Was ist ein NDA?

Linux bedeutet auch mehrere Dinge. Zum einen den Kernel, ursprünglich von Linus Torvalds geschrieben. Linux bezeichnet auch eine Familie von Betriebs-systemen. Wenn wir „Linux“ sagen, meinen wir Red Hat, Slackware, Debian, Gentoo und hunderte andere Distributionen um den Linux-Kernel herum, mit sehr ähnlichem Userland; die meist auf GNU Tools basieren.

Warum brauchen wir freie Dokumentation?

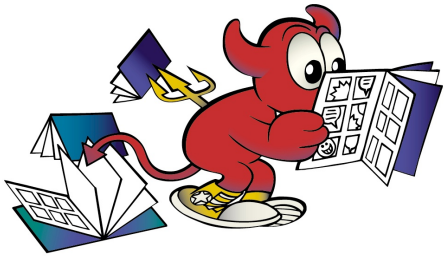
BSD steht für „Berkeley Software Distribution“. Es wurde an der University of California Berkeley (CSRG) entwickelt und der Code stand immer unter der BSD-Lizenz. Die Lizenz besagt mehr oder weniger „Tun Sie damit, was immer Sie wollen. Zollen Sie uns einfach Anerkennung dafür.“ Später startete das 386BSD-Projekt und ermöglichte es auf der Intel i386-Plattform zu laufen. Als das 386BSD-Projekt langsam zu Ende ging, formierten sich zwei Hauptgruppen: FreeBSD und NetBSD. 1995 spaltete sich OpenBSD von NetBSD und 2003 DragonFlyBSD von FreeBSD ab.

Welche Firmen verhalten sich wie?

Das Konzept einer „Systembasis“ kann eventuell schwer zu verstehen sein, denn dieses Konzept gibt es in der Linux-Welt nicht. Linux war von Anfang an immer nur ein Kernel, und ein Kernel allein ist nicht sehr nützlich. Man braucht alle Userland-Programme, um ihn zu nutzen. Linux war schon immer ein Konglomerat; ein Kernel von hier, ein 'ls' von dort, vim, Perl, gzip, tar und alle anderen auch. Linux unterschied nie zwischen einer Systembasis und den Userland-Programmen. Das *ganze System* besteht aus Userland-Programmen.

Wie können Sie uns helfen?

Dann gibt es noch die zweite Kategorie, die Programme, die Zusatzpakete sind. In der BSD-Welt sind das meist sogenannte „Ports“ (FreeBSD und OpenBSD) oder „pkgsrc“ (NetBSD und DragonFlyBSD).



Wieso sollte ich BSD versuchen?

Soll ich BSD oder Linux benutzen? Eine Frage, die nur schwer zu beantworten ist. Hier ein paar Richtlinien:

1. Es funktioniert einfach. Sobald das System steht, kann man loslegen.

2 „Wenn es nicht kaputt ist, soll man es nicht flicken.“ Wenn Sie schon ein Open Source OS benutzen, und Sie damit glücklich sind, so gibt es wahrscheinlich keinen Grund, wieso Sie wechseln sollten.

3. BSD Systeme sind merklich schneller, aber nicht immer. Oft gibt es keinen Unterschied. In manchen Fällen könnte Linux sogar schneller sein.

4. Die BSDs haben einen besseren Ruf bezüglich Zuverlässigkeit wegen der reiferen Code-Basis.

5. Die BSD-Lizenz mit weniger Einschränkungen könnte attraktiver sein, als die GPL.

6. BSD kann die meisten Linux Programme ausführen, aber Linux nicht die BSD Programme.

Wichtige Websites und Links:

Alle BSDs sind hervorragend dokumentiert, auf den Webseiten der einzelnen Projekte finden Sie nahezu jede gewünschte Information:

http://www.FreeBSD.org/

http://www.NetBSD.org/

http://www.OpenBSD.org/

http://www.DragonFlyBSD.org/

Deutsches BSD-Forum: http://www.BSDGroup.de/

Stop Blob! - Deutsch

Dieser Flyer soll Ihnen in verständlicher Form aufzeigen, warum wir in Ihrem Interesse für frei verfügbare Dokumention kämpfen.

Was ist ein Blob?

Open-Source-Betriebssysteme sind nicht gerade reich mit Gerätetreibern ausgestattet. Der eine oder andere Hardware-Hersteller erbarmt sich dann doch und entwickelt die für ein Betriebssystem wichtige Software. Aber leider liegt es anschließend meist als Closed-Source-Entwicklung vor. Die Problematiken, die hierdurch entstehen, untersucht der Verfasser des Artikels.

Um es gleich vorweg zu nehmen: es geht hier nicht um Spinat mit Sahne. Es geht auch nicht um Applikationen, die als binary-only angeboten werden. Es ist ein ernstes Thema, das hauptsächlich Open-Source-Betriebssysteme betrifft: Gerätetreiber! Um das Verständnis für die Problematik zu schärfen, werden zuvor einige Begriffe erläutert.

Den meisten Systemverwaltern sind Gerätetreiber ein Begriff und bedürfen daher keiner Erläuterung, ganz im Gegensatz zu den vielen normalen Benutzern und IT-Entscheidern, die für dieses Thematik auch sensibilisiert werden sollen.

Ein Gerätetreiber (*device driver*) ist ein Programm, das ein Gerät (*device*) für den Betriebssystemkern verfügbar macht. Der Kern kommuniziert mit Hilfe dieses Treibers mit der Hardware

Der Treiber wandelt Steuerbefehle in für das Gerät verständliche Befehle um. In der anderen Richtung werden

Statusmeldungen und Fehlermeldungen für den Betriebssystemkern aufbereitet. Daneben transportiert der Treiber die Daten. Die Programmierer von Gerätetreibern tragen somit eine sehr große Verantwortung bei der Entwicklung dieser Art von Software. Ein kleiner Fehler hat fatale Auswirkungen wie beispielsweise Datenverlust.

Im Zusammenhang mit Software wird auch häufig das Akronym „Blob“ verwendet. Der Begriff stammt aus dem Datenbankbereich und bedeutet „binary large object“ (deutsch: großes binäres Objekt). Es ist also ein Stück Software, das in (ausschließlich) binärer Form vorliegt oder veröffentlicht wird.

Im Umfeld der Systemsicherheit weiterhin oft der Begriff „backdoor“ (deutsch: Hintertür) benutzt. Eine Backdoor ist ein bewußt oder unbewußt (durch Fehler) programmierter Einsprungpunkt. Meist werden sie dazu benutzt, um die Kommunikation von Treiber und Daemon-Prozessen zu gewährleisten oder man verwendet sie für Debugmechanismen.

Closed-Source versus Open-Source versus NDA

Der Leser wird sich nun fragen, wo der Unterschied zwischen Closed-Source-Treibern und Open-Source-Treiber ist. Closed-Source-Treiber erhält der User immer nur in binärer Form. Der Vorteil der Methode ist, dass der Treiber nicht erst aufwendig kompiliert werden muß. Es reicht das Einlegen der Diskette oder CD aus und schon beginnt der Installationsprozess. Bei Open-Source-Treibern ist dies meist anders. Hier ist ein Compiler-Lauf notwendig, manchmal muß auch der gesamte Kernel kompiliert werden, oder schlimmstenfalls ist auch ein Neubau des gesamten Betriebssystems notwendig. Das schreckt viele Anwender ab.

Closed-Source-Treiber haben aber auch entscheidende Nachteile. Auf zwei dieser Nachteile möchte der Autor näher eingehen: Sicherheit und Backdoors.

Sicher! Sicher?

In einer Zeit, in der Sicherheit vor Angriffen aus dem Internet eine immer wichtigere Rolle spielt, sollte sich jeder User oder Administrator sich seine Gedanken gemacht haben, wie „sicher“ sein bevorzugtes Betriebssystem sein sollte oder könnte. Dabei spielen zwei Aspekte eine Rolle: die Sicherheit nach außen und die innere Sicherheit. Nach außen hin werden die Betriebssysteme mittels Paketfilter oder ähnlichem vor unerwünschten Zugriffen hinreichend geschützt. Die innere Sicherheit ist meist durch gute Virenscanner abgedeckt. Was ist aber mit Software, die durch einen Fehler den Datenbestand eines Servers gefährdet? Was ist, wenn ein Treiber für einen SCSI-Hostadapter oder SATA-Controller bei der Übertragung eines bestimmten Datenvolumens einen Fehler produziert? Manche Administratoren oder User werden jetzt denken, dann rufe ich meinen Händler an oder schaue auf der Internet-Seite des Hardware-Herstellers nach Neuerungen zur Software. Es gibt Hersteller, die nur in bestimmten Zyklen Fehlerkorrekturen zu ihrer Software liefern (prominentes Beispiel: Microsofts Patchday). Schwierigkeiten treten dann auf, wenn an so einem Patchday die Fehler im Treiber nicht behoben werden.

Mangels Quelltext hat ein des Programmierens kundiger Administrator keinerlei Möglichkeit, Analysen anzustellen, wo sich der Fehler befindet und diesen eventuell zu beheben. So blieben ihm zwei Möglichkeiten: beten und warten, bis der Hersteller einen Bugfix liefert. Bei älteren Geräten besteht dann durchaus der Zwang neue Hardware anzuschaffen, was auch mit nicht unerheblichen Primär- und Sekundärkosten verbunden ist. Unter Primärkosten fallen die Anschaffungskosten des neuen Geräts, die Sekundärkosten entstehen durch ausgiebige Testphasen, Systemausfälle während der Umbauphase und Personalkosten.

Systemverwalter und User müssen sich vor Augen halten, welches Gefahrenpotienzial in Treiber-Backdoors stecken kann. Angenommen ein Daemon-Prozess nutzt eine Treiber-Backdoor um den Netzwerkverkehr zu überwachen. Durch einen kleinen Fehler im Treiber kommt es dazu, dass der Damon-Prozess die Daten manipuliert oder falsch interpretiert. Dadurch können auch nicht unerhebliche Schäden – auch finanzieller Art! - auftreten

Natürlich treten an dieser Stelle jene Leute auf den Plan, die meinen: „Dafür hab ich doch schließlich...“ oder „Dann mach ich halt...“ und reden so das Problem klein. Sie zeigen dadurch, dass sie es nicht begriffen haben und durch diese Denke die Datensicherheit aller Beteiligten gefährden.

Securitas per definitionem?

Leider denken viele Hardware-Hersteller genau in diese Richtung: Eine Software ist nur dann sicher, wenn der Quelltext im Verborgenen bleibt. Dabei verfährt man nach dem Motto „Was ich nicht weiß, macht mich nicht heiß.“, und das im doppelten Wortsinn.

Zum einen gehen die Hardware-Hersteller davon aus, dass durch Offenlegung der Quelltexte zu den Gerätetreibern bestimmte interne Logikstrukturen der Geräte bekannt werden. Die darf der Mitbewerber nicht wissen. Da darf man froh sein, dass CPU-Hersteller den Befehlssatz und die interne Logik ihrer CPUs nicht geheimhalten. Das trifft auch auf die Entwickler des PCI-Buses zu. Wären die Spezifikationen (Bus-Timings, Trigger-Phasen oder Arbitrierungslogik) Closed-Source, dann gäbe es wahrscheinlich keine PCI-Steckkarten. Man denke dabei an IBMs Disaster mit dem Microchannel-Bus!

Andererseits wird argumentiert, sobald die Spezifikation eines Geräts offengelegt ist, würden durch falsche Programmierung der Register eventuell bestimmte Richtlinien nicht mehr eingehalten.

Wireless-LAN-Netzwerkkarten sind hierfür ein gutes Beispiel. Hier sträuben sich die meisten Hardware-Hersteller Dokumentationen freizugeben. Es wird hauptsächlich damit begründet, dass durch falsche Programmierung der PLL-Bausteine, die Sendefrequenz in einen unzulässigen Bereich verschoben wird. Das mag so stimmen, aber wer garantiert, dass der Entwickler des Closed-Source-Treibers den Steuerbaustein korrekt programmiert hat?

Es drängen sich Fragen auf. Wieso erschaffen die Hardware-Hersteller nicht eine Programmierschnittstelle (API), an die ein Open-Source-Treiber sauber andocken kann? Wieso muß die CPU immer mehr Aufgaben übernehmen, die prinzipiell von der Peripherie erledigt werden sollte (GDI-Drucker, Win-Modems)? Wieso wird die Firmware eines Peripherie-Geräts im Arbeitsspeicher des PCs gehalten und nicht in einem Flash-ROM des Geräts (Manche WLAN-Karten)?

Im Open-Source-Bereich haben viele Programmierer die Möglichkeit, das korrekte Funktionieren der Software zu prüfen. Es finden sich fast immer Leute, die auftretende Fehler binnen kürzester Zeit beheben. Gleiches gilt für die Optimierung von Treibern. Das Mehraugenprinzip ist dabei nicht zu unterschätzen.

Die Beweggründe für die Verweigerung der Hardware-Hersteller sind vielfältig. Meist ist es die Angst, die Konkurrenz würde Einblick in bestimmte Algorithmen erhalten. Dieses Argument ist verständlich und auch nicht von der Hand zu weisen. Dieses Problem läßt sich doch dadurch lösen, dass die Firmware als separate Datei vorliegt und diese muß vom Treiber in das Peripherie-Gerät geladen werden. Damit der Treiber mit der Firmware kommunizieren kann, ist das API genauestens zu dokumentieren. Die Hersteller geben so keine für sie wichtige Algorithmen Preis und die Entwicklergemeinde hat die Chance, Treiber zu entwickeln.

In der letzten Zeit wurde seitens einiger weniger Treiberentwickler aus dem Linux-Umfeld vorgeschlagen, dass

Treiberentwickler ein sogenanntes *non disclosure agreement* (NDA) mit den Hardware-Herstellern eingehen sollten. Dies würde gewährleisten, dass immer die neueste Dokumentation zur Entwicklung von Treibern bereitstehen würde. Ist das die Lösung des Problems? Ganz sicher nicht!

Die Open-Source-Gemeinde ist damit wieder den Herstellern ausgeliefert. Jeder Hardware-Produzent kann in seine NDA hineinschreiben, was er möchte. Die Wahrscheinlichkeit ist hoch, dass durch dieses Hintertürchen Closed-Source-Treiber Einzug in ein Open-Source-Betriebssystem halten. Ist das der Geist von Open-Source? Es bleibt weiterhin die Frage zu klären, ob ein NDA an eine Person oder an einem Entwicklerteam gebunden ist. Falls es nur an einer Person gebunden sein sollte, müssen sich die Befürworter von NDA-Abkommen für den Zeitpunkt Gedanken machen, wenn die Vertragsperson aussteigt! Außerdem darf dabei nicht vergessen werden, dass nur die Vertragsperson Einblick in die Dokumentation haben darf. Wie sollen nach dem Mehraugenprinzip andere Entwickler Fehler in der Software beheben dürfen? Wie werden unterschiedliche Lizenzmodelle (GPL versus BSD) berücksichtigt?

Kampagne von allBSD

OpenBSD-Chefentwickler Theo de Raadt hat im Rahmen einer Universitätsveranstaltung in den USA auf die Misere aufmerksam gemacht. allBSD hat sich daher entschlossen, mit einer Kampagne im europäischen Raum auf die Situation eindringlich hinzuweisen, dass viele Hardware-Hersteller nicht gewillt sind, Informationen zu ihren Geräten für Open-Source-Entwickler zugänglich zu machen.

Es zeigt das für alle BSD-Betriebssysteme gemeinsame Maskottchen *Beastie* mit einem großen Hammer, das symbolisch eine Diskette mit Closed-Source-Treibern einstampft. Damit möchte allBSD zeigen, dass Closed-Source-Treiber in Open-Source-Betriebssystemen nicht zu suchen haben.

Neben diesem Plakat informiert zusätzlich ein Flyer über das Anliegen von allBSD und seiner Kampagne.

Um es deutlich zu machen: Diese Kampagne möchte nicht diffamieren oder marktschreierisch auftreten, sondern sie soll bei alle Beteiligten durch Information zu einem Umdenken animieren. Daher folgender Aufruf an die Hardware-Hersteller, arbeiten Sie mit Open-Source-Entwicklern zusammen. Sie sollten immer folgenden Gedanken im Hinterkopf behalten, dass die Konkurrenz schon Open-Source-Treiber anbietet und dadurch mehr Umsatz und Gewinn macht. Außerdem hat sie deswegen ihren Marktanteil gesteigert! Eine gesunde Symbiose zwischen Open-Source-Entwicklern und Hardware-Herstellern ist von gegenseitigem Nutzen.