ESXi**; die Beklagte nennt die Software selbst ihr „Hauptprodukt". Bei ESXi handelt sich um eine sog. **Virtualisierungssoftware**, die es dem Nutzer erlaubt, einen Computer in mehrere virtuelle Maschinen aufzuteilen. ESXi besteht aus mehreren unterschiedlichen Programmen und Komponenten. Insbesondere enthält ESXi eine Komponente **„Hypervisor"**, die eine besondere Art eines Betriebssystems darstellt, welche es ermöglicht, dass mehrere unterschiedliche Betriebssysteme auf ein und demselben Computer parallel zueinander ablaufen können. Wie andere Betriebssysteme enthält auch ESXi einen Kernel; dieser ist als **„vmkernel"** bezeichnet (dabei steht die Abkürzung „vm" für „virtual machine"). Ist auf einem Computer ESXi installiert, kann der Nutzer den „vmkernel" anweisen, eine virtuelle Maschine zu starten und dann auf dieser virtuellen Maschine ein Betriebssystem installieren. Der Nutzer kann auch mehrere virtuelle Maschinen starten und auf diese Weise eine beliebige Anzahl von Betriebssystemen auf den von „vmkernel" erstellten virtuellen Maschinen installieren. Auf diese Weise kann unter ESXi auch das Linux-Betriebssystem auf einer virtuellen Maschine installiert werden.

Zwischen den Parteien ist unstreitig, dass der **Aufbau von ESXi** eine Architektur aufweist, bei der der Betriebssystem-Kernel von sog. Kernelmodulen getrennt ist:

- Der eigentliche **Kernel** des ESXi ist der sog. **„vmkernel"**. Dieser liegt nur im sog. Binär- oder Objektcode vor, d.h. er kann von einem Programmierer ohne Übersetzung ein Quellcode nicht überarbeitet werden. Den Quellcode für den „vmkernel" hat die Beklagte nicht offen gelegt; sie hat für den „vmkernel" auch keine Lizenzierung, insbesondere keine solche unter der GPL-2.0 angeboten. Der Kläger macht nicht geltend, dass der „vmkernel" als solcher auch selbst Linux-Code enthalte.

- Daneben besteht ein **Modul**, welches die Beklagte als **„vmklinux"** bezeichnet. Zwischen den Parteien ist unstreitig, dass dieses Modul Teile aus bearbeitetem Code des Linux-Betriebssystems enthält (streitig ist v.a., um welche Codeteile es sich handelt und ob diese vom Kläger stammen). Den Quellcode dieses Moduls hat die Beklagte unstreitig auf ihrer Website offen gelegt und zum Download angeboten und insofern unter der GPL-2.0 lizenziert.

- Schließlich enthält ESXi **weitere Module**, in denen sich **Gerätetreiber** finden, die zum Teil bearbeitete Gerätetreiber sind.

Die Beklagte bietet – nach dem klägerischen, von der Beklagten pauschal bestrittenen Vortrag - die Software ESXi in der im Klagantrag zu 1) bezeichneten Version als ausführbares Programm, d.h. im sog. Objektcode, über ihre Website u.a. in Deutschland zum Download über das Internet an. Der Kläger sieht darin eine Verletzung seiner Rechte am verwendeten Linux-Code und mahnte die Beklagte mit Schreiben vom 20.08.2014 ab (Anlage K 8). Die Beklagte antwortete und gab keine Unterlassungsverpflichtungserklärung ab (Schreiben 19.09.2014, Anlage K 9). Der Kläger übersandte eine E-mail vom 14.11.2014 (K 10). Anschließende Gespräche der Parteien führten nicht zu einer Einigung.

Der Kläger macht geltend:

Er, der Kläger, habe **erhebliche Teilen des Linux-Kernels bearbeitet** und insofern Bearbeiter-Urheberrechte erworben. Der vom Kläger entwickelte Sourcecode sei in dem sog. **Git-Repository** unter der URL <https://git.kernel.org/cgit/linux/kernel/> für jedermann öffentlich einsehbar (Ss 25.09.2015 S. 3 = Bl. 147 d.A.). Es könne in den Repositories des Mainline-Kernels (d.h. der offiziellen

Kernelversion) jeder Entwicklungsbeitrag des Klägers detailliert nachvollzogen und belegt werden (Klageschrift S. 15).

Der Kläger hat außerdem auf der **CD-Rom K 12** (Ordner **history.tgz**) den nach seinen Angaben vollständigen Sourcecode des letzten Standes der Entwicklung des Linux-Programms mit dem Sourcecode-Verwaltungssystem Bitkeeper (**Linux Version 2.6.12-rc2**) als Archiv (history.tgz) und die gesamte Versionsgeschichte bis dahin im Git-Format vorgelegt (Ss 25.09.2015 S. 10 = Bl. 154 d.A.). Ferner verweist der Kläger für die CD-Rom K 12 auf den dortigen Ordner **„linux.tgz"** und erklärt, er lege damit das Archiv vor, das sowohl den Stand des Linux-Kernels in Version 2.6.18 enthalte als auch die gesamte Versionsgeschichte zwischen Linux 2.6.12-rc und Linux 2.6.18 im Git-Format (Ss 25.09.2015 S. 10 = Bl. 154 d.A.). Die **Version 2.6.18** des Linux Kernels sei diejenige, die der von der Beklagten verwendeten Version am ähnlichsten sei (Ss 25.09.2015 S. 10 = Bl. 154 d.A.; vgl. auch Ss 29.04.2016 S. 10 = Bl. 274 d.A.). Mit der CD-Rom K 12 liege also der Sourcecode vor, den der Kläger zu Linux beigetragen habe und der sich in dem Modul „vmklinux" der Beklagten wiederfinde (Ss 25.09.2015 S. 10 = Bl. 154 d.A.). Die weiter mit CD-Rom K 12 vorgelegten **„.blame"-Dateien** zeigten, welcher Code vom Kläger stamme und welche Änderungen die Beklagte vorgenommen habe (Einzelheiten Ss 25.09.2015 S. 3 = Bl. 147 d.A.).

Der Kläger macht weiter geltend, dass er mit der – nach Existenz und Wortlaut unstreitigen - vorgerichtlichen **e-mail K 10** der Beklagten gegenüber Beispiele dafür genannt, welche Teile von bestimmten Sourcecode-Dateien (scsi_error.c, scsi_lib.c, scsi_proc.c scsi.c und host.c) er entwickelt habe (Ss 25.09.2015 S. 2 = Bl. 146 d.A.).

Sodann legt der Kläger die dreiseitige Anlage **K 15** vor und erklärt, aus den dortigen Ausführungen ergebe sich, welche Beiträge er geleistet habe, die durch die Beklagte übernommen worden seien (Ss 25.09.2015 S. 11 = Bl. 155 d.A.). Dass sich der vom Kläger entwickelte Code in urheberrechtlich relevantem Umfang in ESXi befinde, könne anhand des in Anlage K 15 beigefügten Sourcecode-Vergleichs beispielhaft nachvollzogen werden; diese Anlage beschreibe auch, wie der Vergleich nachvollzogen werden könne; ein vollständiger Source-Code-Vergleich sei mit Hilfe des Linux-Codes in K 15 und dem Sourcecode von „vmklinux", den die Beklagte anbiete, durchgeführt worden (Ss 25.09.2015 S. 11 = Bl. 155 d.A.).

In zwei Dateien, zu denen die Beklagte den Quellecode offen gelegt habe, fänden sich im Übrigen sog. **Header mit Hinweisen auf Urheberrechte** u.a. des Klägers (Einzelheiten Klageschrift S. 14 = Bl. 14 d.A.). Der Kläger verweist in diesem Zusammenhang auf Anlage K 3.

Der Kläger trägt vor, er habe vor allem am sog. **SCSI-Subsystem des Linux-Kernels** mitgearbeitet. Dieser Code sei seit 1992 (damals noch ohne seine Beteiligung) als Teil des Linux-Kernels entwickelt und immer zusammen mit dem Kernel als Teil eines einzigen Programms veröffentlicht worden. Es sei zwar möglich, das SCSI-Subsystem in ein separates Kernelmodul auszulagern, jedoch werde davon in der Praxis nur selten Gebrauch gemacht. Das SCSI-Subsystem interagiere direkt mit der Speicherverwaltung, den Dateisystemen und dem Scheduler (zur Steuerung von Prozessen des Betriebssystems) und werde nicht über abstrakte Schnittstellen angesprochen (Klageschrift S. 12). Der Kläger selbst sei in der Zeit von 2000-2004 einer der aktivsten Entwickler am SCSI-Subsystem des Linux-Kernels gewesen, das sicherlich eine komplexe und damit schutzfähige Programmierung darstelle (Ss 25.09.2016 S. 8 = Bl. 152 d.A.). Andere Linux-Kernel-Entwickler könnten bezeugen, dass der Kläger wesentliche und komplexe Beiträge zu der Entwicklung des SCSI-Subsystems geleistet habe (Ss 25.09.2015 S. 10 = Bl. 154).

Der Kläger trägt ferner vor, er habe zwar Anzeichen dafür, dass ein Teil der SCSI-Funktionalität sich auch **im „vmkernel"** befinde, müsse jedoch mangels anderer Anhaltspunkte davon ausgehen, dass es sich insoweit um Eigenentwicklungen der Beklagten handele (Ss 29.04.2016 S. 6-7 = Bl. 270 f.).

Der Kläger behauptet, das **„SCSI-Hotplug"** sei eine der Funktionen, die ESXi benutze (Ss 25.09.2015 S. 8 = Bl. 152 d.A.). Innerhalb des SCSI-Subsystems sei die sog. „SCSI-Hotplug"-Funktionalität, also das dynamische Einbinden und Ausbinden von Hardware im laufenden Betrieb, eine der wichtigsten Funktionen (Ss 25.09.2015, S. 8 = Bl. 152 d.A.). Die dieser Funktionalität zugrunde liegende Programmierung genieße für sich genommen urheberrechtlichen Schutz, denn sie sei ein Beispiel für eine komplexe Funktionalität und deren Implementierung, die zu einer tatsächlichen Vermutung für hinreichende Individualität führe (a.a.O. S. 9 = Bl. 153 d.A.). Der entsprechende Code sei dem Programm Linux über viele Schritte hinzugefügt worden und verteile sich über mehrere Einzelfunktionen (a.a.O.) Der Kläger habe dabei die im Ss vom 25.09.2015 S. 9 (Bl. 153 d.A.) genannten **19 Einzelbeiträge (Patches)** beigesteuert; der Kläger verweist auch auf Kopien dieser „Patches" auf der CD-Rom Anlage K 12. Dass es sich um eine komplexe Programmierung handele, könne durch einen Sachverständigen belegt werden (a.a.O.). Soweit die Beklagte zu einzelnen Patches die Urheberschaft des Klägers bestritten habe, trägt der Kläger zu von ihm behaupteten Bearbeitungsteilen vor (Einzelheiten Ss 25.09.2015 S. 6-7 unter II.4.a)-c) = Bl. 150 f. d.A.). Der Kläger trägt ferner vor, er habe am 21.02.2003 zum SCSI einen Beitrag **(Patch) „scsi_scan.c restructuring for ieee1394 hotplugging"** beigesteuert, der am 23.03.2003 in Linux übernommen worden sei. Dass seine Funktionalität in „vmklinux" übernommen worden sei, könne man an bestimmten Aufrufen der Funktionen „scsi_add_device" „SCSI_remove_device" in allen „vmklinux" SAS-Treibern sowie in dem Libata-Modul erkennen (Ss 29.04.2016 S. 11 = Bl. 275).

Der Kläger trägt weiter zu Teilen von SCSI-Funktionalität vor, die sich in „vmklinux" befinden sollen und die für mehrere Hardwaretreiber benötigt würden (sog. **„Midlayer-Code"**) und quasi vor die Klammer gezogen worden sei; das gelte z.B. für die „Error Handling"-Funktion; ein weiterer Teil der SCSI-Funktionalität finde sich in den **Hardwaretreibern**, die an „vmklinux"-angebunden seien (Ss 29.04.2016 S. 7 = Bl. 271 d.A.).

Der Kläger trägt schließlich zum SCSI vor, es sei seitens der Beklagten eine Funktion **„scsi_remove_single_device"** identisch übernommen worden; sie finde sich im Linux Kernel 2.5.64 in der Datei „drivers/scsi/scsi_proc.c" in den Zeilen 423 ff. (K 25) und bei VWware ESXi 5.5 Update 2 in der Datei „vmdrivers/src_92/vmklinux_92/linux/scsi/scsi_proc.c" in den Zeilen 250 ff. (K 26); es bestehe Übereinstimmung von 99% (K 27) (Ss 29.04.2016 S. 12-13 = Bl. 276-277).

Der Kläger macht weiter geltend, er habe auch bei der Entwicklung der sog. **Radix-Trees** wesentlich mitgearbeitet. Bei den Radix-Trees handele es sich um eine skalierbare Implementierung einer speziellen Baumstruktur, die insbesondere in der Verwaltung von Dateisystemen-Puffern (Caches), aber auch in vielen anderen Anwendungsbereichen benutzt werde. Als zentraler Teil des Linux-Betriebssystemkerns sei diese Funktionalität nie zur Auslagerung in Modulen vorgesehen gewesen und werde durch andere Nutzer als die Beklagte auch nicht in Kernelmodule ausgelagert (Klageschrift S. 12). Zuletzt hat der Kläger vorgetragen, Linux „Radix-Tree" sei eine effiziente Implementation einer Datenstruktur zum Finden von Objekten mittels eines Indexes. Die Datenstruktur sei von dem Programmierer Velikov erfunden worden. Die Implementation sei eine gemeinsame Arbeit von Velikov und dem Kläger. Der Kläger habe am 09.04.2002 die Implementation zur Aufnahme in Linux eingereicht. Die Beklagte habe eine neuere Version des Codes von 2012 übernommen und minimal bearbeitet (Ss 29.04.2016 S. 13-15 unter Verweis auf K 29-31). Andere Linux-Kernel-Entwickler könnten bezeugen, dass der Kläger wesentliche und komplexe Beiträge zu der Entwicklung der Radix-Trees geleistet habe (Ss 25.09.2015 S. 10 = Bl. 154).

Ebenfalls habe die Analyse ergeben, dass **viele andere Teile des Linux-Betriebssystems** von der Beklagten verwendet würden, zu denen der Kläger kleinere Bearbeitungen des Quellcodes beigetragen habe (Klageschrift S. 15 = Bl. 15 d.A.).

Der Kläger wirft der Beklagten vor, die Übernahme des Linux-Code in „vmklinux" sei rechtswidrig, weil die **Lizenzierungs-Bedingungen der GPL-2.0 nicht eingehalten** worden seien. Die GPL-2.0 verlange nach ihrer Ziffer 2.b) – insofern unstreitig – dass der Nutzer *„must cause any work [...], **that in whole or in part contains or is derived** from the Program or any part thereof, to be licensed as a whole [...] under the terms of this License"*. Der Kläger gesteht der Beklagten zwar zu, den Quellcode des Moduls „vmklinux" offengelegt und unter GPL-2.0 lizenziert zu haben. Damit allein seien die Anforderungen der GPL-2.0 zur Nutzung des Linux-Codes jedoch noch nicht erfüllt und daher eine etwaige Lizenzierung nach Ziffer 4 der GPL-2.0 entfallen. Denn die oben zitierte Klausel müsse auf den vorliegenden Fall dahin angewandt werden, dass **„vmklinux" und „vmkernel" als ein einheitliches „work"** betrachtet werden müssten und daher auch insgesamt unter der GPL-2.0 lizensiert werden müssten, d.h. auch der **Quellcode zu „vmkernel" müsse von der Beklagten offen gelegt werden**, um den Anforderungen der GPL-2.0 zu genügen. Die Notwendigkeit, „vmklinux" und „vmkernel" als Einheit betrachten zu müssen, ergebe sich aus folgenden, vom Kläger behaupteten Eigenschaften des Programms der Beklagten:

- Die diesbezügliche Architektur von der Software der Beklagten ergebe sich aus Anlage K 2.

- Der Linux-Code werde dynamisch in den Bestandteil „vmkernel" nachgeladen. Das gelte insbesondere auch für das SCSI-Subsystem, das den Code des Klägers enthalte. Dabei werde der Code im sog. Kernel-Space ausgeführt, d.h. nicht in dem Bereich für selbstständige Anwendungen (User-Space), sondern als Teil des Betriebssystems (Klageschrift S. 15 = Bl. 15 d.A.).

- Sowohl „vmkernel" als auch „vmklinux" würden in demselben Adressraum ausgeführt, d.h. Funktionen in „vmkernel" riefen Funktionen in „vmklinux" auf, welche dann wieder Funktionen in „vmkernel" benutzten. Nach der Verkehrsauffassung sei dies ein typisches Merkmal für ein einheitliches Programm, denn selbstständige Programme würden typischerweise in getrennten Adressräumen ausgeführt und implementierten eigenständige Funktionalität, die keine tiefgreifende Interaktion der Komponenten beinhalte (Klageschrift S. 15-16 = Bl. 15-16 d.A.).

- Der „vmkernel" sei ohne die Benutzung des „vmklinux"-Moduls mit dem bearbeiteten Linux-Code nicht lauffähig (Klageschrift S. 29). Im ESXi-Programm der Beklagten sei der Bereich „vmkernel" nicht alleine lauffähig und benötige zwingend zusätzliche Softwarebestandteile, um eine Hardware ansprechen zu können. Diese Softwarebestanteile fänden sich in den aus dem Kernel ausgelagerten Kernel-Modulen, so insbesondere im Modul „vmklinux". Der „vmkernel" sei also ohne die in die Module ausgelagerte Software nicht lauffähig (Klageschrift S. 8).

- Auch „vmklinux" sei ohne „vmkernel" nicht lauffähig. Ziel der Beklagten sei es gewesen, den stark bearbeiteten Linux-Code in enger Integration mit „vmkernel" benutzen zu können. Dafür habe es der Beklagten nicht genügt, die übernommenen Linux-Module im Objektcode zu verwenden und bloß über eine Schnittstelle anzubinden; vielmehr sei eine darüber hinaus gehende Integration in den Kernel von ESXi erforderlich gewesen, nämlich eine jeweils spezielle Anpassung und Einzelabstimmung (Klageschrift S. 13). Ergebnis sei, dass „vmklinux" nicht als Teil von Linux ausgeführt werden könne. Insbesondere könne „vmklinux" der Beklagten nicht in den Linux-Kernel nachgeladen werden (Klageschrift S. 29) oder mit einem anderen Betriebssystem ausgeführt werden (Klageschrift S. 13).

Aus einerseits der Notwendigkeit, „vmkernel" und „vmklinux" als Einheit betrachten zu müssen, und andererseits dem Umstand, dass in „vmklinux" Teile des Codes von Radix-Trees und des SCSI-Subsystems übernommen worden seien, folge die Bewertung, dass die Software VMware ESXi als

„combined word" bzw. „derivative work", also als Bearbeitung der Programmierleistung des Klägers angesehen werden müsse (Klageschrift S. 29).

Der Kläger beantragt,

die Beklagte zu verurteilen,

1.

es bei Vermeidung eines vom Gericht für jeden Fall der Zuwiderhandlung festzusetzenden Ordnungsgeldes bis zu 250.000 € (und für den Fall, dass dieses nicht beigetrieben werden kann, einer Ordnungshaft) oder einer Ordnungshaft bis zu 6 Monaten, diese zu vollziehen am ihrem ständigen Vertreter, zu unterlassen,

den Kernel (also einschließlich der Komponenten „vmkernel", „vmklinux" und „VMK API") der Software Hypervisor vSphere VMware ESXi 5.5.0 (einschließlich Update 1 und Update 2) öffentlich zugänglich zu machen, ohne dass zugleich entsprechend den Lizenzbedingungen der GNU General Public License, Version 2,

- der vollständige korrespondierende Quellcode des Kernels der Software Hypervisor vSphere VMware ESXi 5.5.0 (einschließlich Update 1 und Update 2) lizenzgebührenfrei zugänglich gemacht werde, und
- der Kernel der Software Hypervisor vSphere VMware ESXi 5.5.0 (einschließlich Update 1 und Update 2) unter den Lizenzbedingungen der GNU General Public Licence, Version 2, angeboten werde;

2.

an den Kläger 4.046,00 € nebst Zinsen hieraus in Höhe von 5 Prozentpunkten über dem jeweiligen Basiszinssatz seit Rechtshängigkeit zu zahlen.

Die Beklagte beantragt,

die Klage abzuweisen.

Sie macht geltend:

Der Kläger versuche, die Beklagte zu zwingen, auf den Schutz ihres geistigen Eigentums an dem Produkt ESXi de facto zu verzichten, indem ESXi Gegenstand einer „open source"-Softwarelizenz werden solle, dies allein mit der Begründung, die zentrale Komponente von ESXi, der „vmkernel", enthalte zwar selbst keinen Linux-Code, kommuniziere aber mit einem eigenständigen Modul, welches seinerseits Linux-Code verwende. Ein solcher Anspruch stehe dem Kläger nicht zu. Insbesondere habe der Kläger weder hinreichend dargelegt, dass und welche Anteile er am Linux-Code habe, die überhaupt in „vmklinux" verwendet worden seien, noch sei die Argumentation des Klägers zur Einordnung des „vmkernel" als eines aus Linux im Sinne der GPL-2.0 „abgeleiteten" Werkes tatsächlich und rechtlich zutreffend . Im Einzelnen:

Die **Klage sei schon unzulässig**, da der Antrag zu unbestimmt sei und der Kläger kein Rechtsschutzbedürfnis daran habe (Einzelheiten Klageerwiderung S. 48 f. = Bl. 102 f. d.A.; Ss 15.04.2016 S. 9 ff. = Bl. 259 ff. d.A.). Sie sei aber **auch unbegründet**.

Der Kläger habe ein **schutzfähiges Softwarewerk nicht dargestellt**, geschweige denn, dass ein solches und **welches genau von ihm geschaffen worden sei**. Auch der Verweis auf „linux.tgz-Archivs" und „Git-Repository" sei nicht ausreichend; der Kläger versuche nicht einmal, einzelnen Code zu identifizieren, sodass nur geraten werden könne, welche Code-Zeilen der Kläger zur Begründung seines Anspruch heranziehen wolle (Ss 05.02.2016 S. 7-8 = Bl. 196-197).

Soweit der Kläger für seine Urheberschaft auf das **SCSI-Subsystem** verweise, sei der Vortrag unsubstanziiert, weil der Kläger nicht erläutere, welche bestimmten Dateien des Linux-Kernels seiner Ansicht nach das SCSI-Subsystem bilden sollen (Klageerwiderung Seite 11 = Bl. 68 d.A.). Unklar sei auch, was genau der Kläger dazu beigetragen habe (Klageerwiderung S. 49 f. = Bl. 103 f. d.A.).

Auch soweit sich der Kläger für seine Urheberschaft auf **Radix-Trees** berufe, sei der Vortrag unsubstanziiert. Radix-Trees seien Jahrzehnte alte Dateien-Strukturen, die dem schnellen Aufruf von Dateien dienten. Der Kläger habe nicht erläutert, welchen Code er geschrieben habe und dass dieser Code hinreichend kreativ für einen Urheberrechtsschutz sei (Klageerwiderung Seite 11 = Bl. 68 d.A.). Darüber hinaus seien Radix-Trees schon nach dem eigenen Vortrag des Klägers lediglich Strukturen; bloße Dateistrukturen seien aber nach der Rechtsprechung des EuGH gerade keine urheberrechtlich geschützten Computerprogramme (Klageerwiderung S. 50 = Bl. 104 d.A.).

Der Verweis des Klägers auf ein **Repository** zum Beleg seiner Urheberschaft an bestimmten Code sei ebenfalls nicht ausreichend; der Vortrag des Klägers, er habe zum Code „beigetragen" bzw. sei daran „beteiligt", stelle nicht genau genug dar, was der Kläger geschaffen haben soll; vielmehr ergäben Analysen der Beklagten, dass tatsächlich andere Programmierer den gesamten oder zumindest Teile des Codes, für den der Kläger Urheberschaft beanspruche, geschrieben hätten (Klageerwiderung S. 13 f. = Bl. 67 f. d.A. mit Beispielen und Einzelheiten).

Die Beklagte **bestreitet angebliche Beiträge des Klägers** mit Nichtwissen (Klageerwiderung S. 26 = Bl. 80 d.A.). Sie bestreitet auch die **urheberrechtliche Schutzfähigkeit** etwaiger Beiträge (Ss 05.02.2016 S. 8 = Bl. 197 d.A.).

Selbst wenn aber der Kläger teile des Linux-Kernel geschaffen haben sollte, so könne er insofern **lediglich Bearbeiter-Urheberrechte** erworben haben, und diese seine Rechte als Bearbeiter-Urheber seien dann auf die bearbeiteten Teile begrenzt (Klageerwiderung S. 53 = Bl. 107 d.A.). Der Kläger könne keine Rechte geltend machen, die über seine eigenen Beiträge hinausgingen (SS 05.02.2016 S. 12 f. = Bl. 201 f. d.A.)

Der Kläger liefere **keinen ausreichenden Vortrag zu einer verletzenden Übernahme von Code**, nämlich dazu, dass die Software der Beklagten tatsächlich originäre und/oder bearbeitete Werke/Werkteile enthalte, an denen dem Kläger Rechte zuständen, bzw. „vmkernel" aus schutzfähigen Linux-Anteilen des Klägers abgeleitet sei (Klageerwiderung S. 54 = Bl. 108 d.A.).

Das Produkt **ESXi** der Beklagten sei kein einheitliches ausführbares, sondern ein Softwareprodukt, welches viele Werke sowie Komponenten, Funktionen und Merkmale umfasse; der Kläger habe nicht dargetan, dass und wie die Beklagte ESXi zum Download über das Internet angeboten habe (Klageerwiderung S. 29 = Bl. 83 d.A.).

Soweit der Kläger der Beklagten vorwerfen, sie habe Linux-Code überarbeitet, um ihnen enge Integration mit dem proprietären Bestandteil von „vmkernel" benutzen zu können, so sei dieser Vortrag unsubstanziiert; zudem habe die Beklagte Linux-Kernel-Code verwendet, um eine Technologie zu entwickeln, die einen völlig anderen Zweck habe als der Linux-Kernel, nämlich um ein Interoperabilitätsmodul zu schaffen, was unter der GPL gestattet sein (Klageerwiderung S. 35 = BL. 89

d.A.). Jeglicher SCSI-Code in „vmklinux" diene etwa einem anderen Zweck als das **SCSI-Subsystem** innerhalb von Linux (Klageerwiderung Seite 34 = Bl. 88 d.A.).

Auch soweit der Kläger auf die **SCSI-Hotplug-Funktionalität** verweise, sei die Fähigkeit eines Systems, Komponenten auszutauschen, ohne das System herunterfahren zu müssen, weder eine Erfindung oder Schöpfung des Klägers noch ein Feature, das in einer einzigen Funktion implementiert werde. Die vom Kläger hierzu in Bezug genommenen **„Patches"** seien nicht aussagekräftig, denn sie bezögen sich auf den Linux-Kernel, nicht aber auf „vmklinux" und könnten daher keine Aussage darüber treffen, ob von demjenigen Code, den der Kläger geschrieben haben wolle, überhaupt etwas schutzfähig und von der Beklagten verwendet worden sei. Tatsächlich enthielten die Funktionen in „vmklinux", die sich auf die Hotplug-Funktionalität bezögen, keinen wesentlichen vom Kläger stammenden Code (Ss 05.02.2016 S. 11 = Bl. 200 d.A.).

Es sei im Übrigen insgesamt davon auszugehen, dass etwaige Linux-Code-Beiträge des Klägers, soweit sie innerhalb von „vmklinux" und damit von ESXi verwendet worden sein sollten, dem Umfange nach so gering seien, dass sie im Verhältnis zur Gesamtheit von „vmklinux", „vmkernel" und „ESXi" so sehr zurücktreten, dass **keine Umarbeitung oder Bearbeitung**, sondern **allenfalls eine freie Benutzung** vorliege (Klageerwiderung Seite Ziffer 14 = Bl. 68 d.A.; Ss 05.02.2016 S. 20 ff. = Bl. 209 ff.). Die Beklagte habe erhebliche Änderungen am Linux-Code vorgenommen (Klageerwiderung S. 15 = Bl. 69 d.A.):

- Durch Analyse der vom Kläger vorgelegten Informationen habe die Beklagte in etwa 798 Zeilen Code von „vmklinux" identifizieren können, von denen der Kläger behauptete, dass er sie geschrieben habe. Bereits „vmklinux" selbst umfasse aber 214.000 Code-Zeilen (also das 269-fache); „vmkernel" umfasse 1,3 Mio. Code-Zeilen (das 1.654-fache) (Ss 05.02.2016 S. 4 = BL. 193 d.A.).
- Von den 798 Code-Zeilen seien jedoch 396 Zeilen in „vmklinux" nicht verwendet, sondern „auskommentiert" worden (d.h. sie erscheinen nur im menschenlesbaren Quellcode) (Ss 05.02.2016 S. 6 = Bl. 46 d.A.).
- Von den verbleibenden 402 Zeilen Code stellten wiederum knapp die Hälfte, nämlich 185 Zeilen, solchen Code dar, der nicht vom Kläger stamme, sondern von ihm nur geringfügig geändert oder an eine andere Stelle bewegt worden sei (Ss 05.02.2016 S. 6 = Bl. 46 d.A.).
- Von den verbleibenden 217 Code-Zeilen seien ferner 68 bloße „Kommentare" (zum Beispiel Erklärungen zur Funktionalität des Codes), die auch nicht in den endgültigen maschinenlesbaren, ausführbaren Code kompiliert seien (Ss 05.02.2016 S. 6 = Bl. 46 d.A.).
- Es verblieben damit **letztlich nur noch 149 Zeilen, die möglicherweise von Kläger stammten** und zum Endbenutzer gelangten (Ss 05.02.2016 S. 6 f. = Bl. 46 f. d.A., auch zum Folgenden). Allein die drei „vmklinux"-Dateien, auf die sich der Kläger beziehe, enthielten aber bereits 6895 Codezeile, zu denen der Kläger dann also vom Umfang her betrachtet weniger als 2,2 % beigetragen habe. Zu „vmklinux" insgesamt mit seinen 214.000 Codezeilen haben der Kläger dann also lediglich 0,07% beigetragen. **Zu „vmkernel"** mit seinen 1,3 Mio. Zeilen hätte der Kläger – wenn man, wie von der Beklagten verneint, „vmkernel" und „vmklinux" als Einheit betrachten würde, **weniger als 0,012%** beigetragen.

Es liege **kein Verstoß gegen die Lizenzbedingungen der GPL-2.0** vor (Klageerwiderung S. 56 ff. = Bl. 110 d.A.). Den Quellcode von „vmklinux" als solchen habe die Beklagte (unstreitig) im Einklang mit der GPL lizenziert (Klageerwiderung Seite 9 = Bl. 63 d.A.). Der Quellcode von „vmkernel" müsse nach diesen Bedingungen nicht offen gelegt werden. Zwar sähen die Bedingungen vor, dass der Quellcode eines sogenannten „derivative work's", also eines vom frei lizenzierten Werk abgeleiteten Werks, offen gelegt werden müsse. Jedoch sei **„vmkernel" kein von Linux abgeleitetes Werk.**

„Vmkernel" sei ein unabhängig entwickeltes Werk, dessen Entwicklung die Beklagte vor über 15 Jahren begonnen habe (Klageerwiderung Seite 2 = Bl. 56 d.A.) und für die sie erhebliche Investitionen getätigt habe. „Vmkernel" als solcher sei von der Beklagten (unbestritten) zeitlich entwickelt worden, bevor der Kläger sein angebliches Softwarewerk geschaffen habe (Klageerwiderung Seite 10, 16 = Bl. 64, 70 d.A.). Dass „vmkernel" nicht allein lauffähig sei, sondern zusätzliche Softwarebestandteile benötige und diese aus dem Linux entnommen worden seien, bestreitet die Beklagte (Klageerwiderung S. 29 = Bl. 83 d.A.). Im Übrigen könne kein Betriebssystem-Kernel ohne Gerätetreiber laufen oder habe irgendeinen praktischen Nutzen; aus der Tatsache, dass jeder Kernel also Gerätetreiber benötige, könne nicht gefolgert werden, dass es sich dann automatisch um ein abgeleitetes Werk i.S. der GPL handele (Klageerwiderung S. 64 f. = Bl. 118 f. d.A.).

„Vmklinux" sei eine eigenständige, ebenfalls von der Beklagten entwickelte Komponente; sie ermögliche die Kommunikation zwischen „vmkernel" und Linux-kompatiblen Gerätetreibern, also dass dritte Hardwarehersteller die von ihnen für den Linux-Kernel entwickelten Gerätetreiber nun mit dem „vmkernel" weiterverwenden könnten (Klageerwiderung Seite 3 = Bl. 57 d.A.). Wie auch andere (Linux-kompatible und andere) Gerätetreiber, die mit dem „vmkernel" kommunizierten, sei auch „vmklinux" **ein vom „vmkernel" separates und eigenständiges** Modul (Klageerwiderung Seite 7 = Bl. 61 d.A.); es sei von der Beklagten gezielt als **Gerätetreiber-Interoperabilitätsmodul** entwickelt worden (Einzelheiten Klageerwiderung S. 15 = Bl. 69 d.A.). Dass „vmklinux" nicht ohne „vmkernel" lauffähig sei, sei unerheblich und irreführend (Klageerwiderung S. 36 = Bl. 90 d.A.). Ob eine Komponente selbstständig lauffähig sei, sei nicht entscheidend; so seien ja zum Beispiel Anwendungsprogramme auf Betriebssysteme angewiesen. Wenn der Kläger geltend mache, es gebe eine Verkehrsauffassung dahin, dass Kernel-Module als Teil des Kernel anzusehen sein, wenn sie nicht auch in unveränderter Form mit anderen unixartigen Betriebssystemen lauffähig seien, so sei dies eine nicht bindende nachträgliche Interpretation der GPL; ob sie auf den Linux-Kernel und Linux-Kernelmodule zutreffe, sei unerheblich, jedenfalls gelte sie nicht für „vmkernel" (Klageerwiderung S. 39-40 = Bl. 93-94 d.A.). Im Übrigen bildeten auch Linux-Kernelmodule kein einheitliches Werk mit dem Linux-Kernel (Klageerwiderung S. 32 = Bl. 86 d.A.).

Somit seien **„vmkernel" und „vmklinux" kein einheitliches Programm**. Vielmehr existierten beide (insofern unstreitig) als zwei getrennte Binärdateien. „Vmkernel" sei die Kernkomponente von ESXi, „vmklinux" dagegen ein wesentlich kleineres, getrenntes Modul, welches konzipiert sei, um Geräteherstellern die Wiederverwendung ihrer Linux-kompatiblen Gerätetreiber mit ESXi zu ermöglichen.

Zwar kommuniziere also „Vmkernel" mit dem Modul „vmklinux". „Vmkernel" weise eine vom Linux-Kernel grundverschiedene Architektur auf, da Gerätetreiber nicht (wie beim Linux-Kernel) in den Kernel integriert seien, sondern über eine stabile und dokumentierte Schnittstelle, die „VMK API" angesprochen werden müssten (Klageerwiderung S. 17 = Bl. 71 d.A.). Der Umstand allein, dass „vmkernel" mit dem selbstständigen „vmklinux" kommuniziere und dieses Modul Linux-Code enthalte, führe nicht dazu, auch „vmkernel" als aus Linux abgeleitet ansehen zu müssen (Klageerwiderung Seite 4 = Bl. 58 d.A.).

Diese **Kommunikation zwischen „vmkernel" und „vmklinux"** erfolge nicht über instabile interne Kernel-Schnittstellen (Klageerwiderung S. 64 = Bl. 118 d.A.), sondern über die **Schnittstelle „VMK API"**, bei der es sich (anders, als vom Kläger behauptet) um eine **stabile** und **dokumentierte** Schnittstelle handele (Klageerwiderung Seite 3 = Bl. 57 d.A.). „VMK API" sei eine Schnittstelle, die von vielen Komponenten und Gerätetreibern genutzt werde; sie sei stabil und grundsätzlich **rückwärtskompatibel**; derzeit seien von 21 Unternehmen insgesamt ca. 122 Gerätetreiber und andere Module entwickelt worden, welche alle über „VMK API" mit dem „vmkernel" kommunizierten (Klageerwiderung S. 15 f. = Bl. 69 f. d.A.). So könne „vmkernel" über „VMK API" und

native Treiber ansprechen, ohne dass die Kommunikation über „vmklinux" laufe (z.B. Drucker Hewlett Packard Drucker); „vmkernel" sei insofern nicht von „vmklinux" abhängig (Ss 15.04.2016 S. 4 = Bl. 254 d.A.). „VMK API" sei also eine echte Schnittstelle, weil sie nicht nur für die Kommunikation zwischen „vmklinux" und „vmkernel", sondern **auch von hunderten anderer Module verwendet** werde, von dritten Parteien als auch von der Beklagten entwickelt worden seien; zudem weise „VMK API" die typischen Merkmale einer Software-Schnittstelle auf (Ss 05.02.2016 S. 29 ff. = Bl. 218 ff. d.A.). Die Gerätetreiber, die mit „vmkernel" kommunizierten, seien separate Bestandteile (Klageerwiderung S. 31 = Bl. 85 d.A.). „VMK API" sei von der Beklagten schon in 2004 entwickelt und seit 2006 vertrieben worden, während die Version des Produkts der Beklagten, die nach Ansicht des Klägers das angebliche „SCSI- Subsystem" enthalte, erst seit Mai 2009 vertrieben werde, ferner diejenige Version, die angeblich den Radix-Tree-Code enthalten solle, erst seit 2013 (Klageerwiderung S. 16 = Bl. 70 d.A.). „VMK API" sorge somit für eine **klare Trennlinie** zwischen der proprietären Software der „vmkernel" und dem Modul „vmklinux" (Klagerwiderung S. 3 = Bl. 57 d.A.). Bei einer Verbindung über eine solche Schnittstelle sei gerade von getrennten Programmen und damit unabhängigen Werken auszugehen (Ss 05.02.2016 S. 21 = Bl. 2010 d.A.). Wollte man dies anders sehen, so würde dies zu der absurden Situation führen, dass alle Inhaber dritter Module, die mit „vmkernel" über die Schnittstelle kommunizierten, ebenfalls in der Lage wären, Urheberrechte an „vmkernel" geltend zu machen (Schriftsatz 05.02.2016 S. 2 = Bl. 192 d.A.).

Soweit der Kläger argumentierte, die Schnittstelle teile „vmkernel" und „vmklinux" künstlich auf, so verfange diese Argumentation nicht, denn „vmklinux" habe eine von „vmkernel" getrennte, bestimmte Funktionalität, nämlich es Linux-kompatiblen Gerätetreibern zu ermöglichen, mit „vmkernel" zu kommunizieren (Ss 05.02.2016 S. 27 = Bl. 216 d.A.). Der Kläger verwechsele auch die Schnittstellen, über die er spreche: Richtig sei vielmehr, dass es neben der „VMK API" (über die „vmkernel" und „vmklinux" kommunizierten) eine **weitere Schnittstelle** gebe; dieses sei aber diejenige, **über die „vmklinux" mit den Gerätetreibern kommuniziere**, und auch diese Schnittstelle sei nicht in dem Sinne instabil, wie vom Kläger behauptet (Ss 15.04.2016 S. 5 = Bl. 255 d.A.).

Der weitere Umstand, dass **„vmklinux" im sog. Kernel-Space ausgeführt** werde, führe nicht dazu, „vmklinux" und „vmkernel" als einheitliche Software ansehen zu müssen, denn andere Kernel-Erweiterungen und Gerätetreiber würden ebenfalls an dieser Stelle ausgeführt (Klageerwiderung S. 19 = Bl. 73 d.A.). Es gebe auch keine entsprechende Verkehrsauffassung (Klageerwiderung S. 31 = Bl. 85 d.A.). Ob zwei Softwarekomponenten in demselben Speicherraum oder in verschiedenen Speicherräumen aus dem, sei eine technische Begebenheit ohne Bedeutung für die urheberrechtliche Beurteilung (Klageerwiderung S. 38 = Bl. 92 d.A.).

Die Beklagte wendet ferner u.a. **Verwirkung** ein; schließlich habe der Kläger trotz anzunehmender Tatsachenkenntnis mit der Klageerhebung mehrere Jahre zugewartet habe (Klageerwiderung S. 67 = Bl. 121 d.A.; Ss 13.07.2015 S. 3 = Bl. 132 d.A.; Ss 05.02.2016 S. 43 = Bl. 232 d.A.)

Wegen der weiteren Einzelheiten zum Sach- und Streitstand wird auf die gewechselten Schriftsätze der Parteien, soweit sie Gegenstand der am 25.02.2016 geschlossenen mündlichen Verhandlung geworden sind, sowie auf die nachgelassenen, bis zum 15.04.2016 bzw. nach Fristverlängerung bis 29.04.2016 eingegangenen Schriftsätze verwiesen. Die Beklagte hat ferner einen nicht nachgelassenen Schriftsatz vom 10.05.2016 zur Akte gereicht.

Entscheidungsgründe

Die Klage hat im Ergebnis keinen Erfolg.

A.

Die Klage ist allerdings **zulässig**.

Die **internationale und örtliche Zuständigkeit** des Landgerichts Hamburg folgt aus § 32 ZPO, da der Kläger geltend macht, die Beklagte habe ihr streitgegenständliches Programm im deutschen Inland über das Internet zum Download angeboten (doppelrelevante Tatsache).

Die Klage ist auch hinreichend **bestimmt i.S.v. § 253 ZPO**. Der insofern erhobene Einwand der Beklagten, der Kläger müsse bereits im Antrag bezeichnen, welche konkreten Teile aus dem Programm der Beklagten von dieser nicht genutzt werden sollten, greift nicht gegen die Zulässigkeit der Klage durch. Dazu näher:

Wie in der mündlichen Verhandlung besprochen, kommen als Schutzrechte, aus denen der Kläger einen Unterlassungsanspruch ableiten kann, lediglich Bearbeiterurheberrechte in Betracht, da der Kläger geltend macht, sukzessiv eigenständige Beiträge zum Linux-Kernel beigetragen zu haben. Dass auch an einer Umarbeitung von Software ein Bearbeiterurheberrecht entstehen kann, folgt aus § 69c Nr. 2 S. 2 i.V.m. § 3 UrhG. Ob dabei einzelne dieser Beiträge im Zusammenwirken mit anderen geschaffen worden sein mögen, ändert daran nichts, denn insofern würde auch der etwaigen Miturhebergemeinschaft nur ein Bearbeitungsrecht zustehen, dem Kläger also nur ein „Mit-Bearbeiterurheber-Recht".

Dass sich die Bearbeiterurheberstellung des Klägers nur auf einen Teil des Linux-Kernels beziehen mag, steht grundsätzlich der Annahme von eigenen jeweiligen Bearbeiterurheberrechten für solche Teile nicht entgegen. *„Schutzfähig können dabei auch Teile von Computerprogrammen sein, wenn diese ihrerseits die Voraussetzungen für einen urheberrechtlichen Schutz erfüllen [...], was dann der Fall sein kann, wenn sie jedenfalls für den Programmablauf von nicht nur untergeordneter Bedeutung sind und eine eigene Befehlsstruktur haben, die innerhalb eines vorhandenen Gestaltungsspielraumes Ausdruck individuellen Schaffens ist."* (Hanseatisches Oberlandesgericht Hamburg, Urteil vom 11. Januar 2001 – 3 U 120/00 –, Rn. 37, juris). Damit können sich auch Bearbeiterurheberrechte auf Teile von Software beziehen, wenn die Umarbeitungsleistung des Programmierers ihrerseits die vorgenannten Anforderungen erfüllt.

Unter der hier zunächst unterstellten Prämisse, der Kläger habe an Teilen des Linux-Kernels entsprechende Bearbeiterurheberrechte erworben, wäre die Rechtsstellung des Klägers jedoch eine begrenzte. So heißt es z.B. bei Schicker/Loewenheim, UrhR, 4. Aufl. 2010, § 69c Rz. 20: *„Schutzgegenstand des Bearbeiterurheberrechts ist nur die Umarbeitung als solche, am Originalprogramm erwirbt der Bearbeiter keinerlei Rechte."* Das ist – soweit ersichtlich – in Rechtsprechung und Literatur nicht umstritten. Auch der Kläger persönlich ist letztlich davon ausgegangen, wenn er in dem von ihm zitierten Forumsbeitrag aus 2006 (Anlage K 19) davon spricht, er habe leider keine ausreichenden Urheberrechte an der damaligen Linux-Version, um gegen die Beklagte vorgehen zu können.

Gleichwohl kann der Bearbeiterurheber Unterlassung der Nutzung eines solchen anderen Computerprogramms verlangen, welches die für den Bearbeiterurheber geschützten Anteile in i.S.v. § 69c Nr. 2 UrhG (Umarbeitung) i.V.m. § 23 UrhG (Bearbeitung) in unfreier Weise nutzt. Der dazu prozessual zu formulierende Unterlassungsantrag ist bereits dann hinreichend bestimmt, wenn diejenige Programmversion, die die Rechtsverletzung enthält, hinreichend präzise bezeichnet ist. Dies gibt oftmals bei individuell entwickelten Programmen, die nicht frei gehandelt, sondern nur von

einem einzelnen Auftraggeber genutzt und laufend angepasst werden, Formulierungsschwierigkeiten auf, die es erforderlich machen können, das Verletzungsmuster schon im Rahmen der Antragsfassung genauer zu spezifizieren. Dieses Problem stellt sich vorliegend aber nicht, da das Programm der Beklagten – insofern von ihr nicht bestritten – grundsätzlich käuflich erwerblich ist. Der Kläger hat die Version, die die Rechtsverletzung enthalten soll, hinreichend genau bezeichnet, so dass in einem etwaigen Vollstreckungsverfahren geprüft werden könnte, ob die benannte Version (oder eine als kerngleich zu beurteilende Version) trotz Verbots weiter vertrieben worden ist. Damit sind die Anforderungen an eine hinreichende Bestimmtheit erfüllt.

Der weitere Umstand, dass der materiell-rechtlich überhaupt in Betracht kommende Unterlassungsanspruch des Klägers ebenfalls nur ein begrenzter sein kann, steht der ausreichenden Bestimmtheit des Unterlassungstenors nicht entgegen. Zwar ist denkbar, dass nach dem Erlass eines Verbots die Beklagte ihr Programm ändern und sodann geltend machen könnte, die neue Programmversion falle nicht mehr unter das titulierte Verbot. Dies ist jedoch keine Frage der Bestimmtheit des Verbotstenors, sondern eine Frage der Reichweite nach der Kerntheorie. Diese besagt, dass von dem titulierten Verbot auch solche Verletzungshandlungen erfasst werden, die das charakteristisch rechtsverletzende Moment derjenigen Verletzungshandlung aufweisen, die im Erkenntnisverfahren vor dem Erlass des Verbots geprüft worden ist und Grundlage des Erlasses war. Welches das insofern charakteristische Moment ist, kann und muss ggf. durch Auslegung ermittelt werden; hierzu kann und muss ggf. auf die Ausführungen in den Entscheidungsgründen zurückgegriffen werden.

Das Risiko eines Klägers, der den materiellen Kern des verfolgten Unterlassungsanspruchs nicht hinreichend spezifiziert vorträgt, mag sein, dass das Gericht anhand des nicht ausreichend substantiierten Vortrags eine Verletzungshandlung nicht feststellen kann (dann ist die Klage zulässig, aber unbegründet) oder von möglicherweise mehreren Verletzungshandlungen nur eine feststellen kann (dann ist die Klage zulässig und begründet, der Kern des Verbots ist aber eng, denn er orientiert sich allein an der einen feststellbaren Verletzungshandlung).

Ob der Kläger zusätzlich den Unterlassungsantrag von vornherein auf einen bereits im Antrag spezifizierten rechtsverletzenden Aspekt begrenzt und damit den Streitgegenstand möglicherweise gegenüber einer weiter reichenden Begründung beschränkt, bleibt seiner Parteidisposition überlassen. Ob aus Gründen der Klarheit des Urteilsausspruches auch dann, wenn eine solche Beschränkung nicht in den Antrag aufgenommen ist, sie doch im Tenor der Entscheidung zur genaueren Umschreibung des Verbot aufgenommen werden sollte, ist im Einzelfall vom Gericht zu prüfen.

Dies alles ändert aber nichts daran, dass vorliegend das verlangte Verbot vom Kläger durch Bezeichnung des Programms der Klägerin hinreichend bestimmt worden ist.

B.

Die Klage ist jedoch unbegründet. Dem Kläger steht der mit dem Klageantrag zu 1. geltend gemachte Unterlassungsanspruch nach § 97 I i.V.m. § 69c Nr. 2 UrhG nicht zu. Deshalb ist auch der mit dem Antrag zu 2 geltend gemachte Anspruch auf Erstattung von Abmahnkosten zzgl. Zinsen nach § 97a UrhG, §§ 288, 291 ZPO nicht gegeben.

I.

Wie bereits zur Zulässigkeit ausgeführt, kommen vorliegend als Grundlage der Rechtsposition des Klägers lediglich Bearbeiterurheberrechte in Betracht, da der Kläger geltend macht, sukzessiv

eigenständige Beiträge zum Linux-Kernel beigetragen zu haben. Dass auch an einer Umarbeitung von Software ein Bearbeiterurheberrecht entstehen kann, folgt – wie gesagt – aus § 69c Nr. 2 S. 2 i.V.m. § 3 UrhG. Dies ist im Ausgangspunkt zwischen den Parteien des Rechtsstreits auch rechtlich nicht strittig.

Unter der Prämisse, der Kläger habe an Teilen des Linux-Kernels entsprechende Bearbeiterurheberrechte erworben, wäre die Rechtsstellung des Klägers jedoch – wie ebenfalls ausgeführt – eine begrenzte. Im Rahmen der Begründetheitsprüfung hat das Gericht daher zu prüfen, ob der Kläger hinreichend substantiiert dargelegt und ggf. bewiesen hat,

- welche Teile aus dem Linux-Programm von ihm in welcher Weise umgearbeitet worden sein sollen,

- inwiefern diese Umarbeitungen die Anforderungen an ein Bearbeiterurheberrecht nach § 69c Nr. 2 S. 2 i.V.m. § 3 UrhG erfüllen und

- inwiefern der Kläger darlegt und ggf. beweist, dass gerade die für ihn schutzbegründenden umgearbeiteten Programmteile ihrerseits von der Beklagten (ggf. weiter umgearbeitet) übernommen worden sind.

Nur wenn sich eine solche Übernahme von für den Kläger geschützten Code-Teilen feststellen lässt, müssten sich daran weitere Prüfungsschritte anschließen, insbesondere:

- Können „vmklinux" und „vmkernel" als ein einheitliches Werk angesehen werden (und nach welchen rechtlichen Kriterien ist diese Einheitlichkeit zu bestimmen)? Falls nämlich „vmklinux" als ein von „vmkernel" getrennt zu betrachtendes Werk sollte angesehen werden müssen, so besteht zwischen den Parteien Einigkeit, dass die Beklagte zur Einhaltung der Anforderungen der GPL-2.0 nur zur Offenlegung des „vmklinux"-Codes verpflichtet war und diese Bedingung eingehalten hat.

- Wenn „vmklinux" und „vmkernel" als ein einheitliches Werk angesehen werden könnten, so wäre schließlich zu prüfen, welche Bedeutung die in „vmklinux" übernommenen Code-Teile des Klägers für die dann anzunehmende Einheit „vmkernel+vmklinux" haben und ob unter dieser Perspektive diese Übernahme sich noch als Umarbeitung bzw. Bearbeitung i.S.v. § 69 Nr. 2 i.V.m. § 23 UrhG darstellt oder bereits als freie Benutzung i.S.v. § 24 UrhG (und nach welchen rechtlichen Kriterien dies zu bestimmen wäre). Dabei würde insbesondere zu beachten sein, dass die Rechtsposition des Klägers nur diejenige eines Bearbeiterurhebers wäre und dass die Beklagte in erheblicher Weise geltend gemacht hat, dass der mengenmäßige Umfang des allenfalls übernommenen Codes des Klägers in Relation zum Gesamtumfang „vmkernel+vmklinux" äußerst gering ist, ohne dass sich dem klägerischen Vortrag entnehmen lässt, dass die von der Beklagten geltend gemachten Größenordnungen völlig unzutreffend wären.

Indessen können und müssen diese (wohl im Zentrum des rechtlichen Interesses der Parteien und Parteivertreter stehenden) Fragen offen bleiben. Denn bereits die erste Prüfungsvoraussetzung, dass überhaupt für den Kläger als Bearbeiterurheber geschützter Code in im Produkt der Beklagten übernommen worden ist, lässt sich nicht feststellen. Das gilt auch unter Berücksichtigung des nachgelassenen Schriftsatzes des Klägers vom 29.04.2016, mit dem er (unter Fristverlängerung) nochmals Gelegenheit hatte, zu den vom Gericht bereits in der mündlichen Verhandlung geäußerten diesbezüglichen Bedenken vorzutragen.

1.

Die **allgemeinen Ausführungen des Klägers zu seiner Mitarbeit am Linux-Kernel** und der Möglichkeit, diese auch in öffentlich zugänglichen Quellen zu recherchieren, genügen den prozessualen Anforderungen an die Darlegungslast des Klägers nicht.

a)

So hat der Kläger darauf verwiesen, in einem sog. **Git-Repository** sei für jedermann öffentlich einsehbar, welche Teile des Linux-Codes von ihm, dem Kläger, stammten; ferner (auf S. 15 der Klageschrift), könne in den Repositories des Mainline-Kernels (d.h. der offiziellen Kernelversion) könne jeder Entwicklungsbeitrag des Klägers detailliert nachvollzogen und belegt werden. Ein solcher pauschaler Verweis auf eine Ermittlungsmöglichkeit von Tatsachenvortrag im Internet ist kein zulässiger prozessualer Vortrag.

b)

Auch die Vorlage des angeblich vollständigen **Linux-Sourcecodes** der Version 2.6.12-rc2 mit Versionsgeschichte (Ss 25.09.2015 S. 10) genügt den Anforderungen an einen prozessual nachvollziehbaren Vortrag zu vorliegend relevanten Bearbeiter-Urheber-Anteilen des Klägers nicht; denn es ist nicht Aufgabe des Gerichts oder des Gegners, sich aus einem Gesamtsourcecode etwa vom Kläger stammende Code-Teile herauszusuchen und selbst zu beurteilen, inwieweit der Kläger für welche Teil und Zusammenhänge Urheberrechtsschutz begehren könnte.

Der pauschale Verweis des Klägers, auf der CD-Rom **K 12** seien diejenigen Teile seines Codes enthalten, die sich in „vmklinux" wiederfänden (Ss 25.09.2015 S. 10), genügt auch nicht zur Darlegung der Rechtsverletzung. Der Kläger müsste die Übernahmen konkret benennen. Dies wäre ihm auch möglich, denn unstreitig ist der Sourcecode zum Modul „vmklinux" von der Beklagten öffentlich gemacht. Nicht ausreichend ist in diesem Zusammenhang auch die Vorlage der Datei **„linux.tgz"** auf der CD-Rom K 12 mit einem „Archiv"; denn auch hier müssten Gericht und Gegner wiederum selbst ermitteln, welche Anteile vom Kläger stammen sollen.

Soweit der Kläger geltend macht, die **Linux-Kernel-Version 2.6.18** sei diejenige, die „der von der Beklagten verwendeten Version am ähnlichsten" sei (SS 25.09.2015 S. 10), so ist auch dies kein schlüssiger Vortrag zu einer Übernahme von Code-Bestandteilen des Klägers, denn weder wird individualisiert, welche Bestandteile es sein sollen, noch wird dargelegt, wo im Code der Beklagten solche Teile auftauchen; als von der Beklagten „verwendete" Version, ist hier ersichtlich eine Linux-Version gemeint, nicht eine Version von „vmklinux". Der Vortrag wird auch nicht im Schriftsatz vom 29.04.2016 S. 10 oben (Bl. 274 d.A.), wo sich der Kläger auf die Ausführungen in der Klageschrift bezieht, konkretisiert.

Wenn der Kläger zusammenfassend geltend macht, er habe mit K 12 den von ihm zu Linux beigetragenen Code vorgelegt, der sich in „vmklinux" wiederfinde, so genügt es nicht zur Darlegung seines Bearbeiter-Urheberrechts an Linux den Code dieses Programms insgesamt vorzulegen und lediglich geltend zu machen, darin seien ja die Bearbeitungsanteile mit enthalten, die vom Kläger stammten; das ist kein für Gericht und Gegner nachvollziehbarer Vortrag, an welchen Bestandteilen gerade Bearbeiter-Rechte des Klägers geltend gemacht werden sollen.

c)

Soweit der Kläger schriftsätzlich behauptet hat, die mit CD-Rom **K 12** vorgelegten **„.blame"-Dateien** (bzw. die dazu inhaltsgleichen PDFs) zeigten, welcher Code vom Kläger stamme und welche

Änderungen die Beklagte in „vmklinux" vorgenommen habe, so ist das so nicht zutreffend und nachvollziehbar, wie sich bereits aus der vorgerichtlichen klägerischen e-mail K 10 selbst erschließt.

Auf der CD-Rom K 12 im dortigen Ordner „E-Mail vom 14.11.2014" sind allerdings mehrere PDF-Dateien enthalten, die laut K 10 den ebenfalls beigefügten „.blame"-Dateien entsprechen sollen. In den „.blame"-Dateien sei angegeben, welche Linux-Code-Teile vom Kläger stammten; letzteres scheint zu stimmen, denn wenn man die PDFs öffnet, sind dort Listen mit Code-Zeilen aufgeführt, in denen zu jeder Code-Zeile der jeweilige Autor genannt wird. Dies kann aber nur so verstanden werden, dass es sich um die Bearbeiter-Urheberschaft der verschiedenen Autoren (einschließlich des mit aufgeführten Klägers) innerhalb von Linux selbst handelt. Ein Vergleich mit dem „vmklinux"-Code aus dem Programm der Klägerin wird in diesen Listen jedenfalls nicht vorgenommen.

Ein solcher Vergleich findet sich stattdessen in den in der **e-mail K 10** ebenfalls genannten html-Dateien. Zu ihnen heißt es in K 10, diese Dateien gäben das Ergebnis eines Vergleichs der „Linux-Datei (Version 2.6.18) mit der Datei ihrer Mandantin" wieder; gemeint sind offenbar ein Vergleich des Programmes Linux mit dem Programm der Beklagten. Weiter heißt es in K 10 wörtlich: *„Die verglichenen Dateien ergeben sich aus dem Dateinamen. Sie finden in den Dateien schwarz, grün und rot markierten Code. Der schwarze Code ist solcher, der völlig unmodifiziert sowohl in der ursprünglichen Linux-Version, als auch der modifizierten Version Ihrer Mandantin enthalten ist. Der von Ihrer Mandantin hinzugefügte Code ist grün markiert, während der rot markierte Code solcher Linux Code ist, der in der Version Ihrer Mandantin nicht mehr vorhanden ist. "scsi_error.c" und "scsi_proc.c" zeigen erhebliche Übereinstimmungen. Sie erscheinen in den anderen Dateien geringer, da Code aus mehreren Linux-Dateien in eine Datei gemergt wurde."* Das ist insofern nachvollziehbar, als sich tatsächlich in den html-Dateien schwarz, grün und rot markierte Code-Zeilen finden. Als übernommen macht die Klägerin schwarzen und (mit Veränderungen) grünen Code geltend. Die html-Dateien weisen aber nicht aus, von welchem Linux-Bearbeiter der angezeigte schwarze bzw. grün markierte Code in Linux stammt. Insbesondere wird nicht angezeigt, welche Teile des in den hmtl-Dateien schwarz und grün markierten Codes vom Kläger stammen sollen. Dies müsste sich der Leser also durch einen Vergleich der schwarzen und grünen Teile der html-Dateien mit den PDFs der .blame-Dateien erst selbst erschließen. Jedenfalls dies ist kein prozessual nachvollziehbarer Vortrag mehr, zumal sich aus dieser Art der Darstellung nicht einmal eine schlüssige Behauptung ergibt, es seien überhaupt bestimmte Code-Zeilen des Klägers in „vmklinux" enthalten.

d)

Soweit der Kläger weiter behauptet, es sei ein vollständiger, **beispielhaft aus K 15 nachvollziehbarer Sourcecode-Vergleich** durchgeführt worden, so kann das Gericht lediglich die mit K 15 vorgelegten Beispiele beurteilen; darüber hinaus bleibt der Vortrag zu Übernahmen schon deshalb unsubstantiiert, weil der Kläger das Ergebnis des angeblichen Vergleichs nicht vorlegt und dazu nicht vorträgt.

Doch auch bzgl. der mit K 15 geltend gemachten Code-Anteile hat der Kläger kein verletztes Bearbeiterurheberrecht zu belegen vermocht. Die Beklagte hat überzeugend eingewandt, aus K 15 ließen sich letztlich nur acht spezifische und abtrennbare Funktionen des SCSI-Subsystems erkennen; dem ist der Kläger nicht entgegen getreten.

Die Beklagte hat weiter eingewandt, eine dieser acht Funktionen sei innerhalb von „vmklinux" „auskommentiert" worden, d.h. sie werde in „vmklinux" nicht verwendet; und bei den verbleibenden sieben Funktionen sei über die Hälfte der Codezeilen entweder von einem anderen Autor geschrieben (ohne jeglichen Beitrag des Klägers) oder vom Kläger höchstens einfach überarbeitet oder verschoben worden (Ss 05.02.2016 S. 14 = Bl. 203).

Der Kläger hätte insofern seine Anteile an den mit K 15 geltend gemachten Funktionen genauer spezifizieren und für seine Bearbeiterurheberschaft Beweis antreten müssen. Das hat der Kläger nicht in ausreichender Weise getan:

- In seinem unmittelbaren schriftsätzlichen Vortrag zu K 15 (Ss 25.09.2015 S. 11 = Bl. 155 d.A.) findet sich insofern kein Beweisantritt.

- K 15 selbst ist kein Ausdruck eines Code-Abschnitts und auch keine Veröffentlichung einer Bearbeitungshistorie, sondern eine vermutlich wohl vom Kläger persönlich zusammengestellte Übersicht über die einzelnen dort geltend gemachten acht Funktionen; daher kommt der Anlage K 15 selbst keinerlei Beweiswert bzgl. der Urheberschaft des Klägers zu.

- Soweit der Kläger auf die generelle Möglichkeit verwiesen hat, die Bearbeitungshistorie von Linux im Internet nachvollziehen zu können, ist dieser Vortrag nicht hinreichend spezifiziert bzgl. der Fragen, wo genau die Bearbeitungshistorie der acht spezifischen Funktionen veröffentlicht sei und welche Ergebnisse dort bezüglich der konkreten Anteile des Klägers an den acht Funktionen mitgeteilt werden.

- Das gilt entsprechend aber auch für die vom Kläger mit K 12 vorgelegten Dateien, die Versionshistorien enthalten sollen (history.tgz; linux.tgz). Da der Kläger somit nicht in ausreichend nachvollziehbarer Form konkrete Aussagen dieser Dateien über die spezifischen acht Funktionen geltend macht, kann deren Indizwert für eine Urheberschaft des Klägers schon nicht beurteilt werden.

- Selbst wenn in den Versionsgeschichten jedoch der Kläger als letzter Bearbeiter benannt werden würde, hat die Beklagte zutreffend geltend gemacht, dass Versionshistorien keine Vervielfältigungsstücke des betreffenden Computerprogramms sind, so dass von ihnen keine Vermutungswirkungen nach § 10 UrhG ausgehen können. Der dann allenfalls noch anzunehmende Indizwert müsste aber vom Kläger konkretisiert und auf die jeweiligen Funktionen bezogen dargelegt werden. Das hat der Kläger nicht getan.

e)

Soweit der Kläger in der Klage (S. 14) geltend macht, in zwei Dateien aus „vmklinux", zu denen die Beklagte den Quellecode offen gelegt habe, fänden sich sog. **Header mit Hinweisen auf Urheberrechte** u.a. des Klägers, so ist diesem Vortrag allein nicht zu entnehmen, dass (bearbeiter-) urheberrechtlich gesondert schutzfähige Teile aus dem Code des Klägers übernommen worden sein sollen. Dies kann nicht schon allein deswegen angenommen werden, weil die Beklagte selbst den Kläger als einen möglichen Bearbeiter von Linux genannt hat, sondern muss vom Kläger dargelegt werden.

2.

Die Ausführungen des Klägers zu seiner behaupteten **Mitarbeit am sog. SCSI-Subsystem** erlauben ebenfalls nicht die Feststellung einer Verletzung von Bearbeiterurheberrechten des Klägers.

a)

Wenn der Kläger vorgetragen hat, er habe Anzeichen dafür, dass ein Teil der SCSI-Funktionalität sich auch **im „vmkernel"** befinde, so hat er gleichzeitig selbst eingeräumt, er müsse jedoch mangels anderer Anhaltspunkte davon ausgehen, dass es sich insoweit um Eigenentwicklungen der Beklagten

handele; damit hat der Kläger hier nach seinem eigenen Vortrag ausdrücklich keine Übernahme eigenen Linux-Codes geltend gemacht.

Daher kann sich – wie bereits ausgeführt – auch insofern die Prüfung auf die Frage der Übernahmen in „vmklinux" beschränken.

b)

Soweit der Kläger zunächst **allgemein** behauptet, er sei in der Zeit von 2000-2004 einer der aktivsten Entwickler am SCSI-Subsystem des Linux-Kernels gewesen, das sicherlich eine komplexe und damit schutzfähige Programmierung darstelle (Ss 25.09.2016 S. 8 = Bl. 152 d.A.), so genügt diese generelle Behauptung nicht, urheberrechtlich schutzfähige Leistungen des Klägers darzulegen.

Der Kläger kann sich für die Schutzfähigkeit seiner Beiträge nicht auf die Komplexität des SCSI-Subsystems insgesamt berufen, denn der Kläger räumt ein, auch dieses nur (mit-) überarbeitet zu haben; da er also auch hier nur ein Bearbeiterurheberrecht geltend macht, muss er diejenigen Programmierleistungen, für die er Schutz beansprucht, konkret benennen.

Das Angebot, andere Kernel-Entwickler zu Beiträgen des Klägers zum Linux-Kernel zu hören, ist auf Ausforschung gerichtet und stellt keinen nachvollziehbaren Vortrag zu diesen Anteilen dar.

c)

Soweit der Kläger insbesondere behauptet, das **„SCSI-Hotplug"** sei eine der wichtigsten Funktionen, die ESXi benutze (Ss 25.09.2015 S. 8 = Bl. 152 d.A.), so kann offen bleiben, ob der Kläger mit den PDFs zu **19 sog. Patches** auf der CD-Rom **K 12** ausreichend zu einer Bearbeiter-Urheberleistung vorgetragen hat. Nicht ausreichend ist jedenfalls sein Vortrag zu einer etwaigen Übernahme seiner Code-Anteile in „vmklinux". Die Beklagte hat eine Übernahme relevanten Codes des Klägers im Zusammenhang mit der Hotplug-Funktionalität ausdrücklich bestritten (Ss 05.02.2016, S. 11 = Bl. 200 d.A.).

Im klägerischen Schriftsatz vom 25.09.2015 heißt es dazu lediglich auf S. 8, das ESXi der Beklagten benutze die „Funktion" des „SCSI-Hotplug". Das ist nur die Behauptung einer entsprechenden Funktionalität, nicht aber die Behauptung, dass dieser Funktionalität auch identischer Programmcode zugrunde liege. Erst recht nicht lässt sich diesem Vortrag entnehmen, dass in ESXi, insbesondere in „vmklinux", solcher Programmcode enthalten sei, der gerade die aus der CD-Rom K 12 Ordner „Patches" ersichtlichen 19 Beiträge des Klägers enthalte, für die er Bearbeiter-Urheberrechtsschutz beansprucht, insbesondere nicht in einer gerade bzgl. dieser Beiträge unveränderten Form. Eine solche Behauptung wäre aber zur schlüssigen Darlegung einer Verletzung gerade des Bearbeiter-Urheberrechts des Klägers erforderlich gewesen. Dass auch insofern die Vorlage der Anlage K 15 nicht genügt, ergibt sich aus den Erwägungen oben 1.d) entsprechend.

Auch der Vortrag des Klägers (Ss 29.04.2016 S. 11) zum **Patch „scsi_scan.c restructuring for ieee1394 hotplugging"** (= Patch Nr. 11 im Ss 25.09.2015 S. 9 = Bl. 153 d.A.) genügt nicht. Zum einen wird nicht klar, inwiefern es sich bei diesem einzelnen Patch um eine komplexe, für eine Individualität sprechende Programmierung handelt. Jedenfalls aber wird auch hier nicht dargelegt, dass gerade die Programmierung des Klägers in „vmklinux" übernommen worden sei; vielmehr wird vom Aufruf zweier einzelner Funktionen „SCSI-add_device" und „SCSI-remove_device" gesprochen, an denen die Übernahme solle erkannt werden können. Das genügt nicht. Der Kläger hätte konkret darlegen müssen, welcher Code in „vmklinux" von ihm stammen soll.

Soweit der Kläger im Schriftsatz vom 25.09.2015 S. 6-7 unter II.4.a)-c) Ausführungen dazu macht, dass einzelne dort erörterte weitere einzelne Code-Zeilen zwar von anderen Bearbeitern stammten,

jedoch an seiner urheberrechtlich geschützten Leistung nichts änderten, bleibt der Vortrag zur eigenen Leistung des Klägers nicht nachvollziehbar. Im Schriftsatz unter a) macht er geltend, den Beitrag **(Patch) „fixes and cleanups [usw.]"** (= Patch Nr. 10 auf S. 9 a.a.O.) geschaffen zu haben, ohne diesen Beitrag näher dazulegen. Im Schriftsatz unter b) räumt der Kläger ein, an der betreffenden Stelle lediglich fremden Code bearbeitet zu haben, um eine andere Schnittstelle benutzen zu können, was urheberrechtlich relevant sei; weder wird der von ihm bearbeitete Code vorgelegt, noch wird erläutert, inwiefern die Anpassung an eine Schnittstelle über eine Durchschnittsprogrammierleistung hinaus gehe. Unter c) macht der Kläger geltend, den **Beitrag (Patch) „scsi_device refcounting [usw.]"** (= Patch Nr. 19 auf S. 9 a.a.O.) geschaffen zu haben; auch zu dessen urheberrechtlicher Schutzfähigkeit finden sich keine Erläuterungen. Nur vorsorglich sei außerdem darauf hingewiesen, dass der Kläger weder an dieser Stelle noch später ausdrücklich geltend macht, dass gerade die hier erörterten Code-Teile auch in „vmklinux" der Beklagten enthalten seien.

d)

Auch der Vortrag des Klägers zu sog. **Midlayer-Code** innerhalb der SCSI-Funktionalität (Ss 29.04.2016 S. 7) vermag Rechte des Klägers nicht darzulegen, denn der Kläger behauptet nicht, dass dieser sog. Midlayer-Code von ihm stamme.

Entsprechendes gilt zu den Ausführungen (a.a.O.) zu SCSI-Funktionalität in an vmklinux angebundenen **Hardwaretreibern**.

e)

Soweit der Kläger schließlich zu einer scsi-Funktion **„scsi_remove_single_device"** vorgetragen hat, unterscheidet sich dieser Vortrag von den bisher erörterten Darlegungen des Klägers zum scsi-Subsystem dadurch, dass der Kläger hier die Herkunft konkreten Codes und dessen Übernahme in „vmklinux" nachvollziehbar behauptet, nämlich durch Angaben konkreter Dateifundstellen (im Linux Kernel 2.5.64 in der Datei „drivers/scsi/scsi_proc.c" in den Zeilen 423 ff. (K 25); bei VWware ESXi 5.5 Update 2 in der Datei „vmdrivers/src_92/vmklinux_92/linux/scsi/scsi_proc.c" in den Zeilen 250 ff. (K 26); vgl. (Ss 29.04.2016 S. 12-13); dies belegt, dass dem Kläger ein solcher Vortrag möglich ist.

Nicht ausreichend ist jedoch der Vortrag des Klägers zur urheberrechtlichen Schutzfähigkeit des als von ihm stammend behaupteten Linux-scsi-Codes. Der Kläger trägt zur Funktion weder als solcher noch innerhalb des scsi-Subsystems oder gar innerhalb von Linux vor und ebenso wenig zu der Funktion innerhalb von „vmklinux". Inwiefern sich die Programmierung oder Überarbeitung von einer rein handwerklichen Programmierung abheben oder auch nur für sich genommen eine ausreichende Komplexität aufweisen soll, wird vom Kläger nicht vorgetragen. Der von ihm angebotene Sachverständigenbeweis ist auf Ausforschung gerichtet.

3.

Auch soweit der Kläger geltend macht, er habe bei der **Entwicklung der sog. Radix-Trees** mitgearbeitet, so ist schon nicht schlüssig vorgetragen, dass es sich insofern überhaupt um ein Computerprogramm i.S.v. § 69a I UrhG handelt. Jedenfalls ist zu für den Kläger schutzfähigen und von der Beklagten übernommenen etwaigen Radix-Tree-Code-Teilen nicht nachvollziehbar vorgetragen.

a)

Das Verständnis des Begriffs des Computerprogramms in § 69a UrhG ist richtlinienkonform auszulegen. Der EuGH hat zur Auslegung der Computerprogramm-Richtlinie 91/250 entschieden (EuGH, Urteil vom 02. Mai 2012 – C-406/10 – = GRUR 2012, 814 – SAS – zit. nach juris-Rz. 46), dass

ihr Art. 1 Abs. 2 dahin auszulegen ist, dass weder die Funktionalität eines Computerprogramms noch die Programmiersprache oder das Dateiformat, die im Rahmen eines Computerprogramms verwendet werden, um bestimmte Funktionen des Programms zu nutzen, eine Ausdrucksform dieses Programms sind und daher nicht unter den Schutz des Urheberrechts an Computerprogrammen im Sinne dieser Richtlinie fallen. Diesem Ausspruch vorausgehend findet sich folgende Differenzierung des EuGH (zit. nach juris-Rz. 42 und 43):

> *42. Was die Programmiersprache und das Dateiformat anbelangt, die im Rahmen eines Computerprogramms verwendet werden, um die von den Nutzern geschriebenen Anwendungsprogramme zu interpretieren und auszuführen bzw. um Daten in einem besonderen Dateiformat zu lesen und zu schreiben, so handelt es sich um Elemente dieses Programms, mittels deren die Nutzer bestimmte Funktionen des Programms nutzen.*
>
> *43. Würde sich ein Dritter den Teil des Quell- oder Objektcodes beschaffen, der sich auf die Programmiersprache oder das Dateiformat bezieht, die im Rahmen eines Computerprogramms verwendet werden, und würde er mit Hilfe dieses Codes in seinem eigenen Computerprogramm ähnliche Komponenten erstellen, könnte es sich bei diesem Verhalten möglicherweise um eine teilweise Vervielfältigung im Sinne des Art. 4 Buchst. a der Richtlinie 91/250 handeln.*

Diese Differenzierung lässt sich insofern auf die Frage des Schutzes von Datei-Baum-Strukturen übertragen, als diese Strukturen als solches ebenfalls nur Funktionen des Programmes sind, jedoch keine Ausdrucksform des Programmes. Insofern genügt der bloße Verweis des Klägers auf Baumstrukturen nicht, um ein eigenes Bearbeiterurheberrecht oder dessen Verletzung zu begründen.

In der Klageschrift spricht der Kläger aber nur davon, es handele sich bei Radix-Trees um „eine skalierbare Implementierung einer speziellen Baumstruktur, die insbesondere in der Verwaltung von Dateisystemen-Puffern (Caches), aber auch in vielen anderen Anwendungsbereichen benutzt werde". Daraus wird nicht ersichtlich, dass es sich um einen konkreten Programm-Code handelt, der selbst ausführbar ist oder ein urhberrechtsschutzfähiger Teil eines ausführbaren Programmcodes darstellt.

Vielmehr lässt sich der Vortrag des Klägers nur dahin verstehen, es gehe um eine „Struktur", also die Art und Weise einer Anordnungsmöglichkeit. Selbst wenn sich diese Struktur darauf beziehen sollte, in welcher Weise (Reihenfolge) bestimmter Programmcode (als bestimmte Befehle) am sinnvollsten, effektivsten, Speicherplatz sparend, am schnellsten ausführbar o.ä. anzuordnen sind, so ginge es dann doch nur um ein generelles Gestaltungsprinzip für Computerprogramme, nicht aber um ein Programm als solches. Ein bloßes Gestaltungsprinzip, welches – wie der Kläger selbst geltend macht – in unterschiedlichsten Zusammenhängen anwendbar ist, ist als solches aber nicht urheberrechtlich schutzfähig.

b)

Zu einer Übernahme etwaigen Codes im Zusammenhang mit „Radix-Trees" ist vom Kläger auch im Übrigen nicht schlüssig vorgetragen worden.

Soweit es dazu in der Klageschrift auf S. 15 heißt, eine von ihm durchgeführte Analyse habe ergeben, dass im Quellcode der Beklagten – so wörtlich – *„die ‚Radix-Tree'-Funktionalität verwendet"* worden sei und dafür zum Beweis angeboten wird „Im Bestreitensfall Vorlage von Quellcode-Dateien", so ist der Vortrag zum einen nicht nachvollziehbar, dass *überhaupt* konkrete Codezeilen übernommen worden sein sollen („Funktionalität"), zum anderen welche Code-Zeilen dies sein sollen und

schließlich ist das Beweisangebot nicht nur auf Ausforschung gerichtet, sondern auch selbst völlig unbestimmt.

Soweit der Kläger sodann im Schriftsatz vom 29.04.2016 S. 13-15 Ausführungen zu „Radix Trees" macht, spricht zwar die Vorlage der Anlage K 31 mit Code aus „vmklinux" der Beklagten dafür, dass es sich bei Radix-Trees nicht allein um eine Struktur, sondern um konkrete Programmzeilen handeln mag. Der Kläger legt aber nicht dar, welche Teile aus K 31 (dem Verletzungsmuster) denn aus Linux übernommen und welche davon wiederum von ihm geschaffen worden seien (Klagemuster). Vielmehr räumt der Kläger ein, die Struktur nicht erfunden zu haben; lediglich an der „Implementation" habe er neben dem Erfinder Velikov mitgewirkt. Damit ist aber derjenige Anteil des Codes, den der Kläger selbst geschaffen haben will, aus dem Vortrag des Klägers nicht zu entnehmen.

Auch K 30 gibt insofern keinen Aufschluss. Soweit deren S. 1 darauf hinzudeuten scheint, dass Teile des Code vom Kläger stammen könnten, so bleibt unklar, auf welche konkreten Zeilen sich das bezieht und inwiefern gerade dieses Zeilen auch in „vmklinux" verwendet worden sein sollen. Das gilt umso mehr, als der Kläger selbst vorträgt, die Beklagt habe gar nicht die aus K 30 ersichtliche Version von 2002, sondern eine spätere Version von 2012 zur Grundlage von „vmklinux" genommen; welche Teile der Radix Trees-Version aus 2012 aber vom Kläger stammen sollen, lässt sich dem Vortrag des Klägers ebenfalls nicht entnehmen. Insofern ist der vom Kläger angebotene Sachverständigenbeweis auf Ausforschung gerichtet.

c)

Das Angebot, andere Kernel-Entwickler zu Beiträgen des Klägers zu den Radix-Trees zu hören, ist ebenfalls auf Ausforschung gerichtet und stellt keinen nachvollziehbaren Vortrag zu diesen Beiträgen und zur Frage der Schutzfähigkeit der Radix-Trees dar.

4.

Soweit der Klägerin in der Klageschrift S. 15 weiter behauptet, seine Analyse habe ergeben, dass **viele andere Teile des Linux-Betriebssystems von der Beklagten verwendet** würden, zu denen er kleinere Bearbeitungen des Quellcodes beigetragen habe, und auch hier „Vorlage von Quellcode-Dateien" als Beweis angeboten wird, so ist auch dieser Vortrag in mehrfacher Hinsicht nicht ausreichend, denn weder wird deutlich, um welche „Teile" aus Linux es gehen soll, welche Bearbeitungsleistung der Kläger insofern geleistet habe und welche gerade von ihm stammenden Code-Teile in „vmklinux" übernommen worden sein sollen.

C.

Die Kostenentscheidung folgt aus § 91 ZPO.

Die Vollstreckbarkeitsentscheidung folgt aus § 709 S. 1 und 2 ZPO.

Hartmann	Harders	Dr. Frantzen
Vorsitzender Richter am Landgericht	Richter am Landgericht	Richterin