Themen dieser Ausgabe sind u. a.

ISSN 1867-7991

Editorial

Verzögerung im Betriebsablauf

Wie einige freiesMagazin-Leser sicherlich bemerkt haben, hat es das Magazin letzte Woche nicht rechtzeitig an den „Internet-Kiosk“ geschafft [1]. Die Gründe dafür sind vielfältig.

Der wichtigste ist vermutlich, dass freiesMagazin nur von einigen, wenigen freiwilligen Helfern getragen wird [2]. Die Redaktion ist mit drei Mitgliedern eigentlich ganz gut vertreten, dennoch kann es passieren, dass genau diese drei relativ gleichzeitig private oder berufliche Verpflichtungen haben, sodass niemand sich um die Veröffentlichung des Magazin oder die Betreuung der Leser und Autoren kümmern kann.

Das Korrektorat ist mit sechs Mitglieder ganz gut besetzt, beim Layout des Magazins sind drei Teammitglieder erfahrungsgemäß aber zu wenig. Hier entsteht vor allem im Sommer leicht das gleiche Loch, sodass nicht alle Artikel rechtzeitig zur Veröffentlichung gesetzt werden können. In diesem Bereich suchen wir deshalb immer wieder Unterstützung und freuen uns, wenn ein Leser aus dem passiven in den aktiven Status wechseln möchte, um freiesMagazin mitzugestalten. Bewerbungen und Rückfragen können gerne an redaktion@freiesMagazin.de geschickt werden.

Es ist natürlich auch für uns ärgerlich, wenn wir eine Ausgabe verschieben müssen. Dennoch haben wir auch als Hobbyprojekt einen selbst auferlegten Qualitätsstandard, den wir halten wollen, bevor eine freiesMagazin-Ausgabe veröffentlicht wird. Das ist auch der Grund, wieso wir die nicht ganz fertige Oktoberausgabe lieber um eine Woche verschoben haben.

Programmierwettbewerb

Seit 2009 gibt es (fast) jeden Oktober bei freiesMagazin einen Programmierwettbewerb [3]. Dieser macht den Teilnehmer, Lesern und auch der freiesMagazin-Redaktion immer wieder Spaß. Aber: Er kostet auch viel Zeit in der Vorbereitung und Teilnahme. Diese Zeit hat aus den gleichen Gründen wie oben in den letzten Wochen und Monaten gefehlt, sodass der Wettbewerb mindestens auf Dezember verschoben werden muss.

Und nun wünschen wir Ihnen viel Spaß mit der neuen Ausgabe.

Ihre freiesMagazin-Redaktion

LINKS

[1] http://www.freiesmagazin.de/20151001-oktoberausgabe-von-freiesmagazin-verschiebt-sich
[2] http://www.freiesmagazin.de/impressum
[3] http://www.freiesmagazin.de/mitmachen#wettbewerb

Teilen *Kommentieren*

Inhalt

Proprietär vs. Open Source – Die ewige Debatte um die Sicherheit

von Gerrit Kruse

Die Sicherheit von Open-Source-Software ist ein häufig diskutierter Aspekt, wenn es um das Verhältnis von proprietärer und quelloffener Software geht. Besonders von Seiten der Open-Source-Gemeinde wird gerne der Sicherheitsaspekt von Open-Source-Software hervorgehoben. Wie immer bei einer ideologisch unterfütterten Debatte wird es mit Begriffen, Szenarien und Argumenten nicht immer so genau genommen. Der Artikel soll etwas mehr Aufklärung bieten.

In der Sicherheitsfrage werden viele Aspekte vermischt. Es geht einerseits um die grundsätzliche Sicherheit vor schädlichen Programmen, ebenso wie dem Schutz vor Überwachung und Datenabschöpfung andrerseits.

Begriffsdefinition: Open Source bzw. quelloffen ist jede Software, deren Quellcode einsehbar ist. Es muss sich dabei nicht zwingend um Software unter einer Freien Lizenz handeln, dies ist allerdings oft deckungsgleich. Proprietär bzw. geschlossen ist in diesem Kontext jede Software, die lediglich als binäre Software verteilt wird.

Welche Gefahren sollen abgewendet werden?

Sicherheit im wortwörtlichen Sinne, d. h. ein absolutes Gesichertsein vor Beeinträchtigungen jedweder Art ist ein Zustand, dessen man sich bei Software nie gewiss sein kann. Fehlerfreiheit und die Abwesenheit von Sicherheitslücken ist ein Momentzustand – bis die nächste Sicherheitslücke bekannt und anschließend mit einem Update beseitigt wird. Die Sicherheitsdebatte ist also auch eine Frage, wie mit Sicherheitslücken umgegangen wird.

Grundsätzlich muss man bei einer Sicherheitsdebatte konkretisieren, welche Gefahren man abwenden möchte. Grob vereinfacht können das einige der folgenden sein (ohne Garantie auf Vollständigkeit):

- Gefahr durch staatliche Überwachung
- Gefahr durch Datenabfluss zu großen Unternehmen (und ggf. daraus folgender staatlicher Überwachung)
- Gefahr durch implementierte Dienste, derer man sich als Anwender nicht bewusst ist (korrespondiert mit Punkt 2)
- Gefahr durch Kriminalität

Dabei können die Gefahren durch verschiedene Anwender oder unterschiedliche Anwendungsszenarien jeweils unterschiedlich gewichtet werden. Jeder dieser Punkte impliziert andere Aspekte. So ist es recht unwahrscheinlich, dass Software gezielt Lücken für Kriminelle enthält, wohingegen Backdoors – also Hintertüren – für Geheimdienste in dem Bereich ein omnipräsentes Thema sind – egal ob es sie schon gibt oder nicht.

Gefahr durch Datenabfluss muss noch nicht einmal durch eine Sicherheitslücke entstehen, sondern kann durch die Funktion einer Software bedingt werden. Gerade Cloud-Dienste verleiten den Benutzer offensichtlich dazu, seine Daten vom heimischen Betriebssystem in das Netz einzuspeisen. Solche Herstellerdienste sind in modernen Betriebssystemen oft fest implementiert.

Ist ein System weniger sicher?

Der Sicherheitsvergleich zwischen quelloffener und proprietärer Software wird allzu oft auf den Vergleich Linux vs. Windows reduziert. Allerdings hat die Thematik eigentlich nichts mit beiden Betriebssystemen zu tun, sondern kann jedwede Software betreffen. In der Praxis und in diesem Artikel ist diese Reduktion jedoch sinnvoll, da eine unsichere Basis jeglicher Sicherheit den Boden entzieht. Die Gleichsetzung von Linux und Sicherheit liegt unter anderem daran, dass Linux bislang durch relativ wenig Schadsoftware aufgefallen ist, während Windows in regelmäßigen Abständen Probleme mit Schadprogrammen hat. Zudem spielen freie Alternativen wie die BSD-Familie lediglich eine Nischenrolle und die Kritikpunkte an Microsoft Windows lassen sich auch auf das proprietäre MacOS X von Apple übertragen.

Immer wieder wird angemerkt, dass dies auch am unterschiedlichen Verbreitungsgrad liegt. Es ist schließlich immer noch so, dass die unterschiedli-

chen Windows-Versionen zusammen den Markt für herkömmliche PC-Betriebssysteme dominieren, während Linux im niedrigen einstelligen Prozentbereich stagniert. Offensichtlich dürfte es für Kriminelle interessanter sein sich auf Windows zu fokussieren. Das hat erst einmal nichts mit der Sicherheitsarchitektur zu tun. Dieses Argument wird dadurch gestärkt, dass MacOS X unter Fachleuten als relativ unsicher gilt [1], aber bisher noch keine größeren Probleme mit Schadprogrammen hat (es gab allerdings bereits ein paar in freier Wildbahn). Auch hier dürfte die geringe Verbreitung eine Rolle spielen.

Ein weiteres Problem ist, dass die Debatte oft nicht mit zeitgemäßen Argumenten geführt wird. Insbesondere viele Argumente gegen Windows speisen sich aus der Vergangenheit. Windows war bis zur Version XP ein objektiv unsicheres und für die Internetnutzung unzureichend abgesichertes System. Dies lag nicht zuletzt daran, dass der Nutzer von Haus aus mit Administratorrechten im System angemeldet war. Seitdem hat Microsoft viel an der Sicherheitsarchitektur von Windows verändert und liefert seit Windows 8 auch einen eigenen Virenscanner aus – Aspekte, die gerne unterschlagen werden. Bei Linux hat sich hingegen in den letzten Jahren wenig getan. Allerdings nicht etwa, weil man nachlässig wäre, sondern weil viele Aspekte wie ein Benutzerrechtemanagement bereits vorhanden sind. Andere Bereiche wie der unter Windows obligatorische Virenscanner sind bisher – eben mangels bekannter Schadsoftware – nicht notwendig.

Umgang mit Sicherheitslücken

Dennoch gibt es objektive Unterschiede zwischen den Systemen im Umgang mit Sicherheitslücken. Die meisten Firmen hinter proprietärer Software pflegen keine öffentliche Liste an Sicherheitsmängeln oder Fehlern. Es ist also vollkommen unklar, wie viele Sicherheitsprobleme den Firmen bekannt sind und welche Korrekturen eigentlich schon vorliegen und an den Anwender verteilt werden könnten. Der Benutzer bekommt lediglich bei den Aktualisierungen mit, dass Fehler existierten und beseitigt wurden.

Hinzu kommt die Tatsache, dass viele proprietäre Softwareprojekte festgelegte Zyklen für die Veröffentlichung von Fehlerbehebungen haben, sogenannte „patch days“. Dies trifft z. B. für Adobe, Microsoft und Oracle zu. Das heißt im Umkehrschluss aber auch, dass selbst bekannt gewordene Sicherheitslücken im schlimmsten Fall mehrere Wochen offen bleiben.

Die meisten Open-Source-Projekte pflegen dagegen einen relativ transparenten Umgang mit Sicherheitslücken und Fehlern. Öffentliche Plattformen für Fehlerberichte sind obligatorisch und manche Projekte wie Debian pflegen auch eine Liste der bekannten Sicherheitsmängeln [2] in der Linux-Distribution.

Security through obscurity?

Der Satz kommt eigentlich aus dem Netzwerkbereich, wird aber inzwischen auch oft auf proprietäre, geschlossene Software angewandt. So wird argumentiert, dass die Nichtfreigabe des Codes Kriminelle daran hindert, Sicherheitsmängel zu finden. Allerdings ist Windows das beste Beispiel dafür, dass man eben nicht den Quellcode haben muss, um Sicherheitslücken ausfindig zu machen.

Open-Source-Verfechter betonen hingegen, dass der frei zugängliche Quellcode es vielen unterschiedlichen Personen ermöglicht, diesen zu prüfen sowie Sicherheitslücken zu finden und zu melden. Die Software sei dadurch sicherer als geschlossene Projekte, wo Sicherheitsmängel unbemerkt über Jahre bestehen bleiben können – selbst wenn die dahinter stehende Firma davon weiß. Dies schlägt den Bogen zurück zum Umgang mit Sicherheitslücken.

Realitäten auf beiden Seiten

Abseits der Theorie sieht die Realität auf beiden Seiten inzwischen häufig anders aus. So werden auch bei proprietären Projekten Sicherheitslücken von anderen Angestellten großer IT-Unternehmen gefunden und – sofern das Unternehmen nicht fristgerecht reagiert – öffentlich publik gemacht [3]. Gleichzeitig gibt es Programme mit finanziellen Anreizen für Personen, die Lücken finden und an das Unternehmen melden [4].

Auf der anderen Seite gibt es viele Open-Source-Projekte mit einer sehr dünnen Personaldecke. Teilweise entwickelt ein sehr kleiner Kreis von Programmierern (manchmal auch nur einer) über Jahre hinweg ein Projekt. Es gibt hier also eine sehr beschränkte Anzahl von Personen, die

den Code wirklich kennen und verstehen. Das betrifft nicht nur irgendwelche kleinen Nischenprojekte mit eher geringer Relevanz, sondern auch große und zentrale Bestandteile der Open-Source-Welt [5].

Bekannt wurde dieser Umstand nicht zuletzt durch die Heartbleed-Lücke in OpenSSL [6]. Hier offenbarten sich gleich mehrere Probleme. Ein über Jahre gewachsener Code, den kaum noch jemand durchblickte, eine viel zu dünne Personaldecke und ein schlechtes Qualitätsmanagement. Skeptiker des Open-Source-Prinzips können seitdem – nicht ganz zu Unrecht – behaupten, dass das theoretische Mehr-Augen-Prinzip von quelloffener Software oftmals faktisch nicht existiert. Dies gilt sogar dann – das hat Heartbleed gezeigt – wenn die Software von finanzstarken Unternehmen eingesetzt wird, die theoretisch über Personal verfügen, das den Quellcode betrachten könnte.

Zurück zu den Bedrohungsszenarien

Wenn es um Sicherheit geht, muss man den Aspekt immer in Relation zu den erwarteten Bedrohungen setzen. Datenabfluss kann z. B. durchaus beabsichtigt sein. Es gibt genug Open-Source-Projekte, die Telemetriedaten erheben oder Daten in das Internet verlagern. Open Source schützt einen nicht vor einer beabsichtigten Funktion, auch wenn sich der Anwender nicht in jedem Fall der Konsequenzen bewusst ist. Android ist das beste Beispiel für ein Open-Source Betriebssystem, das nicht datenschutzfreundlicher ausgelegt ist als die proprietäre Konkurrenz [7].

Gleichzeitig kann man davon ausgehen, dass die meisten namhaften Firmen keine Schnittstellen für kriminelle Organisationen implementieren oder Daten in krimineller Absicht abgreifen. Diesen Punkt kann man für irgendwelche kleinen „Klitschen“ natürlich nicht geltend machen. Dafür ist bei proprietärer Software oftmals viel transparenter, welche Firma hinter dem Projekt steht (oder eben auch nicht). Nicht jedes freie Softwareprojekt wird schließlich so transparent entwickelt, wie es die Theorie gerne hätte. Kriminelle können aber durchaus auch Zugang zu Sicherheitslücken haben – seien sie nun offiziell bekannt oder nicht. Es gibt allerdings auch bei freier Software Sicherheitslücken, die über einen längeren Zeitraum existieren, weil es nicht sofort möglich ist sie zu schließen oder das Projekt dazu strukturell gerade nicht in der Lage ist [8]. Einen wirklichen Vorteil gibt es hier für kein Prinzip.

Seit den Enthüllungen von Edward Snowden über die weltweite Datenabschöpfung durch westliche Geheimdienste steht die staatliche Überwachung (wieder) im Mittelpunkt der Sicherheitsdebatte. Insbesondere absichtlich implementierte Hintertüren werden immer wieder thematisiert. Rein theoretisch wäre eine solche Implementierung in proprietärer Software leichter, das Unternehmen dahinter wäre schließlich juristisch verpflichtet (je nach Firmensitz versteht sich) dies umzusetzen und zudem zu schweigen [9]. Gerade der Heartbleed-Fall zeigt aber auch, dass es keineswegs unmöglich wäre, in quelloffene Software eine solche Lücke einzuschleusen. Die Partizipationshürden sind oft niedrig und das Qualitätsmanagement nicht so gut wie es sein müsste.

Es bleibt allerdings fraglich, ob Hintertüren in Betriebssystemen die Bedeutung zukommt, die ihr in der öffentlichen Debatte eingeräumt wird. Moderne PC-Hardware verfügt über eine Vielzahl von Speichern und so genannten Firmwares [10]. Sicherheitsforscher haben immer wieder hervorgehoben, dass es möglich und wahrscheinlich ist, Schadcode unsichtbar für Benutzer und Betriebssystem in den Geräten zu verankern. In diesem Fall wäre es vollkommen bedeutungslos, ob das darauf installierte Betriebssystem quelloffen oder proprietär wäre. Die Entscheidung über das Ausmaß staatlicher Überwachung trifft man also eher an der Wahlurne als bei der Wahl des Betriebssystems.

Letztlich ist es für die allermeisten Anwender eine Vertrauensfrage, auf welches Prinzip sie setzen. Sowohl quelloffene Betriebssysteme und Softwareprojekte als auch ihre proprietären Pendants haben in den letzten Jahren genug Anlass gegeben zu (ver)zweifeln. Den meisten Nutzern wird es nicht möglich sein, den Quellcode mit dem nötigen Fachwissen zu betrachten und selbst jenen die dies könnten fehlt häufig die Zeit. Das Audit der wichtigen Verschlüsselungssoftware TrueCrypt beanspruchte schließlich nicht umsonst sehr viel Arbeitszeit [11] – ein Audit, das ohne

frei zugänglichen Quellcode nicht möglich gewesen wäre. Aber auch so bleiben Zweifel, ob die Binärdateien aus diesem Quellcode gebaut wurden [12].

Ein Thema, das auch durch die Open-Source-Community zu selten thematisiert wird. Es ist auch keineswegs so, dass die Entwickler von Open-Source-Software durchweg mehr für Datenschutzthemen sensibilisiert wären, als die Entwickler proprietärer Software. Man sollte sich jedenfalls nicht in Sicherheit wiegen. Egal, ob man Software des Weltmarktführers oder den pummeligen Pinguin nutzt.

LINKS

[1] http://www.heise.de/security/meldung/Antiviren-Software-und-Apples-Schutzmechanismen-fuer-Mac-OS-X-nutzlos-2620049.html
[2] https://security-tracker.debian.org/tracker/
[3] http://www.zdnet.com/article/microsoft-fumes-google-discloses-another-windows-security-flaw/
[4] http://www.heise.de/security/meldung/Google-erhoeht-Belohnung-fuer-kooperative-Hacker-1546860.html
[5] http://www.golem.de/news/verschluesselung-gnupg-braucht-geld-und-bekommt-es-1502-112184.html
[6] https://de.wikipedia.org/wiki/Heartbleed
[7] https://de.wikipedia.org/wiki/Android_(Betriebssystem)#Kritik
[8] https://www.openoffice.org/security/cves/CVE-2015-1774.html
[9] https://de.wikipedia.org/wiki/National_Security_Letter
[10] http://www.golem.de/news/gchq-macbook-air-des-guardian-mit-merkwuerdigen-komponenten-1405-106690.html
[11] http://www.heise.de/security/meldung/Audit-abgeschlossen-TrueCrypt-7-1-weitgehend-sicher-2595838.html
[12] http://www.heise.de/security/meldung/Verschluesselungssoftware-TrueCrypt-Ein-Zweifel-weniger-2035104.html

Autoreninformation

Gerrit Kruse (Webseite) ([Mer]Curius) nutzt Linux seit 2007. Als Wissenschaftler stehen Datenschutz und produktives Arbeiten auf dem Linux-Desktop im Vordergrund seiner Interessen.

“Pain Rating” © by Randall Munroe (CC-BY-NC-2.5), http://xkcd.com/883/

Der September im Kernelrückblick

von Mathias Menzer

Basis aller Distributionen ist der Linux-Kernel, der fortwährend weiterentwickelt wird. Welche Geräte in einem halben Jahr unterstützt werden und welche Funktionen neu hinzukommen, erfährt man, wenn man den aktuellen Entwickler-Kernel im Auge behält.

Linux 4.3 - Entwicklung

Der August endete sozusagen mit einem sauberen Schnitt: der Veröffentlichung von Linux 4.2. Daher stand der September ganz im Zeichen der Entwicklung des Nachfolgers. Nach nicht ganz vierzehn Tagen zog Torvalds einen Strich unter die eingegangenen Merge Requests und schloss mit der ersten Entwicklerversion [1] das Merge Window. Somit macht nun auch ein vorsichtiger Blick in die Zukunft Sinn, was Linux 4.3 denn nun bescheren könnte:

Zuerst einmal wird 4.3 etwas schlanker werden. Die bereits diskutierte Löschung der Treiber für ext3 (siehe „Der Juli im Kernelrückblick“, freiesMagazin 08/2015 [2]) wurde nach etwas Hin und Her [3] vorgenommen [4]. Die Löschung des Treibers sollte selbst keine so gravierenden Folgen haben, da der ext4-Treiber auch mit Partitionen umgehen kann, die noch das ältere System verwenden.

Da die Verbreitung des „neuen“ Internetprotokolls IPv6 [5] stetig zunimmt, wurde dessen Verwendung nun in der Standardkonfiguration des Kernels aktiviert. Für die meisten Anwender, die den Linux-Kernel ihrer Distribution verwenden, macht dies keinen Unterschied, da diese in der Regel bereits IPv6 aktiviert hatten, doch stellt es auch ein Zeichen dafür dar, dass die Linux-Entwickler selbst ihre Umsetzung des IPv6-Protokolls für den breiten Einsatz als tauglich erachten.

Weiterhin wurde der InfiniPath-Treiber auf das Abstellgleis – sprich: den Staging- Zweig – geschoben und damit seine Löschung aus dem Kernel vorbereitet. Bei InfiniPath handelt es sich um eine Schnittstelle für die schnelle Datenübertragung sowohl zwischen als auch innerhalb von Rechnern nach dem InfiniBand-Standard [6]. Sie findet beziehungsweise fand primär im Umfeld von Hochleistungsrechnern Verbreitung.

Linux 4.3-rc2 [7] ließ es um einiges beschaulicher angehen. Weniger als 300 Änderungen kamen zusammen, doch der Patch war bemerkenswert groß. Hintergrund daran war, dass es im Umfeld der InfiniBand- Treiber einen weiteren Verlust gab. mit dem ehca-Treiber wurde eine weitere Schnittstelle stillgelegt, die bisher den EBus von IBM-Großrechnern bedienen konnte. Auch hier wurde der Treiber erst einmal nicht vollständig entfernt, sondern vorerst nur in den Staging-Bereich verschoben, doch sofern keine begründeten Einwände aufkommen, sind die Tage auch dieses Treibers gezählt.

Etwas lebhafter ging es dann wieder bei der dritten Entwicklerversion [8] zu. Sie konnte fast doppelt so viele Änderungen vorweisen wie -rc2 und hatte fast ausschließlich Fehlerkorrekturen zu bieten. Diese fielen sehr vielfältig aus; Die Palette reicht vom Beheben eines Pufferüberlaufs im amdgpu-Treiber, über einen Kernel Oops bei der Behandlung von zwischengespeicherten Speicherinhalten bis hin zum Verhindern einer Kernel-Panic bei der Behandlung von ICMP („Ping“) Paketen. Alles in allem scheint die Entwicklung auf einem guten Weg; Im Umfeld der Intel- Grafiktreiber wird noch nach der Ursache für einen Fehler gesucht, der zwar aktuell mit 4.3 irgendwie beseitigt wurde, aber unter Linux 4.2 noch sein Unwesen treibt. Dave Airlie versucht derzeit zumindest den Patch zu finden, der das Problem behebt, blieb damit jedoch bislang erfolglos [9].

Wiederum Sonntags gab Torvalds Linux 4.3-rc4 frei [10]. Sie fiel wieder deutlich kleiner aus, brachte jedoch trotzdem einige Korrekturen mit, von denen die meisten im Umfeld der Grafik-Treiber sowie der MIPS-Architektur zu finden sind. Auch wurden einige Patches für die Virtualisierungstechnik KVM [11] zurückgenommen, die die Änderungen der Speicherbehandlung von Gastsystemen und dem SCheduler betreffen. Aufgenommen wurde dagegen die neue Funktion strscpy(). Sie kann fortan von Entwickler genutzt werden, um Zeichenketten in bestimmte Puffer zu kopieren. Von ihren Vorgängern srtncpy() und strlcpy()

unterscheidet sie sich dadurch, dass sie sowohl einen Rückgabewert liefert, der vom aufrufenden Programmcode in Bezug auf den Erfolg der Operation ausgewertet werden kann und dass sie auch mit sinnlosen Übergabewerten klarkommt und gegebenenfalls die Operation abbricht.

Linux-Unterstützung für Steam-Controller

In einem Post [12] auf Google+ erwähnte Greg Kroah-Hartman [13], dass er derzeit einen Steam-Controller mit Linux testet: „Manche Tage sind härter als andere, wenn man Kernel-Treiber auf -rc Versionen testen muss“ schrieb er leicht ironisch. Weiterhin meinte er, dass alles perfekt arbeite, er aber doch noch einige Stunden testen werde, nur um sicher zu gehen.

Bereits als die ersten Vorserien-Versionen dieser Spiele-Steuerungen für einzelne Tester zu haben waren, war immer davon die Rede, dass sie unter Linux Out-of-the-Box funktionieren sollten. Mit dem bevorstehenden Verkaufsstart sowohl des Steam-Controllers als auch der auf Linux basierenden „Spiele-Konsole“ Steam Machine macht es durchaus Sinn, die Funktion prüfen und gegebenenfalls verbessern zu lassen. Kroah-Hartman ist als Initiator des Linux Driver Project [14] durchaus ein kompetenter Kandidat hierfür, doch über welchen Weg einer der Game-Controller vor dem Verkaufsstart nun konkret auf Kroah-Hartmans Schreibtisch gelandet ist, hat er bislang noch nicht erklärt.

Eine Entwicklerin geht

Nachdem sie im Juli 2013 ihrem Ärger Luft machte und den Umgangston auf der E-Mail-Liste der Linux-Entwickler kritisierte, hat die Intel-Entwicklerin Sarah Sharp ihren Rückzug aus aktiven Kernel-Entwicklung bekannt gegeben [15]. Die Zuständigkeit für den USB-Bereich hatte Sie bereits 2014 abgegeben, als Koordinatorin für den Linux-Bereich des „FOSS Outreach Program for Women“ ist sie seit Anfang diesen Jahres nicht mehr tätig und Sie wird sich auch nicht mehr für das Technical Advisory Board der Linux Foundation zur Wahl stellen. Als Grund gab sie den weiterhin „brutalen“ Umgangston unter den Entwicklern an [16], der sich trotz des in Linux 4.0 aufgenommenen „Code of Conflict“ nicht verbessert hat.

Mit Sharp verschwindet damit die in den letzten Jahren präsenteste Linux-Entwicklerin von der Bildfläche. Sie hinterlässt den USB-3.0-Treiber, den mittlerweile die meisten Linux-Nutzer verwenden dürften. Außerdem dürfte – trotz bislang mäßigem Erfolg – ihr Engagement für höflichere Umgangsformen doch vielen Entwicklern in Erinnerung bleiben. Ob und in welcher Form Sharp in Zukunft wieder zum Linux-Kernel beitragen wird, ist offen.

LINKS

[1] https://lkml.org/lkml/2015/9/12/243
[2] http://www.freiesmagazin.de/freiesMagazin-2015-08
[3] http://www.pro-linux.de/-0h2158ae
[4] https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/fs/ext3?id=c290ea01abb7907fde602f3ba55905ef10a37477
[5] https://de.wikipedia.org/wiki/IPv6
[6] https://de.wikipedia.org/wiki/InfiniBand
[7] https://lkml.org/lkml/2015/9/20/159
[8] https://lkml.org/lkml/2015/9/27/42
[9] https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/drivers?id=cc8b8faea417bd264d23fa5d017da52b75bbdf94
[10] https://lkml.org/lkml/2015/10/4/128
[11] https://de.wikipedia.org/wiki/Kernel-based_Virtual_Machine
[12] https://plus.google.com/u/0/+gregkroahhartman/posts/eXcYGjSLVJo
[13] https://de.wikipedia.org/wiki/Greg_Kroah-Hartman
[14] http://www.linuxdriverproject.org
[15] http://sarah.thesharps.us/2015/10/05/closing-a-door/
[16] http://www.pro-linux.de/-0h215928

Autoreninformation

Mathias Menzer (Webseite) behält die Entwicklung des Linux-Kernels im Blick, um über kommende Funktionen von Linux auf dem Laufenden zu bleiben und immer mit komplizierten Begriffen dienen zu können.

Shell Command Injection – Wie fremder Text in das Terminal gelangt

von Bernd Dietzel

Man möchte sicher nicht, dass ein Fremder einen Befehl in das Terminal eingibt. Und doch kann es genau dazu kommen, wenn Skripte auf dem System Sicherheitslücken aufweisen, die es einem Angreifer erlauben, beliebigen Schadcode auszuführen. Eine dieser Sicherheitslücken ist die sogenannte „Shell Command Injection“ [1] oder auch „Shell Injection“, über die in diesem Artikel informiert werden soll.

Angriffsszenario

Es geht um Text. Nichts anderes als Text. Klingt langweilig? Nein, keinesfalls. Den Weg des Textes vom Angreifer zum Kommandozeileninterpreter, der Shell, dem Terminal oder der Konsole zu verfolgen ist eine mitunter spannende Angelegenheit, die man am besten gleich einmal an einem Beispiel ausprobiert, anstatt lange zu theoretisieren.

Man versetze sich in die Rolle des Angreifers, der ein Kommando auf einen PC einschleusen will. Ein E-Mailprogramm soll in diesem Szenario eine Sicherheitslücke aufweisen, die ausgenutzt wird. Angenommen, das E-Mailprogramm wäre in Python geschrieben und verwendet zur Dateivorschau von Anhängen das dazu vorgesehene Standard-Pythonmodul „MailCap“ [2].

Hinweis: Da ein vollständiger Quellcode eines E-Mailprogramms diesen Artikel sprengen würde, wird in dem unten stehenden Beispiel der Dateiname **DATEINAME** fest einprogrammiert. Normalerweise käme der Dateiname über den Angreifer in das Programm.

Ein Abschnitt des Python-Mailprogramms sähe somit in dem stark vereinfachten Beispiel so aus:

```
import mailcap , os
d=mailcap.getcaps()
DATEINAME="';ls;#';ls;#.mp4"
cmd,m=mailcap.findmatch(d, "audio/mpeg4", filename=DATEINAME)
os.system(cmd)
```

Listing 1: shell_code_injection_1.py

Hinweis: Das harmlose Kommando **ls** dient in diesem Beispiel als Platzhalter für das bösartige Kommando des Angreifers. Das Semikolon **;** trennt zwei Befehle voneinander, das Rautezeichen **#** kommentiert den hinteren Teil aus.

Erklärung der Lücke

Das Pythonmodul **mailcap** sucht hier mit **getcaps()** nach einem Programm zur Dateivorschau für die mp4-Datei in **DATEINAME**. Es liefert in **cmd** ein Kommando zurück, dass man mit **os.system(cmd)** an die Shell gibt, um das zum Dateityp passende Anzeigeprogramm zu starten.

So weit, so gut. Wo ist nun das Problem, wird man sich fragen? Sieht doch alles ganz normal aus bis vielleicht auf den merkwürdigen Dateinamen **';ls;#';ls;#.mp4**. Schaut man sich die Sache genauer an, so erkennt man bald, dass hier der **DATEINAME** einfach mit einem anderen Text zusammen kopiert wird.

Es wird mit **mailcap.findmatch()** zunächst geschaut, wie das Kommando zum Ansehen einer mp4-Datei lautet. Das könnte dann je nach verwendeter Linux Distribution entweder **vlc '%s'** oder aber **vlc %s** lauten. (VLC wird hier beispielhaft als Viewer benutzt. Auf manchen System ist es mplayer, Totem oder ein ganz anderes Programm.)

Nun wird **DATEINAME** anstelle von **%s** eingesetzt, was zu einer der beiden Kommandozeilen in der Variable **cmd** führt:

```
> print cmd
vlc '';ls;#';ls;#.mp4'   # Variante 1
vlc ';ls;#';ls;#.mp4     # Variante 2
```

Unabhängig davon, welche der beiden Kommandozeilen zum Tragen kommt, wenn dieser „blind“

zusammenkopierte Text mit **os.system(cmd)** auf eine Kommandozeile gerät, so wird nicht nur VLC ausgeführt, sondern ungewollt auch noch das aktuelle Verzeichnis mit **ls** ausgegeben.

Dass VLC dabei eine Fehlermeldung ausgibt, ist dem Angreifer hierbei gleichgültig, denn es wurde von ihm bereits erfolgreich das Kommando **ls** in die Shell injiziert.

Hinweis: Wer das (in dem Fall harmlose) Verhalten testen möchte, kann entweder mittels **python** eine Python-Konsole öffnen und den Code ausführen oder speichert den Code in einer Datei und ruft diese per **python shell_code_injection_1.py** auf. Nach dem Schließen von VLC sieht man die Ausgabe des aktuellen Verzeichnisses in der Shell.

Dies war es, was hier zu beweisen war. Indem der Dateiname vom Angreifer einfach entsprechend präpariert wurde, kann ein beliebiges Kommando beim E-Mailempfänger ausgeführt werden, obwohl sich dieser die angehängte Datei doch nur hat ansehen wollen. Der Inhalt der Datei spielt hierbei keine Rolle, es ist der Text des Dateinamens der den Schadcode enthält.

Fortsetzung der Lücke

Nun würden erfahrene Programmierer einwenden, dass es üblich ist, Dateinamen und Pfade zu „quoten“, also in Anführungszeichen zu setzen, um damit ggf. bösartigen Text zwischen Hochkommata „einzusperren“, sodass dieser nicht mehr ausgeführt wird. Jedoch lässt sich das mitunter umgehen, wenn der umgebende Text, in den die Ausgabe von **quote()** eingefügt werden soll, bereits Hochkommata enthält, z. B.:

```
import mailcap , os
try:
  from shlex import quote
except ImportError:
  from pipes import quote
d=mailcap.getcaps()
DATEINAME=quote(";ls;#.txt")
cmd,m=mailcap.findmatch(d, "text/plain", filename=DATEINAME)
os.system(cmd)
```

Listing 2: shell_code_injection_2.py

In diesem Fall wird nicht das Programm zum Öffnen einer mp4-Datei gesucht, sondern für eine Textdatei. Sofern MailCap als Kommando **less '%s'** zum Anzeigen einer Textdatei findet, führt dies in **cmd** zu folgendem Inhalt:

```
> print cmd
less '';ls;#.txt''
```

Das führt nun trotz – oder gerade wegen – der Verwendung von **quote()** unerwünschterweise dazu, dass das aktuelle Verzeichnis mit **ls** auflistet wird, nachdem die sinnfreie Textausgabe von **less ''** mit der Taste Q beendet wurde.

Zu hoffen bleibt, dass kein Mailprogramm das Pythonmodul MailCap in der zuvor dargestellten Weise verwendet.

Fazit

Es gibt ein paar Tipps, die man befolgen sollte, damit Angreifer nicht eine Lücke im eigenen Code ausnutzen können:

1. Niemals „blind“ Text zusammenkopieren und ohne vorherige Prüfung an die Shell übergeben.
2. Ausführen von Shell-Befehlen aus den beiden Python-Modulen **os** und **commands** vermeiden.
3. Wenn Shell-Befehle notwendig sind, dann diese mit **subprocess.Popen()** ausführen. Dieses Python-Modul ist resistenter gegen Injektionen.
4. Der Befehl **quote()** kann bei unsachgemäßem Gebrauch ungewollt Beihilfe zum Ausbruch aus dem Gefängnis der Hochkommata leisten.
5. Alle Texte, die von einem Angreifer erreichbar sind, gegebenenfalls temporär vor Benutzung eines Shell-Befehls durch harmlose Texte austauschen, um diese erst nach der Shell-Opera-

tion wieder durch den ursprünglichen Text zu ersetzen.
6. Auch vermeintlich sicheren Standard-Python-modulen sollte man keineswegs uneingeschränktes Vertrauen entgegenbringen.

Das in dem Artikel demonstrierte Problem mit dem Standard-Pythonmodul MailCap wurde bereits gemeldet, zu dem es auch einen Workaround gibt [3].

Auf Launchpad findet man eine Liste derjenigen Programme, die für Shell Command Injection anfällig sind [4].

LINKS

[1] https://en.wikipedia.org/wiki/Code_injection#Shell_injection
[2] https://docs.python.org/2/library/mailcap.html
[3] http://bugs.python.org/issue24778
[4] https://bugs.launchpad.net/~l-ubuntuone1104

Autoreninformation

Bernd Dietzel (Webseite) ist 2010 dank freiesMagazin zur Programmiersprache Python gekommen und programmiert seitdem in Python, z. B. Demos für seinen YouTube-Kanal.

“Unpickable” © by Randall Munroe (CC-BY-NC-2.5), http://xkcd.com/916/

Firefox OS 2.5 – Update des Mozilla Flame

von Dominik Wagenführ

In freiesMagazin 01/2015 [1] wurde das Mozilla Flame vorgestellt. Als Betriebssystem setzt das Gerät auf das hauseigene Firefox OS [2], auf welches auch im Test eingegangen wird. Neben der recht alten Version 1.3 wurde ein Update auf Firefox OS 2.0 vollzogen und vorgestellt, was aber nicht überzeugen konnte. Dieser Artikel zeigt, wie ein manuelles Update auf Version 2.5 aussieht und welche Neuerungen im Betrieb auffallen.

Die Aktualisierung des Mozilla Flame orientiert sich dabei an der englischsprachigen Anleitung zum Aktualisieren des Systems [3].

Vorbereitungen

Bevor man die Aktualisierung von Firefox OS auf dem Mozilla Flame startet, müssen einige Vorbereitungen getroffen werden.

Zuerst benötigt man eine Verbindung zum Flame, um Systemdaten direkt aufspielen zu können. Dies geschieht mittels der Android-Tools ADB und Fastboot, welche über die Pakete **android-tools-adb** und **android-tools-fastboot** (getestet in Ubuntu 14.04) installiert werden können.

Danach muss man die Verbindung mit ADB in Firefox OS erlauben. Die Einstellung dazu findet man im Entwicklermenü. In Firefox OS 1.3 ist dies unter „*Einstellungen* → *Geräteinformationen* → *Mehr Informationen* → *Entwickler*“ [4]. Dort aktiviert man das „*Debbuging über USB*“.

ADB-Aktivierung (hier in Firefox OS 2.5).

Verbindung herstellen

Jetzt kann man eine erste Verbindung zum Gerät herstellen. Hierfür verbindet man dieses mit einem USB-Kabel mit dem Rechner. Ein **`lsusb`** sollte das Smartphone anzeigen:

```
$ lsusb
Bus 002 Device 003: ID 05c6:9025 ↷
Qualcomm, Inc. Qualcomm HSUSB Device
```

Falls nicht, muss die Verbindung noch einmal geprüft werden.

Als nächstes muss man unter Linux noch eine udev-Regel einfügen, damit man genügend Rechte hat, um sich mit dem Gerät verbinden und Daten aufspielen zu können [5]. Hierfür editiert man die Datei **`/etc/udev/rules.d/android.rules`** mit Root-Rechten bzw. legt diese an. Dort fügt man folgende zwei Zeilen hinzu:

```
SUBSYSTEM=="usb", ATTR{idVendor}=="05c6↷
", MODE="0666", GROUP="plugdev"
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1↷
", MODE="0666", GROUP="plugdev"
```

Die erste Zeile ist dabei für das Flame bei einer normale USB-Verbindung. Sobald das Gerät aber in den Fastboot-Modus geht, wechselt es die Vendor-ID (in dem Fall ist es die ID für Google), welches durch die zweite Zeile dargestellt wird.

Nach dem Eintragen der Zeilen und Speichern der Datei sollte der udev-Dämon neu gestartet werden (der exakte Befehl kann von System zu System varieren, hier für Ubuntu):

```
# service udev restart
udev stop/waiting
udev start/running, process 7244
```

Danach kann man prüfen, ob das Flame mit ADB ansprechbar ist:

```
$ adb devices
* daemon not running. starting it now on↷
 port 5037 *
```

```
* daemon started successfully *
List of devices attached
f30ecc6b    device
```

Falls dies eine Fehlerausgabe oder kein Ergebnis liefert, kann man den ADB-Aufruf mit Root-Rechten versuchen. Funktioniert dies, ist die udev-Regel nicht korrekt eingetragen bzw. wurde nicht übernommen.

Backup erstellen

Bevor man das Update auf Firefox OS 2.5 wirklich einspielt, sollte man die Daten sichern. Dies umfasst zunächst einmal die Systempartition [6]:

```
$ mkdir Flame_Backup_20150822
$ adb pull /system Flame_Backup_20150822↷
/system
pull: building file list...
...
1169 files pulled. 0 files skipped.
2104 KB/s (283259275 bytes in 131.439s)
$ adb pull /data Flame_Backup_20150822/↷
data
pull: building file list...
0 files pulled. 0 files skipped.
```

Obige Befehle sichern aber nur das System und dienen der Wiederherstellung, falls das Update schief läuft. Weitaus wichtiger sind persönliche Daten, die man nicht verlieren möchte bzw. nach dem Update neu einspielen will. Dazu zählen z. B. installierte Apps, die Kontakte, empfangene SMS, Bilder und mehr. Hierfür gibt es auf GitHub ein Tool, welches man sich zuerst holen sollte [7]:

```
$ git clone https://github.com/Mozilla-TWQA/B2G-flash-tool
Nach »B2G-flash-tool« wird geklont
remote: Counting objects: 2203, done.
remote: Total 2203 (delta 0), reused 0 (delta 0), pack-reused 2203
Objekte werden empfangen: 100% (2203/2203), 979.80 KiB | 899.00 KiB/s, done.
Unterschiede werden aufgelöst: 100% (1234/1234), done.
Verbundenheit wird überprüft ... Fertig.
```

Danach kann man seine Daten sichern:

```
$ cd B2G-flash-tool
$ python backup_restore_profile.py -b --sdcard
2015-08-22 16:44:39,201 - utilities.adb_helper - INFO - adbd is already running as↷
 root
2015-08-22 16:44:39,202 - __main__ - INFO - Target device [None]
2015-08-22 16:44:39,202 - __main__ - INFO - Stop B2G.
2015-08-22 16:44:39,299 - __main__ - INFO - Backing up profile...
2015-08-22 16:44:39,299 - __main__ - INFO - Backing up Wifi information...
2015-08-22 16:44:39,423 - __main__ - INFO - Backing up /data/b2g/mozilla to /tmp/↷
backup_restore_1ASnpe/b2g-mozilla/ ...
2015-08-22 16:44:44,428 - __main__ - INFO - Backing up /data/local to /tmp/↷
backup_restore_1ASnpe/data-local/ ...
2015-08-22 16:45:05,853 - __main__ - INFO - Removing Mozilla webapps: [/tmp/↷
backup_restore_1ASnpe/data-local/webapps/marketplace.firefox.com]
2015-08-22 16:45:05,854 - __main__ - INFO - Backup profile done.
2015-08-22 16:45:05,854 - __main__ - INFO - Backing up SD card...
2015-08-22 16:45:05,917 - __main__ - INFO - Backup: /sdcard/ to /tmp/↷
backup_restore_1ASnpe/sdcard/
2015-08-22 16:45:14,357 - __main__ - INFO - Backup SD card done.
2015-08-22 16:45:14,358 - __main__ - INFO - Copy profile from [/tmp/↷
backup_restore_1ASnpe] to [mozilla-profile].
2015-08-22 16:45:14,505 - __main__ - INFO - Start B2G.
```

Die gesicherten Daten liegen nach dem Vorgang im Ordner **mozilla-profile**. Die Option **--sdcard** sichert auch den Inhalt der SD-Karte ab, wenn man deren Inhalt nicht verlieren will, wo-

bei dies im Test nicht funktionierte. Alle Bilder der Kamera waren nach dem Update und Einspielen der Sicherung weg.

Hinweis: Ggf. meldet das Skript, dass die Rechte zum Sichern der Daten nicht ausreichen. Dann muss man das Skript mit Root-Rechten starten.

Update auf Firefox OS 2.5

Auf der Flame-Update-Webseite [3] findet man vorgefertigte Images für das Flame. Das im Artikel „Fuchs in Flammen: Mozilla Flame im Test“ in freiesMagazin 01/2015 [1] vorgestellte Firefox OS 2.0 basierte damals auf dem inzwischen veralteten „Base image v188“, welches nicht mehr empfohlen wird. Stattdessen gibt es ein „Base image v18D mit Firefox OS 2.0 oder ein „Base image v18D nightly v4“ mit Firefox OS 2.5. Der Zusatz „*The very latest nightly archive*“ auf der Webseite irritiert, da das Nightly-Image v4 nicht wirklich nächtlich aktualisiert wird, sondern vom 20. Juli 2015 stammt.

Hinweis: Das „Base image v18D nightly v3“ ist vom 9. Juni 2015. Man kann dieses auch verwenden, sollte aber aufpassen, da der entpackte Ordner irrtümlich den Titel **`v18D_nightly_v5`** trägt.

Neues Image einspielen

Zuerst lädt man das Base-Image-Update herunter und entpackt es:

```
$ wget http://cds.w5v8t3u9.hwcdn.net/v18D_nightly_v4.zip
$ unzip v18D_nightly_v4.zip
```

Danach geht man in den Ordner und startet das Update. Bevor man das tut, sollte man sich vergewissern, dass man ein Backup seiner Daten vorliegen hat!

```
$ cd v18D_nightly_v4
$ ./flash.sh
```

Jetzt startet der Flashvorgang. Währenddessen wird entweder das Logo von Fastboot oder ThunderSoft angezeigt. Die Ausgabe sieht in etwa so aus:

```
* daemon not running. starting it now on↷
 port 5037 *
* daemon started successfully *
List of devices attached
f30ecc6b  device
Partition table...
< waiting for device >
target reported max download size of ↷
301989888 bytes
sending 'partition' (33 KB)...
...
Done...
rebooting...

finished. total time: 0.005s
Just close the windows as you wish.
```

Das Telefon sollte nach dem Vorgang automatisch neu starten und den Ersteinrichtungsdialog zeigen, in welchem man die Sprache einstellt, mit dem WLAN verbinden kann etc.

Gesicherte Daten zurückspielen

Bevor man die zuvor gesicherten persönlichen Daten zurückspielt, muss man erneut das Debugging über USB aktivieren. Dies findet man in Firefox OS 2.5 (was als 3.0.0.0-prerelase angezeigt wird) unter „*Einstellungen* → *Entwickler* → *Debugging über USB*“. Dort wählt man „*Debugging über USB*“ aus.

Danach geht man zurück in den Ordner, in dem das **`mozilla-profile`** zuvor gesichert wurde, und spielt dieses ein:

```
$ cd B2G-flash-tool
# python backup_restore_profile.py -r --↷
sdcard
```

Das Telefon sollte dann neu starten und nach der Pin-Eingabe sollten alle Daten im Prinzip wieder vorhanden sein. Zumindest in meinem Fall waren zwar die Apps, WLAN-Einstellungen und Kontakte noch vorhanden, aber die gespeicherten Bilder der SD-Karte wurden nicht gesichert/zurückgespielt.

Hinweis: Sollte nach dem Update auf die neue Firefox-OS-Version ein Systemupdate anstehen, sollte man genau überlegen, ob man dieses installieren will, da es einen gravierenden Bug mit dem Lautstärkeregler gibt, der sich nicht mehr korrekt je nach Kontext umschaltet. Das führt dazu, dass man ggf. die Lautstärke für Anrufe nicht mehr einstellen kann, weil dafür die Lautstärke der Wecker-App verstellt wird. Im Gegenzug führt

das dazu, dass die Wecker-App keinen Mucks mehr macht und man verschläft [8].

Erfahrungen mit Firefox OS 2.5

Die gröbsten Fehler, die das „Base image v188“ noch hatte, wurden inzwischen ausgemerzt. So hält der Akku bei normalen Betrieb wieder 5 bis 7 Tage und ist nicht nach wenigen Stunden leer.

Von der Gestaltung hat sich das Hauptmenü wieder verändert – diesmal zum Positiven: Die aus Firefox OS 1.3 bekannten seitlichen Menüs und die Schnellwahl sind immer noch verschwunden, alle Apps werden durch Linien untereinander in einem Bildschirm dargestellt. Der Vorteil ist, dass diese einzelnen Bereiche durch ein kleines Häckchen oben rechts eingeklappt werden können und so weniger Platz wegnehmen. So kann man seine Haupt-Apps in den oberen Bereich legen und alles andere in den Rest. Die Apps lassen sich aber auch ohne Aufklappen des Bereichs durch einen simplen Klick (mit genauem Zielen) starten.

Die Optik und Farbgebung hat sich ebenfalls geändert, es wird jetzt auf ein hellblau/cyan-farbene Farbpalette gesetzt. Auch neu ist eine Art Wackeleffekt, wenn man an das Ende oder den Anfang einer Liste scrollt. Das sieht nett aus, hat aber sonst keinerlei Effekt – außer dass man weiß, dass man am Ende angekommen ist.

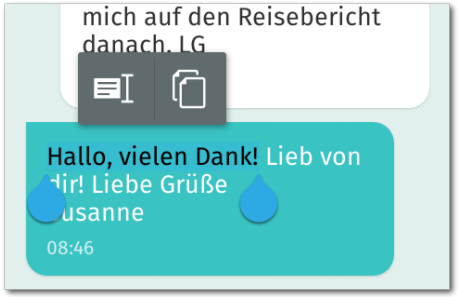
Was in Firefox OS 1.3 noch per Entwickler-Menü aktiviert werden musste, ist nun Standard, kann in den Einstellungen aber deaktiviert werden: der

Das neue Firefox-OS-Menü mit blauer Benachrichtigungsleiste.

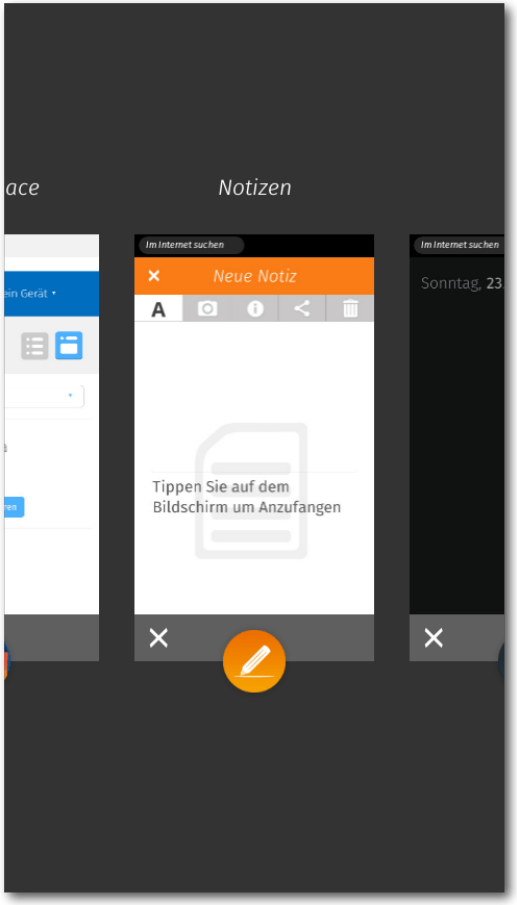
Wechsel zwischen geöffneten Apps mittels dem Wischen über den linken oder rechten Rand. Alternativ kann man auch immer noch den Home-Button lange gedrückt halten, um eine Übersicht aller offenen Apps zu halten. Hier wurde das Layout übersichtlicher gestaltet. Apps können nun durch einen Klick auf ein „X“ oder durch Hinausschieben über den oberen Rand geschlossen werden.

Benachrichtigungen werden nicht mehr in der oberen Menüleiste angezeigt, sondern durch einen dezenten blauen Balken am oberen Bildschirmrand. Ebenfalls neu ist die Copy&Paste-Funktion, mit der man Text markieren, kopieren und einfügen kann.

Dafür gab es bei der deutschen Tastatur eine negative Entwicklung. Wurde in Firefox OS 1.3 mit einem langen Druck auf einen Vokal noch per Standard ein Umlaut angewählt (aus „u“ wurde „ü“), öffnet sich nun nur noch das Sonderzeichenmenü und man muss manuell den Umlaut auswählen. Das kostet Zeit beim Schreiben von Texten. Die Autokorrektur hilft dann dabei, dass aus „Gruse“ ein „Grüße“ wird, weil es zu umständlich ist, die Sonderzeichen anzuwählen. Leider werden auch sinnvolle Worte korrigiert, so wird aus „dir“ immer ein „die“, was auf die Dauer nervt. Immerhin kann man die Autokorrektur deaktivieren.



Markieren, Kopieren und Einfügen von Text ist nun möglich.



Geöffnete Apps in der Übersicht.

Ansonsten ist das System „natürlich“ immer noch nicht fehlerfrei. So kann man den Bildschirmhintergrund nicht mehr ändern. Bei einem Klick auf die Option passiert einfach nichts. Es wurden auch andere Einstellungen gefunden, die plötzlich keine Funktion mehr haben. Auch sind nicht alle Funktionen übersetzt, sodass sich mitunter ein Deutsch/Englisch-Gemisch ergibt.

Fazit

Firefox OS in Version 2.5 ist immer noch (bzw. wieder) benutzbar, auch wenn es weiter Fehler hat. Nach einem Jahr in Benutzung schreitet die Entwicklung des Systems zwar voran, aber die Liste an Fehlern bleibt lang. Ein Systemupdate darf z. B. nicht dafür sorgen, dass plötzlich der Lautstärkeregler nicht mehr korrekt funktioniert. Fehler können passieren, aber wenn das Update einen Monat nach Meldung des Fehlers diesen nicht behebt, ist das kein gutes Zeichen.

Daneben gibt es zwischen Versionen zahlreiche inkompatible Änderungen. Neben neuem Menülayout und dem Verschieben von Optionen wurde z. B. die Screenshot-Funktion geändert. Sie war schon in Firefox OS 1.3 nicht intuitiv mit gleichzeitigem Drücken von Hauptmenü- und Ausschalt-Knopf [9]. Nun ist es der Ausschalt-Knopf und Lautstärke runter, die man gleichzeitig drücken muss, was aber beschrieben ist.

Neue Fehler in jedem Stand zeugen ebenfalls nicht davon, dass die Qualitätskontrolle bei Mozilla richtig funktioniert. Es ist schade, dass ein so vielversprechendes System nicht ausgereizt wird. Ggf. kommt dies noch, aktuell ist das System aber eher Enthusiasten zu empfehlen. Standardanwender sollten auf ein bewährteres System – konkret Android respektive CyanogenMod [10] zugreifen.

LINKS

[1] http://www.freiesmagazin.de/freiesMagazin-2015-01
[2] https://www.mozilla.org/de/firefox/os/
[3] https://developer.mozilla.org/en-US/Firefox_OS/Phone_guide/Flame/Updating_your_Flame
[4] https://developer.mozilla.org/en-US/Firefox_OS/Debugging/Developer_settings
[5] https://developer.mozilla.org/en-US/Firefox_OS/Firefox_OS_build_prerequisites#For_Linux_configure_the_udev_rule_for_your_phone
[6] https://developer.mozilla.org/en-US/Firefox_OS/Firefox_OS_build_prerequisites#Back_up_the_phone_system_partition
[7] https://developer.mozilla.org/en-US/Firefox_OS/Phone_guide/Flame/Updating_your_Flame#Backing_up_and_restoring_your_Flame_data
[8] https://bugzilla.mozilla.org/show_bug.cgi?id=1189175
[9] https://support.mozilla.org/en-US/kb/take-screenshots-firefox-os
[10] http://www.cyanogenmod.org/

Autoreninformation

Dominik Wagenführ (Webseite) nutzt das Flame mit Firefox OS aus Open-Source-Überzeugung und weil er Android nicht mag, auch wenn das Telefon ihn manchmal etwas ärgert.

Teilen *Kommentieren*

Labdoo: Ausgediente Laptops mit Linux für Bildungseinrichtungen

von Sujeevan Vijayakumaran

Die Hilfsorganisation Labdoo [1] setzt sich dafür ein, alte und ausgediente Laptops aus Industrieländern aufzubereiten, mit Lubuntu [2] zu bespielen und anschließend an Bildungseinrichtungen in Entwicklungsländern zu verteilen.

Die Idee

Die Idee hinter Labdoo ist, dass man Laptops, die in Industrieländern wie Deutschland, nicht mehr genutzt werden, so aufbereitetet, dass sie trotzdem weiter genutzt werden können, um Kindern und Jugendlichen in Bildungseinrichtungen in Entwicklungsländern den Zugang zu Computern und idealerweise dem Internet zu ermöglichen.

Viele kennen es vielleicht, dass man selbst, im Bekannten- oder Verwandtenkreis Laptops über einige Jahre nutzt, die später durch neuere, bessere Geräte ersetzt werden. Die alten Laptops funktionieren zwar häufig noch, sind aber mittlerweile langsam und veraltet. Kleine und größere Defekte werden stellenweise als Grund genommen, sich neue Hardware anzuschaffen. Genau an diesen Punkt setzt das Labdoo-Projekt an, denn obwohl die Hardware in Industrieländern von Privatpersonen oder auch von Firmen nicht mehr genutzt werden, funktionieren sie meistens noch oder lassen sich mit ein paar Handgriffen reparieren. Der erste Lebenszyklus des Laptops ist zwar prinzipiell beendet, aber weitergenutzt werden kann er immer noch.

Labdoo macht diese Computer wieder fit, bespielt sie mit Lubuntu als leichtgewichtiges Betriebssystem und organisiert dann den Transport und die Übergabe an Bildungseinrichtungen in Entwicklungsländern – und das möglichst CO2-neutral.

Labdoo Hubs

Die Annahme, Aufbereitung und Verteilung der Laptops wird über sogenannte Hubs organisiert. Die Rechner werden von diesen Hubs entweder selbst eingesammelt oder die Spender können dort ihre Laptops direkt vorbeibringen, sofern man zuvor die Kontaktperson angesprochen hat.

Obwohl Labdoo ein internationales Projekt ist, befinden sich die meisten Hubs in Deutschland, der Schweiz und in Spanien. Innerhalb der Länder gibt es eine verschieden große Anzahl an Hubs und auch diverse Ballungszentren. Im Rhein-Ruhr-Gebiet in Nordrhein-Westfalen findet man bspw. in sehr vielen Städten Hubs und Annahmestellen. Letzteres sind – wie der Name schon verrät – Stellen, wo man Laptops abgeben kann. Die Personen an einer Annahmestelle kümmern sich mit den Zuständigen eines Hubs um die erfolgreiche Übergabe. Wenn man selbst einen Hub betreiben möchte, braucht man mindestens ein Zuhause, wo man auch mal Platz hat, den einen oder anderen Laptop zwischenzulagern. Als Annahmestellen sind besonders Einzelhandelsläden oder andere Firmen sinnvoll, wo man zu Öffnungszeiten die Notebooks abgeben kann.

Edoovillages

Das Wort „Edoovillages“ ist ein Wortspiel aus „Education“ (Ausbildung) und „Village“ (Dorf). Dies sind die Orte, wo die Laptops hingebracht werden, also in der Regel Bildungseinrichtungen in Entwicklungsländern. Teilweise werden aber auch andere Einrichtungen ausgestattet, die keine klassischen Schulen sind, sondern sich um Kinder und Jugendliche kümmern, etwa Waisenhäuser oder Jugendzentren.

Potenzielle Edoovillages können sich an Labdoo wenden, indem sie den Ort, Art der Einrichtung, Anzahl der Schüler und Lehrer und die Anzahl der benötigten Laptops oder auch E-Book-Reader angeben können. Diese Informationen werden auf der Labdoo-Webseite [3] dargestellt, damit für jede Person sichtbar ist, wie viele Geräte schon in ein Edoovillage gebracht worden sind, wie viele auf dem Weg dahin sind und wie viele generell noch gebraucht werden.

Die Laptops

Über die letzten Jahre haben sich zahlreiche Laptops angesammelt, die trotz ihres fortgeschrittenen Alters immer noch funktionsfähig sind. Es eignen sich prinzipiell Laptops ab Baujahr Ende 2000. Zwar werden nicht zwangsläufig nur 15 Jahre alte Laptops ausgeliefert, doch braucht man hin und wieder das eine oder andere Ersatzteil, das man bei reparaturbedürftigen Notebooks einsetzen kann. Neben vollständigen Laptops werden

also auch verschiedene Einzelteile angenommen; dazu gehören auch Netzteile verschiedener Modelle sowie gängige Tastaturen und Mäuse. Viele dieser Teile bekommt man nicht einmal mehr im Einzelhandel – und wenn doch dann zu Mondpreisen. Dies trifft zum Beispiel schon bei alten IDE-Notebook-Festplatten zu.

Jeder Laptop [4] hat einen Status. Zu Beginn bekommt er zunächst den Status „tagged“, in dem er eine eindeutig identifizierbare Nummer bekommt, die auf dem Computer aufgeklebt wird. Falls es schon in einem Hub gelandet ist, erhält es den Zustand „donated“. Im Hub wird der Rechner anschließend aufbereitet, der Status wird dann „sanitized“ genannt. In der Aufbereitungsphase wird der Laptop auf die Funktionsfähigkeit überprüft und defekte Teile gegebenenfalls ausgetauscht. Weiterhin findet auch eine Reinigung des Gerätes und die Installation des Systems statt.

Sobald diese Arbeiten abgeschlossen sind, ist es bereit für den Transport. Dazu werden die Laptops zunächst einem Projekt zugewiesen. Die Zuweisung erfolgt je nachdem, wo sie gebraucht werden und welche Möglichkeiten es für den Transport gibt. Im Anschluss gehen die Laptops in den Transport und werden abgeliefert. Falls es zu Schäden oder Funktionsmängeln kommt, werden die Rechner bei Möglichkeit wieder abgeholt und ordnungsgemäß recycelt.

Jeder Spender kann den Status jederzeit einsehen, man bekommt also volle Transparenz, was mit dem Laptop geschehen ist und wann er sich an welchem Ort befindet. Bei erfolgreicher Übergabe machen die Überbringer ebenfalls Fotos, sodass man nicht nur einen Art „Beweis“ hat, sondern auch in die glücklichen Gesichter der Empfänger schauen kann.

Die Software-Ausstattung

Als Betriebssystem kommt Lubuntu zum Einsatz. Der LXDE-Desktop ist besonders ressourcenschonend und kommt auch mit wenig Arbeitsspeicher und einer schwachen CPU zurecht. Zusätzlich zum Basissystem sind zahlreiche weitere Software-Pakete installiert, darunter hauptsächlich Lern- und Lernspielsoftware, ein Offline-Wiki und Wörterbücher. Jede Software ist möglichst an die Lokalität angepasst, in der die entsprechenden Laptops landen werden. Wenn also Notebooks bspw. nach Indien gehen, werden die Sprachpakete des entsprechenden Gebietes installiert sowie die weiteren Software-Pakete die davon abhängig sind, wie etwa das Offline-Wiki und die Wörterbücher in der entsprechenden Sprache.

Mitmachen

Es gibt viele Möglichkeiten bei Labdoo mitzumachen – und Aufgaben mit wenig und andere mit mehr Zeitaufwand. Die wohl einfachste Möglichkeit dem Projekt zu helfen, ist das Projekt selbst bekannter zu machen, denn sehr viele sind sich nicht bewusst, wie einfach man Laptops für den guten Zweck spenden kann. Eine direkte Spende von Laptops oder Laptop-Kleinteilen gehört ebenfalls zu den einfacheren Möglichkeiten dem Projekt zu helfen. Häufig ist es auch ratsam, beim eigenen Arbeitgeber anzufragen, sofern man nicht weiß, was mit alter Hardware geschieht oder wo Hardware unnötig verschrottet wird. Falls man nicht weiß, ob die vorhandene Hardware gebraucht wird, reicht auch eine kurze E-Mail zum nahegelegenen Hub aus.

Weiterhin kann man selbst einen Labdoo-Hub gründen, Laptops sammeln oder auch nur lagern. Wenn man gerne in Entwicklungs- oder Schwellenländer reist und dann gegebenenfalls die Möglichkeit hat Laptops abzuliefern, ist dies auch eine gute Gelegenheit mitzuhelfen.

Links

[1] https://www.labdoo.org/ GB
[2] http://lubuntu.net/ GB
[3] https://www.labdoo.org/content/edoovillages-dashboard GB
[4] https://www.labdoo.org/content/doojects-dashboard GB

Autoreninformation

Sujeevan Vijayakumaran (Webseite) hat in diesem Jahr selbst einen Labdoo-Trip unternommen und Laptops an eine Schule in Sri Lanka überbracht.

Rezension: Python 3 – Das umfassende Handbuch

von Jochen Schnelle

Python gilt als universelle und leicht zu erlernende Programmiersprache, welche in vielen Bereichen eingesetzt werden kann. Das vorliegende Buch „Python 3 – Das umfassende Handbuch“ [1] bietet einen breit gefächerten Einblick die Welt der Python-Programmierung.

Redaktioneller Hinweis: *Wir danken dem Rheinwerk Verlag für die Bereitstellung eines Rezensionsexemplares.*

Für Anfänger, Fortgeschrittene und Andere

Das Wort „umfassend“ aus dem Titel des Buchs ist hier durchaus wörtlich zu nehmen: Mit einem Umfang von mehr als 1000 Seiten, die sich in 43 Kapitel plus Anhang unterteilen, wird eine breite Themenvielfalt abgedeckt. Und, so viel sei an dieser Stelle schon gesagt, es gelingt dabei auch sehr gut der Spagat zwischen Einsteiger-Themen, Kapiteln für Fortgeschrittene und spezielleren Kapiteln, die bestimmte Themenbereiche vertiefend behandeln.

Leser ohne Programmierkenntnisse werden direkt am Anfang des Buchs abgeholt. Es wird die Philosophie von Python erklärt und eine Einführung in die Grundlagen gegeben. Dazu gehören, neben der Installation von Python, Themen wie Datentypen, Kontrollstrukturen und der Umgang mit Funktionen etc.

Es folgen Kapitel für fortgeschrittene Anfänger, in denen z. B. die eingebauten Funktionen erläutert werden, der Umgang mit Ausnahmefehlern und Iteratoren und Generatoren. Außerdem gibt es ein recht ausführliches Kapitel zur Objektorientierung und der Erstellung eigener Klassen, inklusive Vererbung von Klassen.

Danach gibt es eine Reihe von Kapiteln zu fortgeschrittenen Themen, wie z. B. paralleler Programmierung mit Python, Umgang mit (SQL-)Datenbanken, JSON und XML, Netzwerkkommunikation, Debugging und Schreiben von Tests.

Das letzte Viertel des Buchs ist mit „weiterführende Themen“ überschrieben. Hier blicken die Autoren über den Tellerrand der Standardinstallation von Python hinaus. So gibt es z. B. Kapitel zum Programmieren von grafischen Benutzeroberflächen mit Qt und PyQt, dem Erstellen einer Webanwendung mit Django sowie dem Umgang mit dem Python-Modul Pillow zum Anzeigen und Manipulieren von Grafiken und Bildern.

Das komplette Inhaltsverzeichnis des Buchs, welches eine guten Überblick über alle behandelten Themen gibt, ist auf der dem Buch zugehörigen Webseite des Verlags zu finden [1].

Inhalt und Verständlichkeit

Alle Inhalte des Buchs sind gut und ausführlich dargestellt. In ganz vereinzelten Fällen neigen die Erklärungen etwas zur Langatmigkeit (zum Beispiel im Kapitel zu Klassen und Klassenvererbung). Insgesamt ist das Buch aber gut und flüssig zu lesen.

Alle Themen werden mit passenden Codebeispielen unterlegt, was der Verständlichkeit natürlich sehr zugute kommt. Erwähnenswert ist auch, dass die Autoren kaum „konstruierte“ Codebeispiele verwenden, sondern diese „brauchbar“ und recht praxisnah sind. So wird zum Beispiel im Kapitel zum Webframework Django ein kleiner, voll funktionsfähiger, webbasierter Newsticker programmiert.

Das Buch ist diesen Sommer in der vierten, aktualisierten Auflage erschienen. Und offensichtlich haben die Autoren den Inhalt wirklich erst vor kurzem aktualisiert, dann alle Kapitel behandeln die gerade aktuellen Programmversionen (zum Beispiel von Django, PyQt etc.). Als Python-Version wird durchgehend auf Python 3.4 Bezug genommen.

Buchinformationen	
Titel	Python 3 – Das umfassende Handbuch [1]
Autor	Johannes Ernesti, Peter Kaiser
Verlag	Rheinwerk Computing, 4., aktualisierte und erweiterte Auflage 2015
Umfang	1032 Seiten
ISBN	978-3-8362-3633-1
Preis	39,90 Euro (Print), 34,90 Euro (E-Book)

Fazit

Das Buch „Python 3 – Das umfassenden Handbuch“ umfasst ein breites Spektrum an Themen, welche sehr gut den Bereich von Python-Einsteigern bis -Fortgeschrittenen abdeckt. Der Inhalt ist breit gefächert und zugleich auch gut erklärt. Bei einem Umfang von mehr als 1000 Seiten und einem Preis von 39,90 Euro stimmt auch das Preis-/Leistungsverhältnis. Das Buch ist definitiv das beste mir bekannte, deutschsprachige Python-Buch. Es kann durchweg empfohlen werden.

Links

[1] https://www.rheinwerk-verlag.de/python-3_3789/

Autoreninformation

Jochen Schnelle (Webseite) nutzt Python als bevorzugte Programmiersprache.

Teilen *Kommentieren*

“Skynet” © by Randall Munroe (CC-BY-NC-2.5), http://xkcd.com/1046/

Rezension: Ha3k3ln+Str1ck3n für Geeks

von Christina Möller

Nachdem die sogenannte Gattung Geek im Jahre 2011 von O'Reilly mit einem „Kochen für Geeks" und im darauf folgendem Jahr mit dem Buch „Fitness für Geeks" – man beachte die Reihenfolge – versorgt worden war, folgte 2013 ein neuer Vorschlag für ein Hobby: „Ha3k3ln+Str1ck3n für Geeks" [1], geschrieben von Verena Kuni.

Wer sich an dieser Stelle über den Unterschied zwischen Geeks und Nerds informieren möchte, dem sei die nicht 100 Prozent ernstgemeinte Infografik empfohlen [2].

Redaktioneller Hinweis: *Wir danken O'Reilly für die Bereitstellung eines Rezensionsexemplares.*

Was steht drin?

Das Buch gliedert sich in neun Kapitel. Das Inhaltsverzeichnis sowie ein gratis Kapitel können auf der Webseite des Verlags eingesehen werden [3].

In den ersten drei Kapiteln werden die Grundlagen zum Häkeln und Stricken nahe gebracht. Nach dem ersten Kapitel weiß der Leser, warum Wolle kratzen kann, und hat einen groben Überblick, welche Materialien (u. a. Magnetband, Draht oder Kabel) sich neben Wolle noch für die Handarbeit eignen. Die folgenden zwei Kapitel sind als Einführung zum Häkeln bzw. Stricken zu verstehen. Techniken und Code (Strick-/Häkelschriftzeichen) werden erklärt und auf viele Anleitungen im Internet wird verwiesen.

Das Kapitel vier „Welt am Draht", das auch als Leseprobe zur Verfügung steht, bringt nicht nur Geek-Augen zum Leuchten, sondern auch die zuvor hergestellten Gadgets oder Mützen.

Um die Mustererkennung und speziell das Generieren von Mustern am Computer geht es im fünften Kapitel.

Ein gehäkelter Tux (Häkelmuster [4]).

Für all die Geeks, die nicht mehr hinterherkommen ihre kreativen Ideen umzusetzen, werden im sechsten Kapitel die zahlreichen Möglichkeiten durch die Verwendung von Strickmaschinen vorgestellt.

In den folgenden zwei Kapiteln werden Netzwerke/Strickzirkel (Kapitel sieben) und der Umgang mit Löchern/Motten (Kapitel acht) beschrieben.

Das Buch schließt mit einem „Service"-Kapitel ab. Dort wird endlich auf die Internetseite verwiesen, auf die der lesende Geek unter Umständen die ganze Zeit gewartet hat, um nicht alle Links abtippen zu müssen. Allerdings sorgt direkt nach dem Öffnen der Webseite ein „Coming soon" für Ernüchterung. Auf Nachfrage teilte Frau Kuni mit, dass sie hofft, die Seite bald freischalten zu können.

Buchinformationen	
Titel	Häkeln + Stricken für Geeks [1]
Autor	Verena Kuni
Verlag	O'Reilly, 1. Auflage September 2013
Umfang	400 Seiten
ISBN	978-3-86899-356-1
Preis	24,90 Euro (Print), 20,00 Euro (E-Book)

Zielgruppe

Aus dem Titel lässt sich bereits die Zielgruppe ableiten: Geeks. Der Untertitel „Wissenswertes, Ideen & Inspirationen" sollte nicht überlesen werden. Geeks, die in den ersten Kapiteln eine bebilderte Anleitung zu den Grundlagen (Maschen anschlagen, aufnehmen, oder ähnliches) erwarten, werden auf die entsprechenden Internetseiten verwiesen.

Für bereits mit der Materie Handarbeit vertraute Geeks wurden viele inspirierende Links zusammengetragen.

Wie liest sich das Buch?

Im Buch werden eine Reihe wiederkehrender „Formate" verwendet, die eingangs aufgelistet und erläutert werden. Beispielsweise Realitätskoeffizient, DIY (Do it yourself – Mit Anleitung zum Selbermachen) und GEEK GOSSIP.

Der Text an sich liest sich fließend, ist an einigen Stellen aber sehr in die Länge gezogen und versucht an anderen Stellen, dem Titel und der Zielgruppe gerecht zu werden.

Nahezu auf jeder zweiten Seite befindet sich eine Farbabbildung mit Ideen für Handarbeiten, die zum Teil durchaus witzig sind. Allerdings gibt es darunter auch viele Abbildungen, auf die man hätte verzichten können: Weiß ein Geek wirklich nicht, wie ein Schaf oder eine Spinne aussieht? Bei etlichen Abbildungen ist die Qualität nicht ausreichend für ein Buch, sondern wirken eher wie Schnappschüsse für den Privatgebrauch.

Die diversen Links, auf die in mehreren Kapiteln verwiesen wird, sind sicherlich als kritisch im Hinblick auf das Verfallsdatum des Buches anzusehen. Es bleibt zu hoffen, dass die Internetseite zur Publikation bald online ist.

Fazit

Das Buch liefert eine Vielzahl an Ideen, was aus Wolle oder anderen Materialien alles geschaffen werden kann. Von kleinen Amigurumis [5] über Loops nach dem Prinzip des Möbiusbandes bis hin zu größeren und komplexeren Objekten ist für jedes Level an Können und Ausdauer etwas dabei.

Als digitale Variante, in welcher der Leser per Linkauswahl direkt zur gewünschten Seite im Internet gelangt, ist das Buch sicherlich nützlicher. In der gedruckten Form stören die Link-Listen an einigen Stellen leider den Lesefluss.

Dennoch ist es ein Buch, an dem manche Leser ihre Freude haben werden. Zudem ist es das einzige deutschsprachige Buch zu dieser Materie.

Redaktioneller Hinweis: Da es schade wäre, wenn das Buch bei Christina Möller nur im Regal steht, wird es verlost. Die Gewinnfrage lautet:

„*Worauf sollte man achten, wenn die gehäkelten/-gestrickten Handschuhe das Smartphone bedienen sollen?*"

Die Antwort kann bis zum **18. Oktober 2015, 23:59 Uhr** über die Kommentarfunktion oder per E-Mail an redaktion@freiesMagazin.de geschickt werden. Die Kommentare werden bis zum Ende der Verlosung nicht freigeschaltet. Das Buch wird unter allen Einsendern, die die Frage richtig beantworten konnten, verlost.

Links

[1] http://www.oreilly.de/catalog/haekelnkgger/
[2] http://t3n.de/news/geek-nerd-infografik-368069/geek-vs-nerd/
[3] http://www.oreilly.de/catalog/haekelnkgger/chapter/ch04.pdf
[4] http://kvalitid.dk/opskrift-pa-haeklet-linux-pingvin-tux/
[5] https://de.wikipedia.org/wiki/Amigurumi

Autoreninformation

Christina Möller hat sich mit Videos auf YouTube Stricken und Häkeln beigebracht.

Teilen *Kommentieren*

Rezension: Professionell entwickeln mit JavaScript

von Sujeevan Vijayakumaran

Das Buch „Professionell entwickeln mit JavaScript“ [1] des Autors Philip Ackermann behandelt grundsätzlich das Entwickeln von Anwendungen mit der Programmiersprache JavaScript.

Redaktioneller Hinweis: *Wir danken dem Rheinwerk Verlag für die Bereitstellung eines Rezensionsexemplares.*

Was steht drin?

Das Buch unterteilt sich in acht Kapiteln, die insgesamt 450 Seiten lang sind. Zu Beginn wird eine Einführung in JavaScript gegeben, in dem die gängigen Begriffe erläutert, die vorhandenen Laufzeitumgebungen genannt sowie ein Überblick über eine Auswahl von existierenden Entwicklungsumgebungen und Debugging-Tools geliefert werden. Das Kapitel schließt mit der ersten Einführung in die Sprache selbst ab.

Das zweite Kapitel befasst sich mit den Funktionen und den funktionalen Aspekten von JavaScript. Es wird unter anderem eine Einführung in die funktionale Programmierung gegeben und die Unterschiede zur imperativen Programmierung aufgezeigt. Das dritte Kapitel umfasst das Thema der objektorientierten Programmierung. Dort wird erläutert, wie man objektorientiert in JavaScript programmiert, also wie man Klassen programmiert und wie man mit Prototypen, Vererbungen und Datenkapselungen arbeitet. Weiterhin wird die Emulation von diversen weiteren Prinzipien der objekt-orientierten Programmierung erklärt.

Im vierten Kapitel geht es um den neuen ECMA-Script Standard in Version 6, der erst kürzlich, im Juni 2015, veröffentlicht wurde. Das Kapitel umfasst alle Neuerungen und Änderungen und zeigt, wie man sie im Alltag verwenden kann. Im darauffolgenden fünften Kapitel dreht es sich um den Entwicklungsprozess. Es werden sehr viele JavaScript-Bibliotheken und Node.js-Pakete genannt, die für den Entwicklungsprozess hilfreich sind, um etwa die Code-Qualität zu verbessern oder den Code für das Deployment zu minifizieren.

Das sechste Kapitel dreht sich rund um das Test-Drive-Development von JavaScript-Anwendungen, es begrenzt sich nicht nur auf das Testen von JavaScript-Anwendungen auf Webseiten sondern es wird auch erläutert, wie man mit diversen Test-Frameworks Node.js-Anwendungen testet. Im anschließenden vorletzten Kapitel dreht es sich um die Entwurfsmuster der „Gang of Four“. So werden viele verschiedene Entwurfsmuster aus der Softwaretechnik genannt, erläutert und gezeigt, wie man diese in JavaScript umsetzt. Zum Schluss folgen im letzten Kapitel die verschiedenen Architekturmuster von modernen JavaScript-Frameworks, wie sie in AngularJS, Backbone.js und Co. eingesetzt werden.

Wie liest es sich?

Der Autor beschreibt in dem Buch sehr viele Aspekte von JavaScript, die teilweise kompliziert und umfangreich sind. Einige Passagen muss man gegebenenfalls mehrfach lesen um es vollständig zu verstehen. Die Erklärungung sind in der Regel kurz und knackig, sodass nicht sehr viel um den heißen Brei herumgeredet wird.

Der Autor bringt immer wieder Erfahrungen und Tipps ein, die für das reale Arbeiten hilfreich sind, etwa wann man diverse Dinge tun sollte und wann man es vermeiden sollte.

Kritik

Das Buch richtet sich primär an Leute, die schon Erfahrungen mit einer Programmiersprache haben und sich JavaScript aneigenen wollen. Diesen Umstand merkt man auch im Buch, da das gängige Programmierverständnis vorausgesetzt wird. Nichtsdestotrotz ist das Buch auch für Programmierer geeignet, die bisher nur wenig Programmiererfahrung besitzen.

Buchinformationen	
Titel	Professionell entwickeln mit JavaScript [1]
Autor	Philip Ackermann
Verlag	Rheinwerk
Umfang	450 Seiten
ISBN	978-3-8362-2379-9
Preis	34,90 Euro (broschiert), 29,90 Euro (E-Book)

Der Titel des Buches impliziert, dass man durch dieses Buch mit JavaScript professionell programmieren lernen kann. Dem wird das Buch auch vollkommen gerecht, da eine große Breite an Themen, die sich rund um das Thema JavaScript drehen, behandelt werden. Positiv ist zudem hervorzuheben, dass nicht im Detail auf jede mögliche Syntax eingegangen wird, also wie man etwa Schleifen programmiert oder if-Abfragen tätigt. Dadurch ist viel Platz für die wirklich notwendigen Dinge vorhanden, die sonst zu kurz kämen. Ebenfalls positiv hervorzuheben ist, dass sich der Autor nicht auf spezielle JavaScript-Bibliotheken festlegt, von denen es ja mittlerweile sehr viele gibt. So können auch Leser dem Buch was abgewinnen, die JavaScript nicht im Browser nutzen.

Weiterhin werden häufig bei der Nennung von JavaScript-Bibliotheken Alternativen aufgezeigt und Tipps gegeben, welche Vor- und Nachteile die jeweilige Bibliothek hat, etwa ob man grunt oder gulp einsetzen sollte. Viel von den JavaScript-Funktionen selbst lernt man wiederum nicht, also z. B. wie man mit Datums-Objekten arbeitet. Dieser Fakt schadet aber dem Buch keineswegs, da man solche Dinge sowieso schneller im Internet nachschlägt. Für knapp 35€ erhält man ein gutes Buch, um einen vollständigen Einblick in das Arbeiten mit JavaScript zu gewinnen.

Redaktioneller Hinweis: Da es schade wäre, wenn das Buch bei Sujeevan Vijayakumaran nur im Regal steht, wird es verlost. Die Gewinnfrage lautet:

„*Unter welchen Namen ist die Standardisierung von JavaScript bekannt?*“

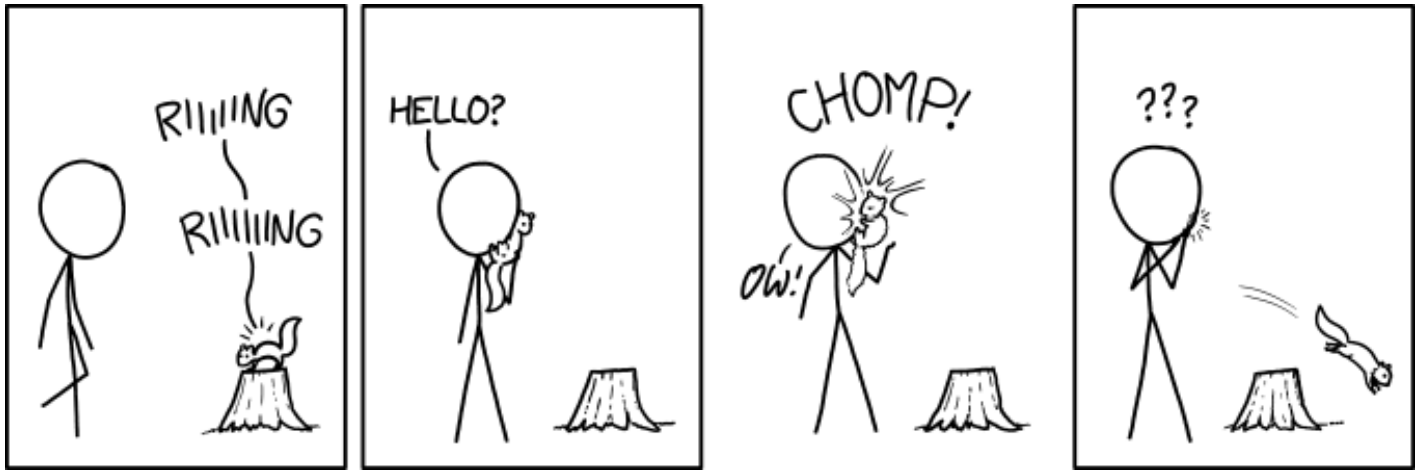
Die Antwort kann bis zum **18. Oktober 2015, 23:59 Uhr** über die Kommentarfunktion oder per E-Mail an redaktion@freiesMagazin.de geschickt werden. Die Kommentare werden bis zum Ende der Verlosung nicht freigeschaltet. Das Buch wird unter allen Einsendern, die die Frage richtig beantworten konnten, verlost.

Links

[1] https://www.rheinwerk-verlag.de/professionell-entwickeln-mit-javascript_3365/

Autoreninformation

Sujeevan Vijayakumaran (Webseite) beschäftigt sich beruflich mit JavaScript. Er setzt die Scriptsprache dabei nicht nur im Web ein.



“Squirrelphone” © by Randall Munroe (CC-BY-NC-2.5), http://xkcd.com/1578/

Rezension: Entwurfsmuster – Das umfassende Handbuch

von Dominik Wagenführ

Wer mit Software-Entwicklung zu tun hat, denkt bei dem Wort „Entwurfsmuster“ fast automatisch an die Gang of Four [1] und die Beschreibung der 23 Entwurfsmuster in ihrem Buch „Entwurfsmuster. Elemente wiederverwendbarer objektorientierter Software“. Wer nicht mit Software-Entwicklung zu tun hat: Bei Entwurfsmustern handelt es sich um eine Art Schablone, wie man wiederkehrende Probleme in der Software-Entwicklung lösen kann [2]. Es gibt Tausende von Entwurfsmustern, 1994 haben vier Entwickler, die sogenannte Gang of Four, ein Buch verfasst, welches die in ihren Augen 23 am häufigsten genutzten Entwurfsmustern enthält. Dass es noch mehr gibt als diese 23 Entwurfsmuster zeigt Matthias Geirhos in seinem Buch „Entwurfsmuster – Das umfassende Handbuch“ [3].

Redaktioneller Hinweis: *Wir danken dem Rheinwerk Verlag für die Bereitstellung eines Rezensionsexemplares.*

Was steht drin?

Das umfassende Handbuch zum Thema Entwurfsmuster beschäftigt sich entgegen des Titels nicht allein mit Entwurfsmustern im herkömmlichen Sinne. Diese nehmen mit über 300 Seiten zwar einen Großteil des Buches ein, sind aber nicht allein für den Inhalt verantwortlich.

Dabei ergänzt der Autor Matthias Geirhos die 23 Entwurfsmuster der Gang of Four zusätzlich noch um das Multiton- und das Interceptor-Muster. Danach geht er auf Architekturmuster ein, die sich aber allein auf verteilte Systeme wie SOA (Service-orientierte Architekturen) beziehen. Strukturelle Architekturmuster wie Pipes und Filter oder Blackboard werden nicht erwähnt.

Der Teil der Datenmuster bezieht sich vor allem auf Datenbanksysteme, obwohl sich die dort erwähnten Verfahren und Probleme in ähnlicher Weise auch auf Multithreading-Systeme anwenden lassen. So kann man auch dort mit optimistischen und pessimistischen Sperren arbeiten und die Probleme von Lost Updates, Dirty Reads, Non-Repeatable Reads, Phantom Reads und natürlich Deadlocks ergeben sich auf ähnliche Weise.

Bei den GUI-Mustern werden hauptsächlich Model-View-Controller (MVC), Model-View-Presenter (MVP) und Model-View-ViewModel (MVVM) erklärt, die aber alle sehr ähnlich sind.

Der letzte Teil beschäftigt sich mit Design- und Entwicklungsprinzipien, in denen Geirhos unter anderem auf SOLID [4], das Agile Manifest [5] und Design Smells und Anti-Pattern eingeht.

Wie liest es sich?

Die Entwurfsmuster sind alle klar strukturiert und in der Regel auch ausführlich gehalten. So folgen auf die Beschreibung in UML und Text die Anwendungsfälle und die Implementierung des Musters in Java – und manchmal C#. Abgeschlossen wird ein Kapitel immer mit weiteren Überlegungen und Alternativen zu den Mustern.

Vor allem die weiteren Überlegungen sind sehr hilfreich, da Geirhos oft sehr ausführlich die Vor- und Nachteile beschreibt, die mit den Mustern in der Praxis auftreten können.

Die Beispiele für die Umsetzung der Entwurfsmuster sind dabei meist der Realität und Projekten entnommen, an denen der Autor gearbeitet hat. Es ist aber wahrscheinlich Geschmackssache, ob man beispielsweise das Decorator-Muster mit Kaffee oder Pizza (wie im Buch „Head First Design Patterns“, freiesMagazin 03/2012 [6]) oder mit einem „realen“ Beispiel mit einem Vertrag, der verschlüsselt wird, erklärt bekommen möchte.

Was mitunter problematisch sein kann, ist, dass der gesamte Beispielcode und auch die Entwurfsmuster in Deutsch erklärt werden. So wird aus der „Chain of Responsibility“ eine „Zuständigkeitskette“ oder aus der „Template Method“ die „Schablonenmethode“. Diese Begrifflichkeiten sind mitunter etwas befremdlich, vor allem, wenn man sich mit anderen Entwicklern unterhält. Hier wäre Englisch als einheitliche Informatik-Sprache besser geeignet, um sprachliche Missverständnisse zu vermeiden.

Vor allem der deutschsprachige Beispielcode liest sich seltsam:

```
public BestellContext (
  String bestelldaten,
  Date dateTimeStamp,
  boolean isVerschluesselt
)
```

Dieses Schema findet man sicherlich in einigen Projekten, bei den meisten Firmen und auch in den meisten Open-Source-Projekten hat sich Englisch als beschreibende Sprache durchgesetzt, was den Austausch auf internationaler Ebene natürlich erleichtert. Problematisch ist die Verwendung von deutschen Begriffen auch für das Lektorat, denn so wird durch die Korrektur des Buches aus **`public void kuendigen(...)`** im Fließtext eine „kündigen"-Methode (S.65).

Ein weitere Eigenart des Autors, an die man sich erst gewöhnen muss, ist, dass in vielen Mustern ein Interface (z. B. „Erzeuger") eine Basisklasse zurückliefert (hier „Produkt"), die konkrete Klasse „KonkreterErzeuger" liefert aber plötzlich ein „KonkretesProdukt". Das heißt, die Signatur der überladenen Methode passt in Interface und Realisierung nicht mehr zusammen. Dies findet man in allen UML-Diagrammen, die im Buch verwendet werden (S. 51, S. 79 etc.). In dem darauf folgenden Code wird die Methode der Schnittstelle/-Basisklasse in der konkreten Klasse glücklicherweise korrekt implementiert – da sonst auch der Quellcode nicht übersetzen würde.

Ganz fehlerfrei ist das Buch leider insgesamt nicht. Manchmal sind es nur kleinere Fehler, sodass ein „I" im Interface vergessen wird (S. 105) oder eine Bildunterschrift nicht passt (S. 509). Etwas verwirrender ist, wie oben erwähnt, wenn UML und Code nicht zusammenpassen. Manchmal ist es dabei auch nur, dass die Groß-/Kleinschreibung nicht passt (S. 120/122) oder Interfaces nicht korrekt bezeichnet werden (S. 337). Man merkt hieran, dass es für die zweite Auflage noch Verbesserungspotential gibt, auch wenn natürlich im Ganzen gesehen die Fehler eher kleinerer Natur sind und auch nicht massiv auf jeder Seite auftauchen.

Ein Fauxpas ist (aus Design-Sicht) aber eigentlich nicht zu entschuldigen. So wird im Teil zu Design-Prinzipien auch das Gesetz von Demeter [7] erwähnt (S. 596). Dort wird gesagt, dass ein Objekt M auf die Methoden der Bestandteile von M, der Argumente von Methoden von M, der von M erzeugten Objekte und von globalen Variablen zugreifen darf. Gerade der letzte Punkt ist aber falsch und eine Verstoß gegen das Gesetz. Glücklicherweise wird eine Seite später bei der ausführlichen Erklärung korrekt erklärt, dass man nicht auf globale Objekte zugreifen soll.

Fazit

Trotz der kleineren Fehler, die eben aufgelistet wurden, ist das Entwurfsmuster-Buch von Matthias Geirhos ein sehr gutes und aktualisiertes Nachschlagewerk für verschiedene Entwurfsmuster, wobei der Begriff vom Autor entsprechend weit aufgefasst wird, was aber nicht schlimm ist. Sicherlich sind nicht alle Teile für jeden interessant, da nicht jeder Entwickler mit Service-orientierten Architekturen oder GUI-Entwicklung zu tun hat, aber als Anreiz zur Weiterbildung sind diese Kapitel selbst für solche Personen interessant.

Buchinformationen	
Titel	Entwurfsmuster – Das umfassende Handbuch [3]
Autor	Matthias Geirhos
Verlag	Rheinwerk Verlag, 2015
Umfang	643 Seiten
ISBN	978-3-8362-2762-9
Preis	39,90 Euro (Hardcover), 34,90 Euro (E-Book)

Der Schreibstil des Autors ist dabei auch locker und nicht zu trocken, was manch einen beim Buch der Gang of Four ggf. gestört hat. Und so gibt es am Ende noch ein schönes Zitat, was die Entwicklung von Software-Systemen sehr gut beschreibt (aus dem Abschnitt zum Fassade-Pattern, S. 171): *„Komplexität ist ein Eichhörnchen. Bis man die Komplexität wahrnimmt, hat es schon viele Haselnüsse versteckt."*

Redaktioneller Hinweis: Da es schade wäre, wenn das Buch bei Dominik Wagenführ nur im Regal steht, wird es verlost. Die Gewinnfrage lautet:

„Es gibt zum Entwurfsmuster Singleton ein recht ähnliches, etwas unbekannteres Entwurfsmuster,

was das gleiche Ziel hat, aber dies anders umsetzt. Wie heißt es?“

Die Antwort kann bis zum **18. Oktober 2015, 23:59 Uhr** über die Kommentarfunktion oder per E-Mail an redaktion@freiesMagazin.de geschickt werden. Die Kommentare werden bis zum Ende der Verlosung nicht freigeschaltet. Das Buch wird unter allen Einsendern, die die Frage richtig beantworten konnten, verlost.

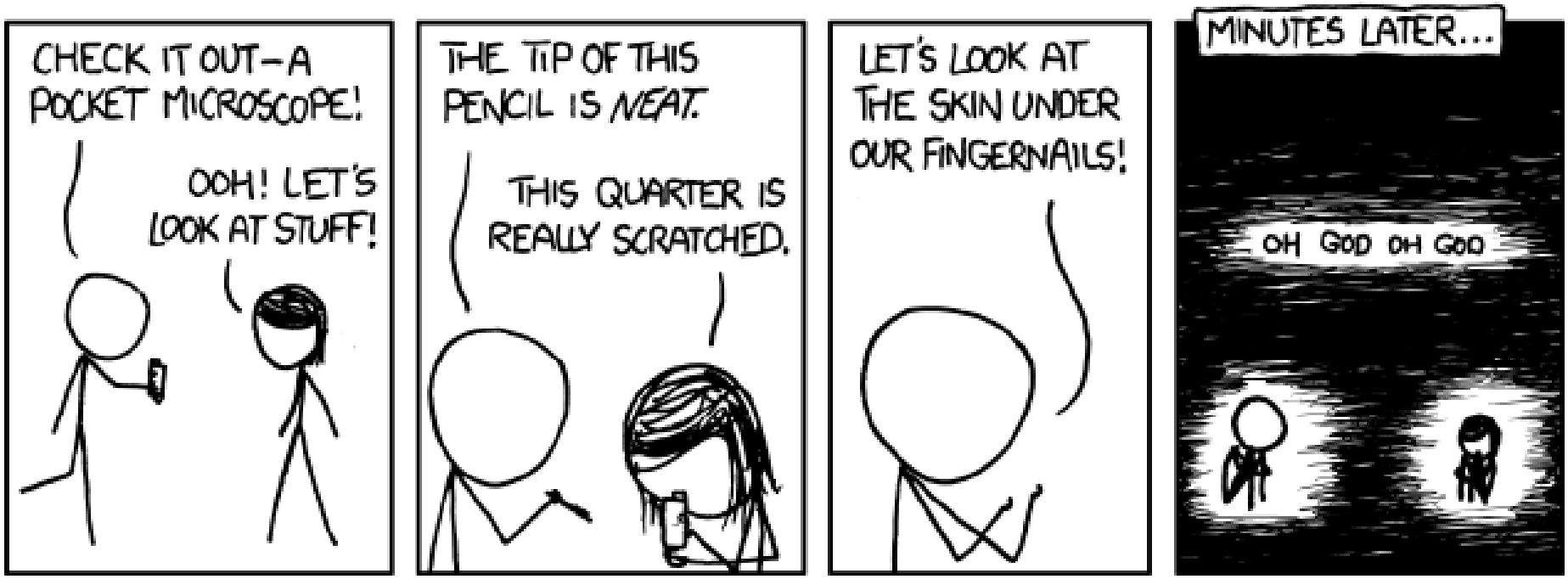
Links

[1] https://de.wikipedia.org/wiki/Viererbande_(Softwareentwicklung)
[2] https://de.wikipedia.org/wiki/Entwurfsmuster
[3] https://www.rheinwerk-verlag.de/entwurfsmuster_3538/
[4] https://de.wikipedia.org/wiki/SOLID
[5] https://de.wikipedia.org/wiki/Agiles_Manifest
[6] http://www.freiesmagazin.de/freiesMagazin-2012-03
[7] https://de.wikipedia.org/wiki/Gesetz_von_Demeter
[8] http://www.codeproject.com/Tips/692285/Monostate-pattern

Autoreninformation

Dominik Wagenführ (Webseite) ist Software-Architekt und hat dabei tagtäglich mit Entwurfsmustern zu tun.

Teilen *Kommentieren*



“Never Do This” © by Randall Munroe (CC-BY-NC-2.5), http://xkcd.com/860/

Leserbriefe

Für Leserbriefe steht unsere E-Mailadresse redaktion@freiesMagazin.de **zur Verfügung – wir freuen uns über Lob, Kritik und Anregungen zum Magazin.**

An dieser Stelle möchten wir alle Leser ausdrücklich ermuntern, uns auch zu schreiben, was nicht so gut gefällt. Wir bekommen sehr viel Lob (was uns natürlich freut), aber vor allem durch Kritik und neue Ideen können wir uns verbessern.

Leserbriefe und Anmerkungen

Fedora 22

Es ist richtig, dass jede neue Fedora-Version erst einige Updates braucht, bis man zufrieden sein kann. Distributions-Upgrades mit FEDUP sind manchmal zickig und es werden auch mal einige Pakete, die nicht als Upgrade vorliegen, entfernt. Das ist aber meist kein Problem, wenn man nicht gerade drauf angewiesen ist. Kurze Zeit später gibt es sie dann wieder im Repo.

In der Bedienung und in der Logik nicht ganz einfach ist die Partitionierung bei der Installation. Keine Neuigkeit und trotzdem jedes mal anders. Ich pfeife drauf und nehme eine Gparted-LiveCD und bereite die Partitionen vorher vor. Dann braucht man sie bei der Installation nur auszuwählen. Das klappt relativ problemlos. Ein Grund FEDUP zum Dist-Upgrade zu nutzen und nicht neu zu installieren.

Wirklich Geschmackssache sind die neuen Icons. Ich finde sie grauenhaft und musste oft rätseln was sie bedeuten sollen. Als Überzeugungstäter beiße ich mich aber durch und fahre die Strategie, immer die Vorgängerversion zu nutzen, bis der Support eingestellt wird (wenn die übernächste Version erscheint).

Bei den RedHat-Distributionen sind kurze unverschachtelte Konfigurationspfade und kurze Befehle eher anzutreffen als bei Debian/Ubuntu, bei denen man sich manchmal die Finger wund tippen muss. Das Update-Tool DNF ist schneller und YUMEX-DNF auch wesentlich schneller als das alte YUMEX. Aber das Layout...! Was ist was?

Zur Kompatibilität mit CentOS (7.1) noch die Info, dass es nur manchmal mit den Abhängigkeiten klappt und die Pakete 1:1 funktionieren. Manche Pakete wie BOINC bekommt man auf CentOS nicht einfach „rüber kopiert“. Ebenso gibt es Paket-Redundanzen bei TOR, die nicht funktionieren. Das Fedora-Paket muss mit `exclude=tor*` in den Repo-Files geblockt werden und ein Repo-File mit den Inhalten vom Tor-Projekt erzeugt werden. Das funktioniert dann. Hintergrund sind falsch gesetzte Berechtigungen beim Fedora-Paket. Dumm ist nur, dass die Update-Info im KDE-Desktop trotzdem das Fedorapaket-Update für TOR meldet, obwohl es in den Repo-Files geblockt ist. **O. Grimm** (Kommentar)

Lob

Ich wollte euch mal wieder für ein gelungenes freiesMagazin gratulieren. Ich habe die Ausgabe sehr genossen. Der Artikel über OpenBSD war zwar an einigen Stellen mehr verwirrend als hilfreich, insgesamt war dieser Blick über den Tellerrand aber sehr interessant. Der Artikel über Fedora war auch etwas zu umfangreich für meinen Geschmack. Aber der Review zum Aquarius E5 kam genau zum richtigen Zeitpunkt, steht doch die Anschaffung eines neuen Smartphones ins Haus. Besonders gefallen hat mir jedoch der Artikel zum distrochooser [1]. Dieses Tool kannte ich noch gar nicht und ich habe so direkt noch ein paar neue Linuxdistributionen entdeckt.

Volker Thiel

LINKS

[1] http://distrochooser.de/

Teilen *Kommentieren*

Die Redaktion behält sich vor, Leserbriefe gegebenenfalls zu kürzen. Redaktionelle Ergänzungen finden sich in eckigen Klammern.

Veranstaltungskalender

Messen				
Veranstaltung	**Ort**	**Datum**	**Eintritt**	**Link**
Hackover	Hannover	16.10.–18.10.2015	25 EUR	https://hackover.de/
Ubucon	Berlin	23.10.–25.10.2015	10 EUR	http://ubucon.de/
OpenRheinRuhr	Oberhausen	07.11.–08.11.2015	–	http://www.openrheinruhr.de/
BeGeistert 029	Düsseldorf	07.11.–08.11.2015	45 EUR	https://www.haiku-os.org/conference/2015_begeistert_029
Linux Presentation Day	Linux Community	14.11.2015	frei	http://www.linux-presentation-day.de/
LinuxDay	Dornbirn	21.11.2015	frei	http://linuxday.at/
Chaos Communication Congress	Hamburg	27.12.–30.12.2015	–	https://events.ccc.de/

(Alle Angaben ohne Gewähr!)

Sie kennen eine Linux-Messe, welche noch nicht auf der Liste zu finden ist? Dann schreiben Sie eine E-Mail mit den Informationen zu Datum und Ort an redaktion@freiesMagazin.de.

Vorschau

freiesMagazin erscheint am ersten Sonntag eines Monats. Die November-Ausgabe wird voraussichtlich am 1. November u. a. mit folgenden Themen veröffentlicht:

- Rezension: Sqhinx
- Rezension: Entwurfsmuster

Konventionen

An einigen Stellen benutzen wir Sonderzeichen mit einer bestimmten Bedeutung. Diese sind hier zusammengefasst:

$: Shell-Prompt

#: Prompt einer Root-Shell – Ubuntu-Nutzer können hier auch einfach in einer normalen Shell ein **sudo** vor die Befehle setzen.

↷: Kennzeichnet einen aus satztechnischen Gründen eingefügten Zeilenumbruch, der nicht eingegeben werden soll.

~: Abkürzung für das eigene Benutzerverzeichnis **/home/BENUTZERNAME**

: Kennzeichnet einen Link, der auf eine englischsprachige Seite führt.

: Öffnet eine höher aufgelöste Version der Abbildung in einem Browserfenster.

Impressum

freiesMagazin erscheint als PDF, EPUB und HTML einmal monatlich.

Kontakt

E-Mail	redaktion@freiesMagazin.de
Postanschrift	freiesMagazin c/o Dominik Wagenführ Beethovenstr. 9/1 71277 Rutesheim
Webpräsenz	http://www.freiesmagazin.de/

Autoren dieser Ausgabe

Bernd Dietzel	S. 9
Gerrit Kruse	S. 3
Mathias Menzer	S. 7
Christina Möller	S. 21
Jochen Schnelle	S. 19
Sujeevan Vijayakumaran	S. 17, S. 23
Dominik Wagenführ	S. 12, S. 25

ISSN 1867-7991

Erscheinungsdatum: 11. Oktober 2015

Redaktion

Christian Schnell, Matthias Sitte, Dominik Wagenführ (Verantwortlicher Redakteur)

Satz und Layout

Moritz Kiefer, Benedict Leskovar, Kai Welke

Korrektur

Daniel Braun, Frank Brungräber, Vicki Ebeling, Stefan Fangmeier, Mathias Menzer, Christian Schnell, Karsten Schuldt

Veranstaltungen

Ronny Fischer

Logo-Design

Arne Weinberg (CC-BY-SA 4.0 Unported)

Dieses Magazin wurde mit LaTeX erstellt. Mit vollem Namen gekennzeichnete Beiträge geben nicht notwendigerweise die Meinung der Redaktion wieder. Wenn Sie freiesMagazin ausdrucken möchten, dann denken Sie bitte an die Umwelt und drucken Sie nur im Notfall. Die Bäume werden es Ihnen danken. ;-)