Fachhochschule Wedel
Seminararbeit

Fachrichtung
Wirtschaftsingenieurwesen

Open Source vs. kommerzielle Software

Erstellt von:	Rebekka-Isabelle Stahl (Mat-Nr. 101234) wing101234@fh-wedel.de
Erarbeitet im	6. Semester
Abgegeben am:	22.06.2017
Betreuender Dozent:	Prof. Dr. Michael Anders Fachhochschule Wedel Feldstraße 140 22880 Wedel Tel. (04103) 804824 E-Mail: an@fh-wedel.de

Inhaltsverzeichnis

1. Einleitung

Open Source Software spielt eine immer größer werdende Rolle in der Weiterentwicklung von Programmen, so laufen die meisten aktiven Websites heutzutage über den Open Source Webserver Apache.[1]

Auch in vielen anderen Bereichen gibt es mittlerweile sehr gute Open Source Alternativen, die in der folgenden Tabelle aufgeführt sind:

Software Art	Kommerzielle Software	Open Source Äquivalent
Betriebssystem	Microsoft Office	Linux Ubuntu
Browser	Microsoft Edge	Mozilla Firefox
Grafikprogramm	Adobe Photoshop	Gimp
Office Programm	Microsoft Office	Libre Office
Media Player	Windows Media Player	VLC
Webserver	IIS	Apache
Datenbank	Oracle	MySQL

Die größten Verfechter von Open Source Software haben das Ziel Software zu entwickeln, zu bearbeiten und zu verbreiten ohne dabei eingeschränkt zu werden.

Diese Arbeit beschäftigt sich mit den Vor-und Nachteilen der wirtschaftlich orientierten kommerziellen Software und der kostenlosen, nutzerorientierten Open Source Alternative. Dabei wird besonders auf die Sicherheit geschaut.

[1] (Anon., 2017)

2. Begriffserklärung

In diesem Abschnitt werden die Begriffe verschiedener Softwarearten voneinander abgegrenzt. Dazu werden Adware, Freeware, Shareware, Open Source Software und kommerzielle (proprietäre) Software grundlegend definiert. Dieser Überblick soll als Einstieg in den Vergleich zwischen Open Source und kommerzieller Software dienen.

Der größte Unterschied zwischen den Softwarearten bezieht sich auf den Zugang zum Quellcode. Dieser kann entweder frei einsehbar (Open Source) oder nicht einsehbar (Closed Source, kommerziell) sein. Der Quellcode (Quelltext) einer Software bezeichnet den in Programmiersprache verfassten lesbaren Text eines Programmes. Die meisten Nutzer sehen den Quellcode der von ihnen genutzten Software nicht an, da er für Laien nicht zu verstehen ist (wie eine Fremdsprache). Änderungen an diesem Quellcode ändern die Funktionsweise des Programmes.[2]

2.1. Adware

Als Adware wird die Vollversion einer kommerziellen Software bezeichnet, die Werbung auf der Benutzeroberfläche anzeigt oder zusätzlich Programme installiert, die Werbung anzeigen. In der Regel ist Adware kostenlos und funktionell nicht eingeschränkt. Die Software wird durch diese Werbeanzeigen (Pop-Ups, Banner etc.) finanziert.

Häufig bietet sich die Möglichkeit für die Software zu zahlen um keine Werbung mehr zu erhalten, dadurch wird sie zu einer normalen kommerziellen Software.

Der schlechte Ruf von Adware ergibt sich daraus, dass zusammen mit dem Programm häufig unerwünschte Spyware installiert wird. Diese Spyware soll dazu dienen die personalisierte Werbung aufgrund des Surfverhaltens der Nutzer anzubieten. Die gesammelten Daten können aber an den Entwickler und/oder Dritte gesendet werden.

Adware ist per se keine Schadsoftware, wird aber durch die häufig vorkommende Mitinstallation von Spyware als unerwünscht und als schädlich betrachtet.[3]

[2] (Anon., kein Datum)
[3] (Unbekannt, 2017)

2.2. Freeware

Freeware ist eine kostenlose vertriebene Software, die von den Funktionen her nicht eingeschränkt ist. Freeware ist urheberrechtlich geschützt, und an Hand der Lizenzen wird festgelegt, dass weder eine Einsicht in den Quellcode der Software, noch die Modifikation ebendieses möglich sein soll. Somit bleiben die Rechte für die Modifikation beim Entwickler.

Die Programme dürfen kopiert und weitergegeben werden.

2.3. Shareware

Shareware ist ein Vermarktungsmodell kommerzieller Software. Es gibt zwei verschiedene Arten von Shareware:

Das Programm kann mit den gleichen Funktionen wie die gekaufte Version kostenlos heruntergeladen und installiert werden und über einen bestimmten Zeitraum (meist 30 Tage) genutzt werden. Danach muss man sich kostenpflichtig für das Programm registrieren, wenn man es weiter nutzen möchte.

Die Software lässt sich als eine funktionell eingeschränkte Version von der kostenpflichtigen Version downloaden, installieren und nutzen, so lange man möchte. Ab dem Moment, in dem die volle Version genutzt werden möchte muss man für das Programm zahlen.[4]

2.4. Open Source Software

Open Source bezeichnet Software, deren Quellcode von jedem eingesehen, modifiziert und genutzt werden kann. Des Weiteren darf Open Source Software nach Belieben kopiert und weitergegeben werden, das heißt es gibt keine Nutzungseinschränkungen bezüglich Anzahl der Installationen und Benutzer.

Meist gibt es keine oder sehr geringe Lizenzgebühren, die von der jeweiligen Lizenz abhängig sind.[5]

---

[4] (Unbekannt, 2017)
[5] (Anon., 2017)

2.5. Kommerzielle Software

Der Begriff proprietäre Software beschreibt eine Software, die kein Kopieren und Weiterverbreiten des Programms erlaubt. Des Weiteren ist die Einsicht in den Quellcode und somit auch dessen Modifikation nicht möglich.

Da viele Unternehmen die Software unter Gewinnabsicht vertreiben, den Quellcode als Betriebsgeheimnis sehen, ist die meiste proprietäre Software gleichzeitig kommerziell. Kostenlose proprietäre Software wird als Freeware bezeichnet.

Im weiteren Verlauf dieser Arbeit werden die Begriffe kommerziell und proprietär Synonym verwendet, da es sich bei beiden Arten um Closed Source Software handelt.

3. Open Source Software

Der Begriff Open Source Software wurde erstmals von der Open Source Initiative (OSI) benutzt. Laut dieser wird Open Source Software durch folgende Kriterien definiert:

Freie Weitergabe: Die Weitergabe oder der Verkauf der Software oder Komponenten der Software in einer Software-Distribution darf nicht durch die Lizenz verhindert werden. Für die Lizenz dürfen keine Lizenzgebühren anfallen.

Verfügbarer Quellcode: Der Quellcode muss für alle Nutzer verfügbar sein.

Beliebige Modifikation der Software: Der Quellcode darf von den Nutzern beliebig verändert und angepasst werden.

Abgeleitete Arbeiten: Modifikationen müssen unter derselben Lizenz wie die ursprüngliche Software vertrieben werden.

Keine Diskriminierung von Personen oder Gruppen: Die Software muss allen Menschen zugänglich sein. Es dürfen keine Personen von der Nutzung ausgeschlossen werden (z.B. Bürger eines bestimmten Staates).

Keine Nutzungseinschränkung: Die Software darf nicht für einen bestimmten Nutzen ausgeschlossen werden (z.B. militärische Verwendung).

Technologie-neutrale Lizenz: Die Lizenz darf keine reine Distribution via CD oder WEB verlangen.[6]

3.1. Entstehung

Die Entwicklung von Open Source Software im eigentlichen Sinne begann mit dem Beginn der Verbreitung des Computers Anfang der 60er Jahre. Zu diesem Zeitpunkt gab es noch keine proprietäre Software, da an Software die zu diesem Zeitpunkt entstand, aufgrund der geringen Anzahl an Experten gemeinsam entwickelt wurde. Es wurde also alle Software wie OS Software behandelt. Bis 1969 war der freie Austausch von Quelltexten und Software vollkommen normal.

Die Entstehung von proprietärer Software wurde durch die Entkopplung von Soft-und Hardware ermöglich, danach war Software unabhängig von Hardware erhältlich. Ende der

[6] (Anon., 2007)

70er Jahre erhöhte sich der Wettbewerb in der Softwarebranche, da sich Computer exponentiell verbreiteten. Aufgrund dieses Wettbewerbes fingen die ersten Softwarehersteller an, ihren Quellcode geheim zu halten.[7]

Dass viele Programmierer weiterhin Software untereinander austauschten, fand keinen Anklang bei den Vertreibern kommerzieller Software. Aufgrund dessen schrieb Bill Gates 1976 einen Brief mit dem Titel „Open Letter to Hobbyists“ in welchem er sich darüber beklagte, dass nur 10% der Microsoftnutzer diese Software auch käuflich erworben habe, der Rest habe sie gestohlen. Er empfand es als unfair, die Mühen und Zeit der Programmierer nicht zu entlohnen. Dies sorgte für Aufruhr in der „Tauschgemeinde“, da Bill Gates etwas zuvor Selbstverständliches nun als Straftat titulierte.[8]

Um den Nutzern die ihnen genommene Freiheit wiederzugeben gründete Richard Stallman, der Entwickler des freien Betriebssystems GNU, 1985 die Free Software Foundation. Damit wollte er den freien Zugang zum Quellcode für alle Nutzer gewähren. Das Free in Free Software Foundation soll dabei mit Freedom (deutsch: Freiheit) und nicht mit kostenlos in Verbindung gebracht werden.

„This is a matter of freedom, not price, so think of “free speech,” not “free beer.”[9]

Die Eigentliche Einführung des Begriffs Open Source entstand durch die Gründung der Open Source Initiative (OSI) 1998. Die OSI ist eine non-profit Organisation. Sie definiert den Open Source Begriff und somit, welche Software die Kriterien erfüllt. Des Weiteren führt sie eine Liste von allen Open Source Lizenzen. Ihr Ziel ist nicht die Verbreitung kostenloser Alternativen für kommerzielle Programme, sondern die Entwickelbarkeit qualitativ hochwertiger Software sicherzustellen und zu verhindern, dass dieselben Komponenten von unterschiedlichen Entwicklern neu entwickelt werden.

[7] (Anon., 2017)
[8] (Anon., 2017)
[9] (Anon., 2016)

3.2. Aktueller Stand

Für die meisten kommerziellen Programme gibt es mit mittlerweile Open Source Alternativen. Der Reifegrad dieser Alternativen hängt von dem jeweiligen Einsatzgebiet ab. In einigen Bereichen sind die Alterativen noch nicht ausgereift zum Beispiel im Bereich firmenspezieller Software (z.B. CNC Programme). Die betroffenen Programme sind aber noch aktiv, so dass eine schnelle Entwicklung und Verbesserung möglich ist.[10]

3.3. Lizenzen

Es ist ein weit verbreitetes Missverständnis, dass Open Source Software Lizenzfrei ist. Jede Software ist grundsätzlich urheberrechtlich geschützt, das bedeutet, dass nur der Urheber das Recht hat, die Software zu vervielfältigen und zu verteilen. Deshalb muss auf dieses Recht explizit verzichtet werden, da jede Weitergabe ansonsten illegal wäre. Dies erfolgt in der Regel durch speziell auf Open Source Software zugeschnittene Lizenzen.

Es gibt über einhundert verschiedene Lizenzen, die auf die unterschiedlichen Bedürfnisse angepasst sind. In etwa zwei Drittel aller Open Source Software Programme sind unter den drei größten Lizenzanbietern lizensiert.

3.3.1. Gnu General Public Licence

Die GPL wurde von Richard Stallman entwickelt. Es handelt sich um eine copyleft Lizenz, das bedeutet, dass alle Änderungen an der Software mit den gleichen Freiheitsgraden weitergegeben werden müssen, es darf also keine Nutzungseinschränkungen geben. Eine unter der GPL stehende Software und deren Modifikationen dürfen nicht Ausgangsmaterial für proprietäre Software sein.

3.3.2.Apache License

Die Apache License gehört zu der Apache Software Foundation. Die Lizenz hat kein Copyleft, das heißt, der Quellcode darf auch in proprietärer Software verwendet werden. Der Quellcode darf nach Belieben verändert und genutzt werden, allerdings muss jede Version eine Kopie der ursprünglichen Lizenz beinhalten

[10] (Anon., 2014)

3.3.3. MIT License

Wie die Apache License erlaubt auch die MIT License die freie Weitergabe und Modifikation der Software, auch für kommerzielle Zwecke. Software unter dieser Lizenz beinhaltet einen Copyright Vermerk, der nicht entfernt werden darf, wenn die Software modifiziert weitergegeben wird.

3.4. Vorteile

Qualitätsnachweis: Aufgrund des einsehbaren Quellcodes ist es für unabhängige Programmierer möglich, die Qualität der Software nachzuweisen.

Man-Power: Die Open Source Gemeinde hat viele Mitglieder, die sich an den Projekten beteiligen können. Es muss allerdings nicht ein riesiges Team sein, die Entwicklung einiger Open Source Projekte wird durch wenige Mitglieder vorangetrieben.

Verbreitung: Die öffentliche Verbreitung der Programme ermöglicht ein schnelleres Weiterentwickeln der Software. Durch die Anzahl an Menschen, die Zugriff auf den Code haben, können sowohl Fehler schneller beseitigt werden, als auch Sicherheitslücken (Backdoors) besser erkannt werden. In einigen Bereichen sind Open Source Programme sogar Marktführer (z.B. der Apache Webserver).[11]

Namenhafte Hersteller: Die Unterstützung von Herstellern wie IBM oder Sun Microsystems, welche ihre Produkte als Open Source Software zur Verfügung stellen, beschleunigen die Weiterentwicklung und die Bekanntheit von Open Source.[12,13]

Ansporn: Die Programmierer haben eine hohe Motivation sich in Open Source Projekten zu beteiligen, da sie die Software meist selber nutzen.

Anpassbarkeit: Für Experten im Bereich der Programmierung ist es möglich den Quellcode zu bearbeiten und so an Bedürfnisse anzupassen. Für Laien besteht die Möglichkeit zwischen vielen Desktopumgebungen für ihr Open Source Betriebssystem zu wählen.

---

[11] (Anon., 2017)
[12] (Anon., 2017)
[13] (Anon., 2017)

Kosten: Es fallen meist keine Lizenzgebühren für die Anschaffung der Software an und es gibt ein großes Angebot von kostenlosen Open Source Erweiterungen (Gimp, Libre Office etc.).

Einfachheit: Es besteht keine Einschränkung bezüglich der Häufigkeit von Installationen eines Programmes.

Unabhängigkeit: Die Weiterentwicklung von Open Source Programmen ist nicht von dem ursprünglichen Entwickler abhängig, sie kann auch ohne diesen weitergeführt werden.

Verbesserungen: Durch die hohe Kommunikation und Weitergabe von Wünschen innerhalb der Community lassen sich Lösungen von qualifizierten Entwicklern frei von Firmenvorgaben realisieren.[14]

Schnelligkeit: Das Open Source Betriebssystem Linus gilt als besonders schnell, so laufen um Beispiel 498 der 500 schnellsten Supercomputer über Linux.[15]

Auswahl: Bezüglich des Betriebssystems besteht bei Open Source die Auswahl zwischen vielen verschiedenen Distributionen (z.B. ubuntu, redhat).[16]

Wenig Schadsoftware: Aufgrund der geringen Verbreitung von Open Source Betriebssystemen wird Schadsoftware eher für Betriebssysteme mit einem höheren Marktanteil (Windows Betriebssysteme) entwickelt.

3.4. Nachteile

Support und Garantie: Die Nutzer von Open Source Software haben keinen Anspruch auf Garantie und Support gegenüber dem Entwickler der Software. Allerdings gibt es eine große Community in diesem Bereich, die den Support übernimmt.

Unsichere Weiterentwicklung: Einige Projekte können zum Stillstand kommen, die Weiterentwicklung hängt vom Engagement der Entwickler ab.

Integration: Im Zusammenhang mit kommerzieller Software kann es zu Schwierigkeiten kommen. So läuft das kommerzielle Programm Adobe Photoshop zum Beispiel nicht auf Open Source Betriebssystemen.

[14] (Anon., 2014)
[15] (Vaughan-Nichols, 2016)
[16] (Anon., 2017)

Mangel an Know-How: Das Fehlen von Kenntnissen bezüglich des Umgangs mit Open Source Programmen führt dazu, dass weniger Menschen von kommerziellen Programmen umsteigen.

Bekanntheit: Der Mangel an Bekanntheit von Open Source Alternativen sorgt für eine langsamere Verbreitung und ist somit ein großer Nachteil. Das Betriebssystem Linux hat im Mai 2017 einen Marktanteil von nur 1,99%.[17]

Zeitintensiv: Die benötigte Zeit um neue Programme zu installieren und sich mit ihnen auseinanderzusetzen ist für viele ein Hindernis für den Umstieg auf Open Source.[18]

Fehlende Treiber: Da der Marktanteil von Open Source Betriebssystemen so gering ist, erstellen viele Hardware Hersteller keine Treiber für diese.

In Sicherheit wiegen: Die Nutzung von Open Source sollte die Nutzer nicht in Sicherheit wiegen, denn auch hier können Fehler auftreten.[19]

[17] (Anon., 2017)
[18] (Anon., 2014)
[19] (Anon., 2017)

4. Kommerzielle Software

Kommerzielle Software wird, wie der Name sagt, mit einer Gewinnerzielungsabsicht entwickelt. Die Lizenzen, denen kommerzielle Programme unterliegen, verbieten Veränderung und Weitergabe der Software. Die Vor-und Nachteile der Software, die das Gegenstück zu Open Source Software bildet, werden im Folgenden erläutert.

4.1. Vorteile

Bequemlichkeit: Für Unternehmen ist es wesentlich einfacher ein kommerzielles Softwarepaket zu beschaffen und upzudaten. Aufgrund von Bequemlichkeit wird kommerzielle Software häufig weiter genutzt, da sie sich trotz Fehlern über Jahre bewährt hat.

Planungssicherheit: Im Gegensatz zu Open Source Programmen, kann die Weiterentwicklung von kommerzieller Software in einem bestimmten Rahmen vertraglich abgesichert werden. Allerdings kann die Weiterentwicklung nicht immer gesichert sein. Wenn ein Softwareentwickler insolvent geht, wird auch die Weiterentwicklung gestoppt, da der Quellcode nicht mehr verfügbar ist, um das Programm selbstständig weiterzuentwickeln.

Marktdurchdringung: Kommerzielle Programme werden verstärkt in der Praxis eingesetzt, daher werden Lernende häufiger an diese Programme herangeführt.

Support: Das Angebot an Support und Gewährleistung ist bei kommerzieller Software professioneller, da unter anderem auch Vertragsstrafen für das Nichteinhalten festgelegter Reaktionszeiten vereinbart werden können.[20]

Kompatibilität: Die kommerziellen Programme sind untereinander kompatibel.

Vertrautheit: In Schulen und Universitäten wird meist der Umgang mit kommerzieller Software gelehrt. Dadurch fällt einem der Umgang mit dieser vertrauten Umgebung leichter.

Treiber: Aufgrund der weiten Verbreitung der Windows Betriebssysteme führt nahezu jeder Hardwarehersteller Treiber für Windows. Dadurch ist der Umgang mit Drucker, Scanner etc. erheblich erleichtert.

[20] (Anon., 2014)

4.2. Nachteile

Kosten: Beim Erwerb kommerzieller Software fallen Lizenzgebühren an. Des Weiteren müssen teilweise auch Updates bezahlt werden.

Abhängigkeit: Die Nutzer haben keinen Einfluss auf die Software, im Gegensatz zur Open Source Software, wo Wünsche der Community besser berücksichtigt werden können, da die Entwickler selbst Teil der Community sind. Die Abhängigkeit besteht insofern, als viele kommerzielle Lösungen nur auf den dazu passenden Programmen laufen, nicht aber mit der Open Source Alternative. Dadurch wird ein Wechsel erschwert. Eine andere Möglichkeit kommerzieller Anbieter, die Abhängigkeit schafft, ist die Erhebung immens erhöhter Lizenzgebühren für den Fall, dass die Software auf einer anderen als der dazugehörigen Plattform laufen soll. Dies ist der Fall bei der kommerziellen Datenbankmanagementsystem-Software Oracle. Wer diese auf einer anderen als der Oracle Virtualisierungsplattform nutzen möchte, muss höhere Lizenzgebühren zahlen. Dadurch wird dem Nutzer die eigene Software fast aufgezwungen.[21]

Kein einsehbarer Quellcode: Der Quellcode kommerzieller Software kann nicht eingesehen werden, daher können Fehlfunktionen oder Sicherheitslücken nicht von den Nutzern selbst erkannt werden. Auch wird den Nutzern mit dem fehlenden Zugriff auf den Quellcode die Anpassung der Software, wenn die Kenntnisse darüber vorhanden sind, verwehrt.

Support: Der Support im Bereich kommerzieller Software ist häufig mit Kosten verbunden, wenn nicht direkt, sind die Kosten schon in den Lizenzgebühren enthalten.

Kompatibilität: Die von kommerzieller Software erstellten Dateiformate lassen sich häufig nur mit identischer Software einsehen und verändern, auch besteht häufig ein Problem der Abwärtskompatibilität.[22]

Customer lock-in: Der Customer lock-in ist eine Strategie Kunden an das Produkt zu binden, indem der Wechsel erschwert wird. Dafür gibt es bei kommerzieller Software mehrere Möglichkeiten. Aufgrund von Bequemlichkeit und Gewöhnung binden sich die Kunden an das Produkt, da es Aufwand erfordern würde zu wechseln. Des Weiteren können die Lizenzkosten erhöht werden, unmittelbare Abhängigkeit zu komplementären Produkten bestehen und die Kundenbindung vertraglich abgesichert werden.[23]

---

[21] (Nickel, 2017)
[22] (Anon., 2014)
[23] (Anon., 2017)

Release Termine: Bei kommerzieller Software wird der Nutzer gezwungen, die Schnelligkeit der Veröffentlichung von Updates zu akzeptieren, da hierfür manchmal spezielle Termine angesetzt werden (z.B. Messetage)[24]

[24] (Anon., 2014)

5. Sicherheit

Dieser Abschnitt befasst sich mit der Sicherheit, da es sich hier um zwei zu vergleichende Softwarearten handelt, genauer gesagt um IT-Sicherheit. IT-Sicherheit verfolgt das Ziel, Angriffsmöglichkeiten zu verhindern und sich vor unberechtigten Zugriffen zu schützen. Die drei wichtigsten Ziele zum Schutz der Nutzer lassen sich wie folgt definieren:

Vertraulichkeit: Die vertraulichen Informationen der Nutzer müssen vor dem Zugang durch andere geschützt werden.

Integrität: Es darf keine unbemerkte Modifikation stattfinden, jede Veränderung muss festgehalten werden.

Verfügbarkeit: Den Nutzern müssen Dienstleistungen, Funktionen und Informationen bezüglich des IT-Systems zum gewünschten Zeitpunkt zur Verfügung stehen.

Des Weiteren gibt es neben den drei Hauptzielen noch weitere Ziele, zum Beispiel Datenschutz, Datensicherheit, Authentisierung und Autorisierung. [25]

Die absolute Sicherheit einer Software ist nie gegeben, denn eine Software ist nur so lange fehlerlos und ohne Sicherheitslücken, bis diese Fehler erkannt werden und mit dem nächsten Update beseitigt werden. Um die Sicherheit von Open Source und proprietärer Software zu vergleichen, ist es notwendig auch den Umgang mit Sicherheitslücken bezüglich Schnelligkeit und Gründlichkeit zu untersuchen.

Bezüglich der Sicherheit gibt es im Open Source wie im kommerziellen Bereich konkrete Gefahren, die abgewendet werden müssen. Einige dieser Gefahren sind:

Staatliche Überwachung: Es besteht die Gefahr von sogenannten Backdoors, also implementierte Dienste, die ohne das Wissen der Nutzer Daten an staatliche Behörden (Nachrichtendienste) weitergeben.

Datenabfluss: Äquivalent zu der Datenweitergabe an staatliche Dienste, können auch Daten an größere Unternehmen weitergeleitet werden.

Kriminalität: Es besteht die Gefahr, das kriminelle Hacker Sicherheitslücken einer Software ausnutzen, um Daten zu entwenden oder zu manipulieren.[26]

[25] (Anon., 2012)
[26] (Kruse, 2015)

5.1. Sicherheit von Open Source Software

Da wie bereits erwähnt, Open Source Software bedeutet, dass der Quellcode frei zur Verfügung gestellt wird, erweist sich dies als großer Vorteil bezüglich der Sicherheit. Aufgrund dieser Eigenschaft ist es tendenziell jedem möglich den Quellcode auf Qualität und mögliche Fehler einzusehen. In der Realität ist dieser Vorgang aber nur für Menschen mit dem technischen Wissen dafür möglich. Da aber der Einsatz von Open Source Software auch in der Wirtschaft steigt, wird die Anzahl an Nutzern mit einem großen Interesse an sicherer Software immer größer. Durch dieses „Viele-Augen-Prinzip“ soll die Qualität von Open Source Software, vor allem in Bezug auf Sicherheit, gewährleistet werden.

Um die Qualität von Änderungen an Software sicherzustellen durchlaufen diese immer einen ähnlichen Prozess. Dabei wird jede Änderung in Form eines Patches eingereicht, dieser wird anschließend öffentlich diskutiert. Daraus folgt dann entweder die Annahme der Änderung, eine Verbesserung mit anschließender Annahme oder die Ablehnung. Um die Änderung durchzuführen wird sie von einem Verantwortlichen des Projektes in den Quellcode eingefügt.[27,28]

Im Gegensatz zu den meisten kommerziellen Anbietern ist der Umgang mit Sicherheitslücken und Fehlern bei Open Source Projekten wesentlich transparenter, da öffentliche Plattformen für Fehlerberichte obligatorisch sind.[29]

5.1.1. Heartbleed Bug

Der Heartbleed Bug war einer der schwerwiegendsten Fehler in der Open Source Geschichte. Er bezeichnet eine Schwachstelle in der OpenSSL Bibliothek, aufgrund derer über die verschlüsselte Transport Layer Security (TLS) Verbindung geheime Daten von Servern ausgelesen werden konnten.

Die Heartbeatfunktion von OpenSSL wurde am 14.März 2012 veröffentlicht und im April 2014 wurde der Sicherheitshinweis für die kritische Lücke veröffentlicht. Damit bestand der Heartbleed Bug für 27 Monate, betroffen war allerdings nicht die Grundfunktion von OpenSSL sondern die Heartbeat Erweiterung.

---

[27] (Gillies, 2014) (Anon., 2017)
[28] (Anon., 2017)
[29] (Kruse, 2015)

Die Funktion der Heartbeat Erweiterung war die Sicherstellung einer intakten Verbindung zwischen Klient und Server, da TLS keine Verbindung aufrechterhalten kann, wenn kein ständiger Datenaustausch besteht. Um die Verbindung sicherzustellen werden zwischen Klient und Server sogenannte Heartbeatnachrichten übertragen.

Normalerweise schickt der Klient eine Hearbeatrequest in Form einer Zeichenfolge und der dazugehörigen Länge an den Server, dieser sendet dann die Zeichenfolge mitsamt Länge (Heartbeatresponse) zurück, um zu bestätigen, dass die Verbindung intakt ist. Währenddessen befindet sich die Zeichenfolge auf dem Arbeitsspeicher des Servers.

Beim Erstellen der Heartbeatresponse wurde allerdings nicht geprüft, ob die Länge der Zeichenkette mit der angegebenen Zeichenlänge übereinstimmt. Deshalb war es möglich, eine kürzere Zeichenfolge als die angegebene zu senden, daraufhin wurden dann vom Server die fehlenden Zeichen aus dem Inhalt des Arbeitsspeichers entnommen und als Antwort zurückgesendet. Auf diese Weise konnten sensible Daten, wie etwa private Schlüssel und Passwörter aus dem Server ausgelesen werden. Die maximale Größe dieser Nachrichten und somit der durch einen Angriff zu erhaltenen Informationen betrug 64 Kilobyte.[30,31]

Die Folgen waren laut Aussage des Sicherheitsexperten Bruce Schneier katastrophal.[32] Laut Netcraft waren in etwa 17% aller Websites betroffen. Die Folgen waren immens, mehrere Dienstanbieter im Internet mussten ihre Nutzer auffordern die Passwörter zu ändern und neue Zertifikate zur sicheren Verschlüsselung austeilen. [33]

Der Heartbleed Bug ist ein Beispiel dafür, dass die 100 prozentige Sicherheit auch bei Open Source Software nicht gegeben ist, sondern dass auch hier schwerwiegende Sicherheitslücken auftreten und erst nach einiger Zeit entdeckt beziehungsweise geschlossen werden können.

[30] (Anon., 2014)
[31] (Anon., 2017)
[32] (Schneier, 2014)
[33] (Mutton, 2014)

5.2. Sicherheit von kommerzieller Software

Das wohl wichtigste Sicherheitskonzept von proprietärer Software ist das Geheimhalten des Quellcodes. Dieses Konzept wird auch „Security through Obscurity“ (Sicherheit durch Unklarheit) genannt.[34] Es besagt, dass mit der Geheimhaltung des Quellcodes auch mögliche Schwachstellen vor Angreifern versteckt werden können (gleichzeitig können aber auch andere Dinge geheim gehalten werden, zum Beispiel Backdoors für Geheimdienste). Diese Methode wird wahrscheinlich das Entdecken von Schwachstellen erschweren, macht es aber gleichzeitig auch nicht unmöglich. Eine der bekanntesten Methode um einen Teil des verborgenen Quellcodes zu analysieren ist das sogenannte „Reverse Engineering“.[35]

Für proprietäre Software gibt es keine öffentliche Liste für Sicherheitsmängel oder Fehler. Die Entdeckung und Behebung von Fehlern werden den Nutzern erst im Moment der Aktualisierung der Software mitgeteilt. Bei vielen proprietären Softwareanbietern gibt es festgelegte Tage zur Veröffentlichung der Fehlerbehebung, diese werden „Patchdays“ genannt. Dies führt dazu, dass selbst bekannte Sicherheitslücken für einige Wochen nicht behoben werden. Unter anderem wird dieses Verfahren von den kommerziellen Anbietern Microsoft (Patches werden jeden zweiten Dienstag im Monat freigegeben), Oracle (Patches werden vierteljährlich freigegeben), SAP und Adobe genutzt.[36,37]

Bezüglich der Sicherheit des Betriebssystems wird häufig der Vergleich Linux vs. Windows angeführt. Dabei ist auffällig, dass es für Linux kaum Schadsoftware gibt, was sich auf den geringen Marktanteil zurückführen lässt, welcher Linux Nutzer zu einem unattraktiveren Ziel macht.

Abschließend lässt sich sagen, dass es für kommerzieller Software keine Sicherheit bezüglich enthaltener Backdoors gibt, da der Quellcode nicht von unabhängigen Programmierern eingesehen werden kann, was eine riesige Lücke für die Datensicherheit der Nutzer kommerzieller Programme bedeutet.[38]

---

[34] (Anon., 2017)
[35] (Stefan Eicker, 2012)
[36] (Kruse, 2015)
[37] (Oedinger, 2016)
[38] (Kruse, 2015)

6. Verschlüsselungssoftware

Das Kerkhoffs' Prinzip besagt, dass die Sicherheit eines Verschlüsselungsverfahren nicht auf der Geheimhaltung des Verschlüsselungsalgorithmus, sondern auf der Geheimhaltung des Schlüssels basiert. Der Verschlüsselungsalgorithmus ist die Rechenvorschrift, mit der ein aus einem Ursprungsdokument ein verschlüsseltes Dokument entsteht und mit deren Hilfe später wieder der Klartext gewonnen werden kann.

Es ist also nicht wichtig, wie stark der Algorithmus geheim gehalten wird. Die bekanntesten Verschlüsselungsalgorithmen sind vollständig veröffentlich. Da es nicht die Geheimhaltung ist, von der die Sicherheit abhängt, ergibt sich dadurch auch kein Nachteil für Open Source Verschlüsselungsprogramme.

Im Gegensatz dazu lässt sich bei kommerzieller Software nicht sagen, ob das Verschlüsselungsprogramm nicht Backdoors für Nachrichtendienste eingebaut hat. So steht das Windows Verschlüsselungsprogramm BitLocker unter Verdacht, Daten von Nutzern an die NSA weiterzuleiten. Dies ist der fatalste Nachteil kommerzieller Software, denn der eigentliche Sinn von Verschlüsselungsprogrammen besteht darin, die Daten der Nutzer geheim zu halten und nicht sie an staatliche Behörden weiterzuleiten.

Ein wesentlicher Vorteil von Open Source lässt sich am Beispiel von der Verschlüsselungssoftware TrueCrypt beschreiben. Die Software wird seit 2014 nicht mehr weiterentwickelt, da der Quellcode aber öffentlich zugänglich ist, konnten andere Entwickler das Projekt weiterführen. Auf diese Weise entstand die Software VeraCrypt. Da die Systeme aufeinander beruhen ist es auch möglich, mit TrueCrypt verschlüsselte Container mit VeraCrypt zu entschlüsseln. Eine solche Kompatibilität ist bei proprietären Programmen nicht möglich.[39]

6.1. E-Mail Verschlüsselung (PGP & GnuPG)

Im Zeitalter der digitalen Kommunikation gehört das Versenden von E-Mails zu den täglichen Aktionen. Insbesondere durch den PRISM Skandal entstand ein höheres Bedürfnis an Sicherheit für die eigene Kommunikation.

Um E-Mails zu verschlüsseln gibt es zwei grundsätzliche Methoden. S/MIME ist ein Verfahren, bei dem die Nutzer Schlüssel von offiziellen Zertifizierungsstellen erhalten. Eine 100%ige

[39] (Oedinger, 2016)

Sicherheit der Schlüssel ist also nicht gegeben, da das Kompromittieren der Zertifizierungsstellen durch Geheimdienste nicht ausgeschlossen werden kann.
Im Gegensatz dazu steht Pretty Good Privacy (PGP), bei diesem Programm erstellt jeder Nutzer seinen Schlüssel selber.

E-Mails werden von PGP asymmetrisch verschlüsselt, das bedeutet, dass die E-Mail mit dem öffentlichen Schlüssel des Empfängers verschlüsselt wird. Der Empfänger kann dann die E-Mail mit seinem privaten Schlüssel entschlüsseln. Das Prinzip funktioniert aufgrund des mathematischen Zusammenhangs des öffentlichen und privaten Schlüssels.
Von PGP wird nur der E-Mail Inhalt verschlüsselt, die Metadaten wie Sender und Empfänger bleiben offen.[40]

Bisher ist es noch nicht gelungen, den Algorithmus von PGP zu knacken. Um das System anzugreifen bedarf es also einer Methode, die den Schlüssel herausfinden kann. Dies wäre bei PGP rein theoretisch über Primfaktorzerlegung möglich. Theoretisch, da es mit der momentanen Leistung von Computern und der aktuellen Schlüssellänge von 2048 Bit (617-stellige Dezimalzahl) Millionen Jahre dauern würde.

Die Open Source Alternative zu PGP, die höher angesehen ist aufgrund der Sicherheit keiner versteckten Backdoors, heißt Gnu Privacy Guard (GnuPG). GnuPG verdrängte PGP als Verschlüsselungssystem, da PGP das patentierte RSA Verschlüsselungsverfahren enthielt, welches bis zum Jahr 2000 nicht aus den USA exportiert werden durfte. Mittlerweile verwendet GnuPG auch das RSA Verschlüsselungsverfahren[41,42]

[40] (Anon., 2017)
[41] (Düvel, 2015)
[42] (Anon., 2017)

8. Fazit

Wenn man die verschiedenen Softwarearten und ihre Vor-und Nachteile betrachtet, lässt sich, vor allem in Bezug auf die Sicherheit, die Open Source Variante als die bessere bezeichnen.

Ist es einem aber zum Beispiel wichtig, bestimmte Computerspiele spielen zu können, kommt man um das kommerzielle Betriebssystem Windows nicht herum.

Wie diese beiden aufgeführten Punkte gibt es sehr viele, die jeweils für die eine oder die andere Seite sprechen. Um für sich entscheiden zu können welche Variante man nutzen möchte, gilt es erst zu bedenken, welche Gewichtung die verschiedenen Kriterien für jemanden haben und welche Kriterien vielleicht ganz herausfallen, wenn man zum Beispiel gar keine Computerspiele spielt.

Des Weiteren ist natürlich nicht gesagt, dass es nur die eine oder andere Wahl gibt. Die meisten Nutzer verwenden beides und haben zum Beispiel den Open Source Browser Mozilla Firefox auf ihrem Windows Rechner installiert.

Natürlich hört sich der Vorteil, man könne die Programme selber modifizieren, sehr gut an, allerdings wird dieser Vorteil von den wenigsten genutzt. Einfach aus dem Grund, dass es den meisten an Kenntnissen mangelt.

Für mich persönlich stellt sich Open Source Software als Gewinner heraus. Das ist aber vor allem durch diese Ausarbeitung herausgekommen, da ich mich vor dem Schreiben dieser Arbeit nicht sehr viel mit dem Thema beschäftigt habe. Mein Nichtwissen ist vorrangig durch Bequemlichkeit begründet, wie es denke ich, bei vielen der Fall ist. Das beginnt damit, dass die meisten Laptops und Desktop PCs Windows als vorinstalliertes Betriebssystem haben und man sich dadurch an ein Betriebssystem gewöhnt hat und mit diesem umzugehen weiß. In der Schule und der Universität wird der Umgang mit kommerziellen Office-und Bildbearbeitungsprogrammen gelehrt. Das erleichtert meistens auch den Einstieg ins Berufsleben, da in der Praxis häufig die gleichen kommerziellen Programme genutzt werden. Und wenn man den Umgang mit diesen Programmen schon beherrscht, warum sollte man sie dann nicht auch im Privatleben nutzen? Aus diesen Gründen habe ich mich nie sehr viel mit Open Source Software beschäftigt. Ich habe dank dieser Arbeit angefangen mich mit der Thematik zu beschäftigen und die und habe mir fest vorgenommen mich auch weiterhin darüber zu informieren, die Bequemlichkeit abzulegen und den Umgang mit Open Source Programmen zu erlernen.

9. Verweise

Anon., 2007. *The Open Source Initiative.* [Online]
Available at: https://opensource.org/osd
[Accessed 19 06 2017].

Anon., 2012. *Bundesamt für Sicherheit in der Informationstechnik.* [Online]
Available at:
https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/Leitfaden/GS-Leitfaden_pdf.pdf?__blob=publicationFile
[Accessed 17 06 2017].

Anon., 2014. *net.in.tum.* [Online]
Available at: https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2014-08-1/NET-2014-08-1_20.pdf
[Accessed 05 06 2017].

Anon., 2014. *opensourceecology.* [Online]
Available at:
https://wiki.opensourceecology.de/Vorteile/Nachteile_von_Open_Source_Software
[Accessed 2017 06 05].

Anon., 2016. *GNU.* [Online]
Available at: https://www.gnu.org/philosophy/open-source-misses-the-point.en.html
[Accessed 06 05 2017].

Anon., 2017. *Netcraft.* [Online]
Available at: https://news.netcraft.com/archives/2017/04/21/april-2017-web-server-survey.html
[Accessed 01 06 2017].

Anon., 2017. *netmarketshare.* [Online]
Available at: https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0
[Accessed 19 06 2017].

Anon., 2017. *Wikipedia.* [Online]
Available at: https://en.wikipedia.org/wiki/Open-source_model
[Accessed 19 06 2017].

Anon., 2017. *Wikipedia.* [Online]
Available at: https://de.wikipedia.org/wiki/Propriet%C3%A4re_Software
[Accessed 04 06 2017].

Anon., 2017. *Wikipedia.* [Online]
Available at: https://en.wikipedia.org/wiki/Open_Letter_to_Hobbyists
[Accessed 05 06 2017].

Anon., 2017. *Wikipedia.* [Online]
Available at: https://de.wikipedia.org/wiki/Sun_Microsystems
[Accessed 01 06 2017].

Anon., 2017. *Wikipedia.* [Online]
Available at: https://de.wikipedia.org/wiki/IBM
[Accessed 01 06 2017].

Anon., 2017. *Wikipedia.* [Online]
Available at: https://en.wikipedia.org/wiki/Open-source_software_security]
[Accessed 05 06 2017].

Anon., 2017. *Wikipedia.* [Online]
Available at: https://en.wikipedia.org/wiki/Vendor_lock-in
[Accessed 19 06 2017].

Anon., 2017. *Wikipedia.* [Online]
Available at: https://en.wikipedia.org/wiki/Software_peer_review
[Accessed 10 06 2017].

Anon., 2017. *Wikipedia.* [Online]
Available at: https://en.wikipedia.org/wiki/Software_peer_review
[Accessed 10 06 2017].

Anon., 2017. *Wikipedia.* [Online]
Available at: https://en.wikipedia.org/wiki/Heartbleed
[Accessed 15 06 2017].

Anon., 2017. *Wikipedia.* [Online]
Available at: https://de.wikipedia.org/wiki/Security_through_obscurity
[Accessed 10 06 2017].

Anon., 2017. *Wikipedia.* [Online]
Available at: https://en.wikipedia.org/wiki/GNU_Privacy_Guard
[Accessed 05 06 2017].

Anon., 2017. *Wikipedia.* [Online]
Available at: https://en.wikipedia.org/wiki/Pretty_Good_Privacy
[Accessed 05 06 2017].

Anon., n.d. *Opensource.* [Online]
Available at: https://opensource.com/resources/what-open-source19
[Accessed 19 06 2017].

Anon., n.d. *wikipedia.* [Online]
Available at: https://en.wikipedia.org/wiki/Open-source_software_security]
[Accessed 19 06 2017].

Düvel, S., 2015. *t3n.* [Online]
Available at: http://t3n.de/magazin/pgp-gpg-einsatz-e-mail-verschluesselung-leicht-gemacht-234668/
[Accessed 05 06 2017].

Gillies, S., 2014. *opensource.* [Online]
Available at: https://opensource.com/business/14/12/importance-peer-review-code
[Accessed 19 06 2017].

Kruse, G., 2015. *pro-linux.* [Online]
Available at: http://www.pro-linux.de/artikel/2/1796/propriet%C3%A4r-vs-open-source-die-ewige-debatte-um-die-sicherheit.html
[Accessed 12 06 2017].

Mutton, P., 2014. *Netcraft.* [Online]
Available at: https://news.netcraft.com/archives/2014/04/17/netcraft-releases-heartbleed-indicator-for-chrome-firefox-and-opera.html
[Accessed 05 06 2017].

Nickel, O., 2017. *golem.* [Online]
Available at: https://www.golem.de/news/oracle-lizenzvereinbarung-erhoeht-preise-fuer-datenbanksysteme-1701-125909.html
[Accessed 19 06 2017].

Oedinger, M., 2016. *Security-Insider.* [Online]
Available at: http://www.security-insider.de/open-source-geht-nicht-zu-lasten-der-sicherheit-a-521988/
[Accessed 10 06 2017].

Oedinger, M., 2016. *Security-Insider.* [Online]
Available at: http://www.security-insider.de/open-source-geht-nicht-zu-lasten-der-sicherheit-a-521988/
[Accessed 10 06 2017].

Schneier, B., 2014. *Schneier.* [Online]
Available at: https://www.schneier.com/blog/archives/2014/04/heartbleed.html
[Accessed 19 06 2017].

Stefan Eicker, R. J., 2012. *Enzyklopädie der Wirtschaftsinformatik.* [Online]
Available at: http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Integration-und-Migration-von-IT-Systemen/Software-Reengineering/Reverse-Engineering
[Accessed 10 06 2017].

Unbekannt, 2014. *wikipedia.* [Online]
Available at:
https://wiki.opensourceecology.de/Vorteile/Nachteile_von_Open_Source_Software
[Accessed 19 06 2017].

Unbekannt, 2017. *Searchsecurity.* [Online]
Available at: http://searchsecurity.techtarget.com/definition/adware
[Accessed 19 06 2017].

Unbekannt, 2017. *Wikipedia.* [Online]
Available at: https://de.wikipedia.org/wiki/Shareware
[Accessed 19 06 2017].

Vaughan-Nichols, S. J., 2016. *ZDNet.* [Online]
Available at: http://www.zdnet.com/article/almost-all-the-worlds-fastest-supercomputers-run-linux/
[Accessed 01 06 2017].