# UpTimes

Mitgliederzeitschrift der
German Unix User Group


## Praxishack Gateway-Timeouts

## Weiterbildung für Fachinformatiker

## Remote-Webcam basteln

2014-1

# Inhaltsverzeichnis

## Gruß vom Vorstand
### Liebe Mitglieder, liebe Leser!

von **Wolfgang Stief, Vorstandsvorsitzender**

So viele Berichte. So viele Fragen.

Bertolt Brecht, aus dem Gedicht *Fragen eines lesenden Arbeiters*

Wenn diese UpTimes erscheint, steht der LinuxTag in Berlin unmittelbar bevor. Nach einigen Jahren in Kooperation mit der Messe Berlin gehen die Organisatoren dieses Jahr neue Wege. Wir vom Vorstand wünschen den Veranstaltern, dass ihre Pläne und Erwartungen aufgehen. Neue Wege einschlagen ist nicht immer einfach und oft genug mit Risiko verbunden…Für die Mitglieder der GUUG bleibt aber alles beim Alten: Am ersten Tag des LinuxTag wird es auch dieses Jahr wieder den GUUG-Empfang für Mitglieder und Geschäftspartner geben. Am Donnerstag, 8. Mai 2014, sind dazu alle Mitglieder ab 19:30h in die Alte Pumpe, Lützowstrasse 42 in Berlin eingeladen. Hoffentlich ist das Wetter dieses Jahr schöner als im letzten Jahr, sodass wir wieder draußen im Garten sitzen und grillen können.

Auch die GUUG schlägt für ihre traditionelle Vortragsveranstaltung neue Wege ein, das aber – so der Plan – nur temporär. Im Grußwort der letzten UpTimes hatte ich erläutert, dass wir allerlei Schwierigkeiten hatten, für das Frühjahrsfachgespräch 2014 die passenden Räume zu finden. Schließlich sind wir fündig geworden, und so wird das FFG 2014 dieses Jahr erst im Herbst, vom 23. bis 26. September, an der Ruhr-Universität Bochum stattfinden. Wir hoffen, damit allen genug Vorlauf für die Terminplanung zu geben. Der *Call for Presentations* läuft bereits und noch bis Ende Mai (siehe Seite 5). Vier Wochen später – noch vor den Sommerferien – wollen wir das Programm veröffentlichen.

## GUUG-Präsenz im Veranstaltungsjahr 2014

Das GUUG-Jahr 2014 begann mit der Teilnahme an den *Chemnitzer Linux-Tagen* Mitte März. Neben Sponsoring waren wir wieder mit einem Stand vertreten, haben vielen Menschen erklärt, was die GUUG ist und was sie so macht. Einige Mitglieder hatten sich außerdem besorgt um den Fortbestand des FFG gezeigt. Diese Sorge konnten wir mit Hinweis auf die Planung für den Herbst ausräumen. Anfang April fand dann in Leipzig das *Libre Graphics Meeting 2014* statt, das die GUUG finanziell unterstützte. Wir haben quasi dafür gesorgt, dass die gut 200 Teilnehmer die vier Konferenztage lang nicht dursten mussten. Um auf die GUUG als Sponsor aufmerksam zu machen, ließen wir ein neues Rollup erstellen, das auch in Zukunft auf unseren Ausstellungsständen zu sehen sein wird. Wer neugierig auf das Motiv ist, schaut im GUUG-Wiki vorbei. Dort sind auch ein paar ältere Plakatmotive hinterlegt [1].

Gerade noch so in der ersten Jahreshälfte wird am DESY in Hamburg das weltweit zweite *Exceptional Hardware Software Meeting* sein. Dort geht es um eine etwas andere Dimension des Hardware-Hackings: beispielsweise Elektronenstrahlschweißen, Gensequenzer, Quantenkryptographie, Radioastronomie, Massenspektroskopie. Wie schon bei der ersten Veranstaltung im Winter 2012 in Berlin leistet die GUUG auch dieses Mal wieder finanzielle Unterstützung. Wir freuen uns im Vorstand immer, nicht nur etablierte Veranstaltungen zu unterstützen, sondern auch frischen Ideen und Konzepten unter die Arme zu greifen. Wer Ideen dazu hat oder solch eine junge Veranstaltungen kennt, die zur GUUG passen und finanzielle Unterstützung brauchen können, wendet sich bitte direkt an den Vorstand (<vorstand@guug.de>).

Für das zweite Halbjahr 2014 haben wir im Moment nur Etabliertes auf dem Schirm: *FrOSCon* am 23. und 24. August in St. Augustin, unser eigenes *Frühjahrsfachgespräch* Ende September in Bochum und sehr wahrscheinlich auch wieder die *OpenRheinRuhr* (ORR), voraussichtlich am 8. und 9. No-

vember in Oberhausen. Letzteres lohnt übrigens in zweifacher Hinsicht: Das Veranstaltungsticket berechtigt zu einem Besuch im Rheinischen Industriemuseum, dessen Räume die ORR beherbergen.

## GUUG-Arbeitstreffen

In einigen Gesprächen und auf mindestens zwei Mitgliederversammlungen habe insbesondere ich wiederholt die Idee geäußert, einen Wochenendworkshop mit interessierten Mitgliedern zu veranstalten, bei dem es um die Zukunft der GUUG geht. Ich wurde bereits mehrfach darauf angesprochen, was denn nun damit sei. Das Thema ist nicht vom Tisch, und ich habe es auch nicht aus den Augen verloren. Aber ich hätte das gerne professionell durchgeführt, bevorzugt zusammen mit einem Moderator, der sich damit auskennt: Denn am Ende soll möglichst mehr dabei herauskommen als ein paar nette Diskussionen am runden Tisch darüber, was man denn vielleicht irgendwann, wenn denn mal Zeit ist, so alles machen könnte und sollte, mit oder im Verein. Für eine entsprechend sorgfältige Organisation fehlt mir im Moment leider die Zeit. Und deshalb hört ihr derzeit nichts davon.

Was sehr viel einfacher zu organisieren ist, ist die meines Wissens erste Redaktionssitzung der UpTimes. Sie ist einfacher, weil die Redaktion eine viel kleinere Gruppe ist und weil der zu besprechende Themenumfang relativ klar umrissen ist. Bisher arbeiten wir als virtuelles Team zusammen und treffen uns bestenfalls mal zu zweit zufällig am Rande einer Veranstaltung. Weil wir aber im Moment alle motiviert sind, die UpTimes weiter nach vorne zu treiben, und uns noch nie alle gemeinsam getroffen haben, wollen wir nach Erscheinen dieses Heftes mal daran gehen, einen Termin zu finden – aus dem dann ein in Zukunft noch interessanteres Heft hervorgeht. (Sagt man eigentlich Heft, wenn man elektronisch vertreibt?)

Bei der Lektüre dieser Ausgabe wünsche ich Euch allen viel Spaß und neue Impulse für die tägliche Arbeit mit Unix. Außerdem danke ich einmal mehr dem UpTimes-Team für seine Arbeit!

**Links**

[1] GUUG-Plakatmotive: https://wiki.guug.de/offen/guug-plakate/index

**Über Wolfgang**

Wolfgang Stief hat Linux während des E-Technik-Studiums kennen und schätzen gelernt (Kernel 0.99). Seit 1998 verdient er sein Geld überwiegend mit Solaris, Storage-Systemen und verteilten Dateisystemen. Wenn er gerade mal nicht beruflich oder für die GUUG unterwegs ist, züchtet er Gemüse, treibt sich in sozialen Netzwerken herum oder macht Musik.

## Call for Presentations
### GUUG-Frühjahrsfachgespräch 2014

Das Frühjahrsfachgespräch (FFG) ist eine Veranstaltung von und für IT-Profis aus dem Unix-/Linux-Umfeld. Als Hauskongress der German Unix User Group (GUUG) e.V. ist das FFG gleichzeitig die ideale Gelegenheit zu einem persönlichen Treffen der GUUG-Mitglieder. Der Call for Presentations läuft bis zum 31. Mai 2014.

von **Programmteam FFG**

Es gäbe keine Geselligkeit,
wenn die Gedanken der Menschen
auf ihrer Stirn zu lesen wären.

Marie von Ebner-Eschenbach

Der Schwerpunkt des Kongresses liegt auf technischen und praxisorientierten Fragestellungen beim Betrieb von IT-Systemen. An Bedeutung gewinnen auch rechtliche, ethische, soziale und ökonomische Aspekte der Informationstechnologie.

Die viertägige Konferenz im September 2014 an der Ruhr-Universität Bochum [1] teilt sich in zwei Tutoriums- und zwei Vortragstage. Die Tutorien können ein- oder zweitägig ausgelegt sein. Für Vorträge sind auf der Konferenz zwei parallele Tracks vorgesehen. Ein Vortragsslot hat 45 Minuten, 5 Minuten sind für Diskussionen und Raumwechsel eingeplant. Die Präsentationen werden im Nachgang über unseren Webserver zur Verfügung gestellt, sofern uns eine Erklärung des Autors vorliegt, dass seine Slides Rechte Dritter nicht verletzen.

### Gesuchte Themen

Möchten Sie als professioneller System- oder Netzwerkadministrator, Entwickler, Datenbankexperte, Systemarchitekt oder IT-Sicherheitsexperte über Ihre Erfahrungen berichten, Ihre Fragestellungen vortragen und diskutieren? Dann bieten wir Ihnen dafür einen im deutschsprachigen Raum einzigartigen Teilnehmerkreis. Für Tutorien und Vorträge suchen wir Referenten, die
Dem Fokus der Konferenz entsprechend sollten sich die eingereichten Beiträge in mindestens eine der folgenden Kategorien einordnen lassen:

- Betriebssysteme/Applikationen: Architekturen, Rechtekonzepte, neue Entwicklungen, Administration, Mobilsysteme
- Kernel-Nahes mit Praxisbezug: Neuentwicklungen im Linux-, BSD- oder anderen Open-Source-Kerneln
- Netzwerke: Protokolle, Technologien
- Virtualisierung/HA: OS, Netze, Cluster, SAN, Dateidienste, (Hoch-)Verfügbarkeit, OS/Storage-Virtualisierung und Cloud
- Dateisysteme: Verteilte Dateisysteme, Clusterdateisysteme, Dateisysteme für Spezialanwendungen
- Middleware: Datenbanken, Applikations-Server u. ä.
- Informationssicherheit: Organisatorische, betriebliche und insbesondere technische Aspekte
- Infrastruktur des Rechenzentrums: Klima, Energie und Überwachung
- Betrieb: Monitoring, Backup/Datensicherung
- Nicht-technische Themen: Arbeitsorganisation, rechtliche Fragestellungen, Lizenzen und Patente, Aus- und Weiterbildung

Falls Sie sich nicht sicher sind, ob Ihr Thema für das FFG geeignet ist, wenden Sie sich bitte per E-Mail an <ffg2014@guug.de>. Ein Blick in die Programme vergangener FFG gibt thematische Einblicke [2]. Wir freuen uns über alle Beiträge mit konkretem Praxisbezug. Produktpräsentationen sind nicht erwünscht, das FFG dient dem technischen Austausch und der Weiterbildung: Vorträge sind bei uns keine Sales- oder Marketingplattform. Herstellern geben wir im Rahmen der Sponsoringmöglichkeiten [3] gerne die Gelegenheit, sich und ihre Produkte auf der Veranstaltung zu präsentieren.

### Vortragsvorschläge einreichen

Die Einreichung von Kurzfassungen (maximal 2.000 Zeichen) erfolgt ausschließlich über unsere

Event-Webseite [4]. Das Programmkomitee unterzieht die Einreichungen einer Review und informieren die Autoren über Annahme oder Nichtannahme. Wir ermuntern alle angenommenen Sprecher, einen Beitrag zur Veröffentlichung auf der Webseite der Veranstaltung zu verfassen (acht bis 20 DIN-A4-Seiten). Dafür benötigen wir eine schriftliche Zusicherung, dass die Beiträge keine Rechte Dritter verletzen. Für Tutorien werden ausführliche Unterlagen benötigt, die ausschließlich den Teilnehmern der entsprechenden Tutorien zur Verfügung gestellt werden.

Alle Referenten sind zur Teilnahme am Frühjahrsfachgespräch und der Abendveranstaltung herzlich eingeladen. Reisekosten können leider nicht übernommen werden.

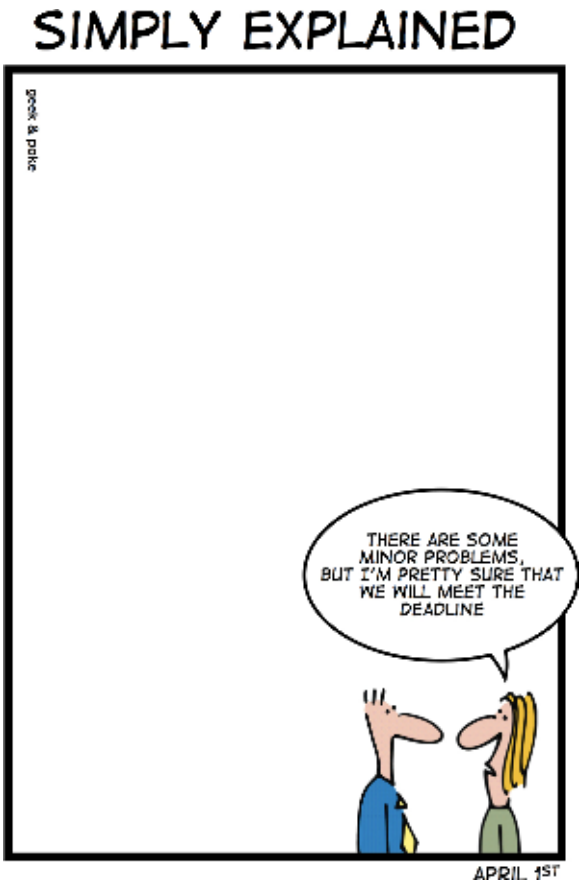
## Termine

- Einreichungsschluss: 31. Mai 2014
- Benachrichtigung an die Autoren: bis 14. Juni 2014
- Veröffentlichung des Programms: bis 28. Juni 2014
- Abgabetermin für Beiträge zur Veröffentlichung: eine Woche vor Konferenzbeginn
- Tutorien: 23./24.09.2014
- Konferenz: 25./26.09.2014

## Programmkomitee

- Michael Barabas (dpunkt.verlag)
- Andreas Bunten (Controlware GmbH)
- Detlef Drewanz (Oracle Deutschland B.V. & Co. KG)
- Lenz Grimmer (TeamDrive Systems GmbH)
- Bernd Neubacher (GUUG e.V.)
- Stefan Neufeind (LinuxTag e.V.)
- André von Raison (iX)
- Ralf Spenneberg (OpenSource Training Ralf Spenneberg)
- Dr. Christoph Wegener (wecon.it-consulting)
- Dirk Wetter (Dr. Wetter IT-Consulting) (Programmverantwortlicher)

Wir freuen uns über Firmen, die Interesse an einem Sponsoring haben oder auf der begleitenden Ausstellung auftreten. Bitte schicken Sie eine E-Mail an <ffg2014@guug.de>.




### Links

[1] Event-Location: http://www.ruhr-uni-bochum.de/universaal/raumuebersicht/vz.html
[2] Programme vergangener FFGe: https://www.guug.de/veranstaltungen
[3] Sponsoring: https://www.guug.de/veranstaltungen/ffg2014/sponsoring.html
[4] Vortrag einreichen: https://www.guug.de/veranstaltungen/ffg2014/submission.html

## Wer zu spät kommt, den bestraft das Gateway

### Geschichte einer Fehlerdiagnose

Der Alltag des Sysadmins lässt sich in drei große Bereiche einteilen: erstens – Systeme planen, bereitstellen, aufrechterhalten; zweitens – einen guten Eindruck machen; drittens – dringende Fehler beheben. Diese Geschichte ist aus dem dritten Bereich. Der Fehler bestand aus einer fehlerhaften Datenübertragung, und seine Behebung hat mich einiges an Zeit und Nerven gekostet.

Die Zeit ist selbst ein Element.

Johann Wolfgang von Goethe

Das Problem, von dem hier die Rede ist, trat alle paar Wochen mal auf. Doch bei manueller Wiederholung funktionierte die Übertragung am folgenden Tag wieder. Der Fehler zeigte sich seit einigen Monaten, doch bei uns war in der fraglichen Zeit nichts am System geändert worden. Zwischen zwei Fehlern funktionierte die Übertragung manchmal wochenlang.

Im Normalfall liefern wir die Daten, die unsere Kunden am Tag interaktiv am Server eingeben, nachts im Batchbetrieb via HTTP an die Server verschiedener Rechenzentren aus. Alles, was wir von den Kommunikationspartnern im Rechenzentrum wissen, ist Rechnername, Portnummer und eine Telefonnummer für den Fall, dass etwas nicht funktioniert. Selbst, dass die Daten via HTTP übertragen werden, bekam ich erst im Laufe der Untersuchung heraus.

Bis dahin hatte es mich auch nicht interessiert: Es ging um Daten der Kunden, und die verwendete Software kümmerte sich sowohl um die eingegebenen Daten als auch um die Verteilung der Daten an die Rechenzentren. Mein Kollege, der die Software betreute, hatte schon versucht, hinter das Problem zu kommen. Er kam aber nur so weit, dass es irgendwo im Netzwerk hakt.

## Festgefahren

Nun ja: Ein intermittierendes Problem. Auf jeden Fall funktionierte etwas, wenn auch nicht alles. Das heißt, es galt herauszufinden, was genau nicht funktioniert (siehe Kasten *Ablauf einer Fehlerdiagnose*).

Mein Kollege hatte bereits den Admin der Gegenstelle gefragt, ob ihm etwas aufgefallen wäre. Der gab zur Antwort, ihm sei aufgefallen, dass es mit allen anderen funktioniere, nur mit uns nicht. Er dachte nicht daran, etwas zu unternehmen, geschweige denn in seinen Protokollen nachzusehen: Der Fehler lag seiner Meinung nach bei uns.

Oha: Da funktioniert etwas nicht, und jemand behauptete, es liege an uns. Womit er aus seiner Sicht übrigens recht hatte, solange er sich nicht darauf einließ, genauer hinzuschauen. Dass es aus unserer Sicht genauso aussah, bloß umgekehrt, war ja nicht sein Problem. Und da es sich um ein mutmaßliches Netzwerkproblem handelte, lag der Ball jetzt in meinem Feld. Wir waren an dem Punkt, wo ich das Problem nicht mehr auf die lange Bank schieben konnte – mein Ehrgeiz war geweckt. Ich musste Daten sammeln.

Ich weiß nicht, wie es Euch geht, aber nach meinem Ideal besteht die hohe Kunst der Fehlersuche darin, ein tiefes Verständnis für das System zu entwickeln, genau hinzuschauen, scharf nachzudenken und dann den Fehler abzustellen. Oder so ähnlich. Aber Daten sammeln? Da hätte ich ja Buchhalter werden können. Noch dazu, wenn die Daten alle gleich aussehen. Oder fast gleich. Ein bisschen anders sind sie ja immer. Was also ist ihr wesentlicher Unterschied?

Es ging darum, mehr Informationen zu bekommen. Da es für uns nur eine problematische Gegenstelle gab, konnte ich den Filter für *tcpdump* eng einstellen, das heißt auf genau diesen Server, und mit wenigen Mitschnitten erste Erkenntnisse gewinnen. Es kam mir zupass, dass die Daten als Datei via HTTP-POST-Request versendet wurden. Erreichte die Sendung ihr Ziel, so das Ergebnis, kam der Statuscode `200 OK`, bei Fehlern hingegen der Code `502 Bad Gateway`.

Mit diesem Wissen ausgerüstet, sprach ich das erste Mal selbst mit dem Admin der Gegenstelle.

### Exkurs: Ablauf einer Fehlerdiagnose

Um ein technisches Problem zu diagnostizieren, gehe ich nach einem bestimmten Schema vor. Es umfasst die folgenden Fragen:

1. Funktioniert überhaupt irgendwas?
2. Funktioniert alles, was dazu gehört?
3. Ist es schnell genug?

Ich stelle die Fragen in genau dieser Reihenfolge und gehe erst dann zur nächsten über, wenn ich die vorherige mit „Ja“beantwortet habe. Denn bei „Nein“hat es keinen Zweck, die nächste Frage zu stellen, und ich kümmere mich zunächst um das zugehörige Problem. Erhalten alle Fragen ein Ja zur Antwort, dann handelt es sich um ein Problem, bei dem ich mit technischen Mitteln nicht weiterhelfen kann.

Mit der ersten Frage evaluiere ich den Umfang des Problems und schließe zugleich simple Fehler wie herausgezogene Stecker aus. Wenn überhaupt noch etwas funktioniert, kann ich mir das Problem oft via Fernwartung aus der Nähe ansehen und muss nicht vor Ort anwesend sein.

Um die zweite Frage zu beantworten, muss ich das betroffene System schon recht genau kennen. Während ich die erste Frage bei der Problemaufnahme im Gespräch mit dem Kunden beantworten kann, bekomme ich Informationen für die zweite Frage oft nur aus der Systemdokumentation sowie aus meinem Verständnis dafür, wie das betroffene System funktioniert. Bei der Problemlösung nutze ich die verbliebenen Funktionen des Systems zur Diagnose, da ich diese Frage ja erst stelle, wenn überhaupt etwas funktioniert.

Die dritte Frage ist noch schwerer zu beantworten. Zum einen ist die Antwort subjektiv, zumindest, wenn es keine detaillierten SLAs gibt. Zum anderen ist es nicht einfach, herauszufinden, was genau das Gesamtsystems verlangsamt – vor allem, wenn die beteiligte Software selbst keine Hilfe bei der Diagnose derartiger Probleme bietet.

Werde ich bei der Problemstellung mit einer der späteren Fragen konfrontiert, prüfe ich auf jeden Fall dennoch die vorhergehenden Fragen. Das heißt, bei einem Geschwindigkeitsproblem prüfe ich als erstes, ob alles funktioniert und was genau dazu gehört – DNS-Probleme zum Beispiel können den Internetzugriff sehr gut entschleunigen, auch wenn er grundsätzlich funktioniert. Oder: Meldet jemand ein Teilproblem („Unsere Website funktioniert nicht“), dann finde ich zunächst heraus, ob überhaupt etwas funktioniert („Sind andere Websites zu erreichen?“).

Außer diesen drei gibt es noch eine vierte Frage, die ich mir in jeder Phase stelle: die Frage nach der Reproduzierbarkeit beziehungsweise nach intermittierenden Fehlern. Habe ich es mit solchen zu tun, wie in der Geschichte dieses Artikels, kann ich nicht mit Sicherheit wissen, ob ich das Problem gelöst habe. Denn vielleicht habe ich es gerade nur mit einer beschwerdefreien Zeit zu tun, und nächste Woche kommt das Problem zurück. Wenn ich ein intermittierendes Problem habe, dann sammle ich daher Daten und versuche über Korrelationen der Ursache auf die Spur zu kommen.

Ich bat ihn, in seinen Logs nachzusehen, ob sich ein Hinweis ergab, warum unsere Seite manchmal die Antwort `502` und manchmal die Antwort `200` erhielt. Er gab an, dass sein System keine 502-der Antwort sende und diese wohl von unserem Gateway komme. Außerdem – das Argument kannten wir schon – funktioniere es mit allen anderen, nur mit uns nicht, daher werde er nicht nachsehen. Immerhin gab er zu, dass die IP-Adresse des Rechners, der laut TCP-Dump `502 Bad Gateway` gesendet hatte, in seinem Netzbereich lag.

Ich war ob so viel Unverfrorenheit und mangelnder Kooperationsbereitschaft auf Hundertachtzig und drauf und dran, ihm ein paar gepflegte Bösartigkeiten an den Kopf zu werfen. Zum Glück bekam ich gerade noch die Kurve, beendete das Telefonat mit irgendeiner Abschiedsfloskel und legte auf, bevor ich explodierte. Mit meinem Kollegen vereinbarte ich, dass von da an er sämtliche Gespräche mit jenem Admin führen solle.

Schöne Bescherung. Wir haben ein intermittierendes Problem, den Server der Gegenstelle als Blackbox, die das Problem zwar anzeigt, aber nicht näher spezifiziert, und dazu einen Partner, der nicht kooperiert. Was ich brauchte, war ein Ansatz, um dem Problem auf den Grund zu gehen. Und mehr Daten, viel mehr Daten, damit ich Regelmäßigkeiten und Unterschiede finden konnte. Denn wenn man keine Idee hat, hilft es manchmal, nach Mustern und Abweichungen in den Daten zu suchen.

## Vor, zurück, vor, zurück

Zumindest dabei konnte mir mein Kollege weiterhelfen. Und zwar sagte er mir, dass die Datenübertragung idempotent sei. Das heißt, ich konnte dieselben Daten mehrfach übertragen, Duplikate würde der Server verwerfen. So war ich nicht darauf angewiesen zu warten, bis der Server regulär

seine Daten verschickt. Mit *Wireshark* extrahierte ich also die übertragenen Dateien inklusive HTTP-Präambel aus den Mitschnitten. Mit *netcat* sandte ich sie an den Server. Er quittierte sie alle mit `200 OK` – auch die vormals mit `502 Bad Gateway` abgewiesenen. Wieder und wieder und wieder.

Jetzt konnte ich zwar Daten haben, so viel ich wollte, aber den Fehler nicht hervorrufen.

Also zurück auf Anfang – noch mal die Originalmitschnitte untersuchen. Worin unterschieden sich diese von meinen Versuchen mit *netcat*? Die Dumps, die beide Male funktioniert hatten, unterschieden sich lediglich im Datum und ähnlichen Kleinigkeiten. Bei den Dumps, die nicht immer funktionierten, sah ich nur eine unterschiedliche Antwort vom Server.

Schließlich kam mir eine Idee. Ich hatte bisher fast ausschließlich auf die Daten geschaut. Nun sah ich mir das Zeitverhalten an. Ein HTTP-POST-Request läuft ja wie folgt ab:

1. Der Client etabliert eine TCP-Verbindung.
2. Der Client sendet `POST` und die Daten.
3. Der Server antwortet mit `200` beziehungsweise `50`*x* und seinen Daten.
4. Client und Server schließen die TCP-Verbindung.

Wollen wir mal sehen, dachte ich, wie lange es vom Beginn des ersten Punktes bis zu Ende des zweiten Punktes dauert.

Dazu lud ich einen Mitschnitt in *Wireshark*, markierte ein Datenpaket der Verbindung und rief die Funktion *Follow TCP Stream* auf. Dann suchte ich das letzte TCP-ACK-Paket des Servers, bevor er seine HTTP-Antwort sendete. Von dessen Ankunftszeit subtrahierte ich die Zeit des ersten SYN-Paketes und erhielt die Zeitdauer, die unser Rechner benötigte, den POST-Request zur Gegenstelle zu senden.

Siehe da – ausnahmslos alle Anfragen, die mit `200` beantwortet wurden, kamen in weniger als drei Sekunden an. Bis auf eine Anfrage benötigten alle, die mit `502` beantwortet wurden, mehr als drei Sekunden. Anscheinend gab es da ein zu ungeduldiges Gateway beim Server der Gegenstelle. Immerhin ging die Verbindung über das Internet – da kann niemand für weniger als drei Sekunden Übertragungszeit zum Server der Gegenstelle garantieren.

Ich war mir sicher, dass dies das Problem war. Aber ich hatte das Telefongespräch mit jenem Admin in Erinnerung. Die Mitschnitte, die mich zur Lösung geführt hatten, reichten möglicherweise nicht, um ihn zur Kooperation zu bewegen. Mein Kollege konnte die technischen Details des Problems nicht darlegen, und ich selbst befürchtete, ausfällig zu werden, wenn besagter Admin sich von den bisher gesammelten Daten nicht überzeugen ließ. Hier musste etwas Wasserdichtes her. Etwas, bei dem ich beliebige Daten schicken und stets vorhersagen konnte, ob der Server sie annimmt oder nicht.

## Hilfe von der Hilfsaufgabe

George Pólya schreibt in seiner *Schule des Denkens* [1]:

> Eine Hilfsaufgabe ist eine Aufgabe, die wir nicht um ihrer selbst willen betrachten, sondern weil wir hoffen, dass ihre Betrachtung uns helfen kann, eine andere Aufgabe, unsere ursprüngliche Aufgabe, zu lösen.

Gut, ich gebe zu: Ich wollte einfach mal wieder programmieren. Wenn es auch mit Perl, CPAN und dem Perl-Kochbuch [2] eher dem Zusammenbauen eines Lego-Spielzeugs gleichkommt, bei dem CPAN die Teile liefert, und das Perl-Kochbuch die Aufbauanleitung. Ich wollte ein Programm, bei dessen Aufruf sich angeben ließ, wie lange die Übertragung einer Datei dauern darf.

Als erstes war mir die Idee gekommen, eines dieser Programme zu nehmen, die eine reduzierte Bandbreite simulieren, um dann das bewährte *netcat* für die Übertragung zu nehmen. Aber dann hätte ich für jede Übertragungszeit anhand der Dateigröße die Bandbreite ausrechnen und einstellen müssen. Zuviel Aufwand und unhandlich.

Die nächste Idee war, nach dem Aufbau der Verbindung oder der HTTP-Präambel die betreffende Zeit zu warten, und dann den Rest zu senden. Dabei sah ich jedoch das Problem voraus, dass die Gegenstelle die Verbindung trennt, bevor ich alle Daten senden konnte. Lieber wollte ich also die Verzögerung über die ganze Übertragung verteilen, sodass die Gegenstelle immer etwas zu tun hat. Da die Datei Text mit verhältnismäßig vielen kurzen Zeilen enthielt, wollte ich vor jeder gesendeten Zeile ein wenig warten. Damit war klar, was es werden sollte.

Schlagen wir also das Perl-Kochbuch auf. Rezept 17.1 – *Einen TCP-Client entwickeln* – war, was ich brauchte. Mit dem Modul *IO::Socket*, welches seit Perl 5.004 mit der Standard-Distribution daher kommt, lässt sich eine TCP-Verbindung zum Server aufbauen und das erhaltene Perl-Objekt wie eine normale Datei verwenden. Das hieß, mit `print`

`$socket $_;` würde ich Daten zum Server schicken und mit `$line = <$socket>;` die Antwort lesen können. Prima, damit hatte ich die halbe Miete drin.

Nun die Verzögerung: Zwischen jeder gesendeten Zeile wollte ich einen Bruchteil der Gesamtverzögerung warten. Um die Größe dieses Bruchteils zu bestimmen, las ich die gesamte Datei in ein Array ein und dividierte die Gesamtverzögerung durch die Anzahl der Zeilen. Mit `sleep $del;` konnte ich dann zwischen zwei `print`-Aufrufen warten.

Halt, wird hier der aufmerksame Perl-Kenner rufen: Die Funktion *sleep* arbeitet nur mit ganzen Sekunden, nicht mit Bruchteilen davon, die ich mit `$del` übergeben wollte. Korrekt. Aber erwähnte ich schon das Perl-Kochbuch? Rezept 3.10 *Kurze Pausen* bietet für Fälle wie diesen das Modul *Time::HiRes*. Es beschert eine *sleep*-Funktion, die mit Intervallen kleiner als eine ganze Sekunde arbeitet und auch noch Fließkommazahlen akzeptiert.

Die Kurzversion des fertigen Skripts zeigt Kasten *Listing: `http-injektor.pl`*. Wer will, findet die ausführlichere Version mit Glöckchen und Pfeifen unter [3].

**Listing: `http-injektor.pl`**

```
01 #!/usr/bin/perl
02 #
03 # Usage: http-injector --delay=DELAY SERVER [PORT] < FILE
04 #
05 use Getopt::Long;
06 use IO::Socket;
07 use Time::HiRes qw(sleep);
08
09 my %opt = ( delay => 0 );
10 GetOptions( \%opt, 'delay=i');
11 my $server = shift;
12 my $port   = shift || 80;
13
14 my $socket = IO::Socket::INET->new(PeerAddr => $server,
15                                    PeerPort => $port,
16                                    Proto    => 'tcp',
17                                    Type     => SOCK_STREAM);
18
19 my @in = <>;
20 my $del = $opt{delay} / ( 1.0 + scalar @in );
21 foreach (@in) {
22     sleep $del;
23     s/[\r\n]+$//;
24     print $socket $_, "\r\n";
25 }
26 sleep $del;
27 print $socket "\r\n";
28
29 while (my $line = <$socket>) {
30     print $line;
31 }
```

Dieses Skript öffnet mit `IO::Socket::INET->new()` eine TCP-Verbindung zum Server (Zeilen 14 bis 17). Dann liest es die komplette HTTP-Anfrage auf einmal von der Standardeingabe in das Array *@in* (Zeile 19). Es berechnet anhand der Zeilenzahl aus `scalar @in` die Länge der Pausen zwischen den Zeilen für die Gesamtverzögerung (Zeile 20). Dabei erzwingt der Ausdruck `1.0 + scalar @in` eine Fließkommadivision um die Verzögerung präzise genug zu berechnen. In der *foreach*-Schleife ab Zeile 21 schickt es die einzelnen Zeilen der HTTP-Anfrage jeweils mit einer kleinen Pause (Zeile 22) und standardisierten Zeilenenden an den Server (Zeilen 23 und 24). Am Ende schließt es die HTTP-Anfrage nach einer weiteren Pause ab (Zeilen 26 und 27) und kopiert die Antwort des Servers auf die Standardausgabe (Zeilen 29 bis 31).

## Finale – Pauken und Trompeten

Mit Hilfe des Skripts machte ich ein paar Tests, die bestätigten, was ich vermutet hatte. Mit dieser Gewissheit in der Hinterhand fühlte ich mich entspannt genug, mit dem besagten Administrator zu reden. Nachdem wir in aller Ruhe geklärt hatten, dass die Antwort wirklich von seinem System kam, dass diese jedes Mal `502` lautete, wenn das Senden mehr als drei Sekunden dauerte, und jedes Mal `200` hieß, wenn ich die Datei in weniger als drei Sekunden schickte, suchte er schließlich das Gateway und korrigierte den Timeout.

Mein Kollege kam seitdem nicht mehr mit diesem Problem zu mir. Auf gelegentliches Nachfragen sagte er, dass es nicht mehr aufgetreten sei.

Über den letzten Jahreswechsel kam ich mit einem meiner Brüder ins Gespräch, der viel auf großen Baustellen zu tun hat. Er erzählte, dass es unter den verschiedenen Gewerken Usus sei, bei Baufehlern den schwarzen Peter zunächst wortreich dem nächstbesten Gewerk zuzuschieben: „Das müssen die in Ordnung bringen, nicht wir."

Es gäbe nur zwei Wege, sagte er, jemand dazu zu bringen, seinen Fehler selbst anzupacken: Entweder man sitzt am längeren Hebel, oder man kann mit genügend Sachverstand nachweisen, dass derjenige wirklich dafür verantwortlich ist.

Aha, dachte ich, so etwas passiert also nicht nur mir.

### Literatur

[1] George Pólya: Schule des Denkens; Vom Lösen mathematischer Probleme. Tübingen, A. Francke 1994.
[2] Tom Christiansen, Nathan Torkington: Perl Kochbuch. Köln, O'Reilly 1999.
[3] Langfassung des im Artikel zur Fehlerbehebung verwendeten Perl-Skriptes `http-injektor.pl`:
http://weidner.in-bad-schmiedeberg.de/archives/2012/06/wie-ich-ein-script-schreibe/http-injektor.pl.txt
[4] Mathias Weidner: Fehlersuche bei Linux-Servern und Netzwerken. Leanpub 2014 (geplant):
https://leanpub.com/troubleshoot-linux-server-and-networks

### Über Mathias

Mathias Weidner studierte Ende der 80er Jahre des vorigen Jahrhunderts Automatisierungstechnik in Leipzig. Nach verschiedenen Stellen in der Software-Entwicklung landete er bei einem kleinen Rechenzentrum in Wittenberg als Administrator für Unix/Linux und Netzwerke. Seit einiger Zeit schreibt er hin und wieder Bücher zum Thema, die gedruckt, als PDF oder ePUB erhältlich sind. Der vorstehende Artikel ist ein überarbeiteter Ausschnitt aus dem Buch *Fehlersuche bei Linux-Servern und Netzwerken* entnommen, dass 2014 bei Leanpub erscheinen soll [4].

## Homemade Hashes
### Passwort-Hash ohne Passwort erzeugen

Inspirierendes Problem aus der Praxis: Um Zugang zur MySQL-Datenbank zu erhalten, soll eine Anwenderin, die das Passwort vergessen hat, dem Datenbankadmin den Hash eines neuen Passworts durchgeben. Um den spezifischen MySQL-Hash zu erzeugen, braucht sie jedoch Zugang zur MySQL-Datenbank, für die sie ja das Passwort nicht hat. Was soll sie machen – und zwar mit möglichst wenig Tippaufwand?

Das Fleisch in Stücke schneiden,
das eingeweichte Weißbrot fest ausdrücken
und die Zwiebel schälen.
Diese Zutaten zweimal durch den Fleischwolf drehen.
Mit Salz, Pfeffer oder Paprika, Kümmel oder Muskat kräftig würzen.

Aus dem Rezept *Hackbraten* in „Wir kochen gut: Rezepte mit Tradition“

Deadlock beim Datenbankadmin: Eine Anwenderin möchte am Telefon Zugriff auf die MySQL-Datenbank erhalten, denn sie hat ihr Passwort vergessen. Der Admin sitzt am anderen Ende der Republik. Klartextpasswörter zu versenden ist ja nun verpönt, und in Zeiten des Special Collective Service (SCS) des NSA-Programms *Stateroom* [1] möchte man die Angaben auch nicht mehr gern per Telefon oder SMS übermitteln.

Auf dieses Problem hat die Kryptographie glücklicherweise eine Antwort: Übertrüge die Anwenderin einen Hash statt eines neuen Klartextpassworts, würde der Lauscher außen vor bleiben und der Datenbankadmin einfach den Hash hinterlegen. Die Grundlagen zu diesem Ansatz implementiert auch MySQL. Der Kasten „MySQL-User und ihre Hashes“ erläutert die praktische Umsetzung dieses an sich einfachen Verfahrens samt ihren kleinen Fußangeln, die die MySQL-Syntax stellt.

Doch für die Anwenderin liegt nun ein Henne-Ei-Problem vor: Der Datenbankadmin fordert einen Passwort-Hash, um ihn in der Datenbank zu hinterlegen. Somit könnte sie sich mit ihrem neuen, selbstgewählten Passwort einloggen. Sie kann jedoch keinen Hash liefern, weil sie noch keinen Zugang zu der betreffenden MySQL-Datenbank hat, mit deren Kommandozeile der Hash zu generieren wäre. Mit Zugang zu einer MySQL-Datenbank könnte sie das nämlich folgendermaßen tun:

```
user@db> SELECT password('geheim');
+-------------------------------------+
| password('geheim')                  |
+-------------------------------------+
| *462366...A48E87D8EF59EB67D2CA26F |
+-------------------------------------+
1 row in set (0.00 sec)
```

Das kommt aber nicht in Frage, denn der Zugang erfordert ja eben das Passwort – und vor allem hat die Anwenderin keine lokale MySQL-Installation zur Hand, die sie stattdessen zur Generierung eines MySQL-Hashes nutzen könnte.

Die große Frage ist: Wie lässt sich ohne MySQL ein MySQL-Hash erzeugen?

### Die Antwort liegt im Quelltext

Etwas Recherche im Netz, ein Blick in den MySQL-Quelltext und eine praktische Verifikation zeigen, dass die MySQL-Entwickler seit der Version 4.1 den Hash auf Grundlage des *SHA-1*-Algorithmus konstruieren. Vorher setzte das Verfahren auf *MD5* auf. Beide Algorithmen gelten heute nicht mehr als neuester Stand der Kryptographie – aber das ist ein anderes Thema.

Der Algorithmus SHA-1 verarbeitet einen beliebig langen Byte-Strom als Eingabe und erzeugt daraus eine 160 Bit lange Prüfsumme. Um die vernünftig anzuzeigen, stellen Programme sie meist als eine Folge von 40 Hexadezimalziffern dar, von denen jeweils eine Ziffer vier Bit repräsentiert.

MySQL verwendet nun aber einen eigenen Weg, um seinem Hash zu kommen. Statt einer direkten Berechnung – so offenbart der Quellcode – wendet MySQL SHA-1 auf das Klartextpasswort an. Auf das resultierende 20-Byte-Binäroutput wendet MySQL nun ein zweites Mal SHA-1 an. Anschließend wandelt die Datenbank-Software die Hexadezimal-Buchstaben in Großbuchstaben um und fügt vorn noch ein Sternchen hinzu.

Als interessante Nebenerkenntnis ergibt sich, dass die MySQL-Hashes nicht gesalzen sind [2], ein Passwort also immer zum gleichen Hash führt (siehe Kasten „Kryptographische Grenzen von Hashes“).

### Exkurs: MySQL-User und ihre Hashes

Die Datenbank MySQL implementiert ihre eigene Authentifizierung in Form der System-Tabelle *user* im Datenbankschema *mysql*. Wer als eingeloggter User mit Adminrechten

```
select User, Host, Password from mysql.user;
```

eingibt, erhält die Liste der Anwender und ihrer Credentials.

Für Unix-Admins ist womöglich etwas gewöhnungsbedürftig, dass in MySQL ein User im Sinne der Datenbank immer aus Benutzer, Host und einem Klammeraffen dazwischen besteht. So unterscheidet sich etwa `root@localhost` von `root@127.0.0.1`. Im folgendem Beispiel, das die Liste der registrierten User einer MySQL-Datenbank zeigt, haben die ersten beiden Anwender das gleiche Passwort, da die Hashes übereinstimmen. Die Angabe `LIMIT 4` beschränkt hier die Ausgabe auf die ersten vier User. Den Passwort-Hash berechnet MySQL nach der im Artikel beschriebenen Methode.

```
dbadmin@db> SELECT User, Host, Password FROM mysql.user LIMIT 4;
+-----------+-----------+-------------------------------------------+
| User      | Host      | Password                                  |
+-----------+-----------+-------------------------------------------+
| root      | localhost | *EEB94B479A0D15748BA794A34EDEE39AE59FD5B9 |
| root      | 127.0.0.1 | *EEB94B479A0D15748BA794A34EDEE39AE59FD5B9 |
| myappw    | %         | *6BFE538B79ED3A909AF1ADD052FD209E64A37FD9 |
| myappr    | %         | *47EFE65FEE05868AE9D5F8463DA09AEB2DD3B610 |
+-----------+-----------+-------------------------------------------+
4 rows in set (0.00 sec)
```

Um einen Anwender anzulegen, reicht der MySQL-Befehl `CREATE USER 'john'@'localhost' IDENTIFIED BY 'geheim';`. Die Anführungszeichen sind gewöhnungsbedürftig – doch noch verwirrender ist, dass es an dieser Stelle ausreicht, ein Klartextpasswort einzugeben, das dann in der History landet (`geheim`). Hat der Datenbankadmin jedoch einen fertigen Hash bekommen, ist ein anderer Weg notwendig, nämlich `CREATE USER 'john'@'localhost' IDENTIFIED BY PASSWORD '*4290E4...9AB09D';`. Wer sich das nicht merken kann, erhält mit `help create user;` eine Gedächtnisstütze, falls eine Online-Verbindung besteht.

Soweit die Theorie. Wie lässt sich diese nun mit üblichen Programmiersprachen in die Praxis umsetzen?

## Implementation in Perl und PHP

Der erste Weg führte den Autoren zu Perl, das leider nicht direkt – quasi in der Hausapotheke – eine Lösung an Bord hat. Da jedoch das CPAN-Repository praktisch alles enthält, was denkbar ist, gilt es nur, das richtige Modul zu identifizieren: In *Digest::SHA* finden sich die Funktionen *sha1()* für den ersten und `sha1_hex()` für den zweiten Schritt. Die für MySQL notwendige Umwandlung in Großbuchstaben erledigt *uc()*, die Strings fügt der Concatenation-Operator . zusammen. Als Ergebnis lässt sich folgender Perl-Einzeiler direkt in die Kommandozeile tippen:

```
perl -MDigest::SHA=sha1_hex,sha1 \
  -le 'print "*" . uc(sha1_hex(sha1("geheim")));'
```

Mit PHP lässt sich der gleiche Ansatz ähnlich umsetzen: Hier ist die Funktion *sha1()* bereits eingebaut. Wer den zweiten, optionalen Parameter auf *true* setzt, erhält Binäroutput. Ohne dieses Argument kommt der bekannte Hex-String zurück. Die Umwandlung in Großbuchstaben erledigt hier die Funktion *strtoupper()*. Wer den PHP-Code anschließend mit `php -a` in die interaktive PHP-Shell füttert, spart sich die Einbettung in ein HTML-Dokument:

```
echo 'print "*" . \
  strtoupper(sha1(sha1("geheim",true)));' | php -a
```

Bleibt noch die Frage, was die Anwenderin tun könnte, wenn sie gar keine externe Programmiersprache zur Hand hat.

### Kryptographische Grenzen von Hashes

Hashes statt Klartext zu übertragen ist ein eleganter Kniff, der folgende Grenze besitzt: Zwar gibt es theoretisch $2^{160}$ verschiedene Klartexte, die alle auf einen eigenen Hash abbilden, aber meist verwenden Anwender nur eine kleine Teilmenge davon. Wer also Wörterbücher mit beliebten Passwörtern hasht, der kann sich eine durchsuchbare Datenbank mit Hashes und den Klartexten dazu anlegen (eine so genannte *Rainbow Table*). Solche Listen gibt es im Netz und man kann davon ausgehen, dass die einschlägigen Drei-Buchstaben-Organisationen so etwas zur Hand haben.

Für eine vollständige Liste nach Brute-Force-Methoden reicht die heutige Technik zwar noch nicht, da immer noch weit über $2^{100}$ TByte Daten nötig wären. Andererseits haben Forscher in den letzten Jahren Fortschritte bei kryptologischen Angriffen erzielt: Mehrere Kryptoexperten raten davon ab, in neueren Systemen SHA-1 zu implementieren, und verweisen stattdessen auf die Nachfolgefamilie SHA-2 [3].

Solange Anwender und Admins sich nur gegen neugierige Kollegen schützen wollen, scheint im Alltag das Vorgehen akzeptabel zu sein, solche Hashes zumindest für Initialpasswörter zu verwenden und den Anwender dann zu bitten, es unmittelbar zu ändern. Das bedeutet natürlich, dass der Übertragungsweg gesichert sein muss und dass keine Unbefugten Zugang zu der Tabelle *mysql.users* erhalten dürfen.

## Implementation mit *GNU Coreutils*

Die praktisch überall installierten *GNU Coreutils* bringen das Kommando *sha1sum* mit [4]. Leider ist das Ausgabeformat schlecht zur Weiterverarbeitung geeignet, außerdem beherrscht es keinen Binär-Modus für die Ausgabe. Trotzdem gelingt die Berechnung mit diesem Aufruf:

```
01 echo -n \*; \
02 echo -ne $(echo -n geheim \
03   | sha1sum \
04   | awk '{print $1}' \
05   | sed 's/../\\x&/g' \
06 ) \
07 | sha1sum \
08 | sed 's/\s.*$//' \
09 tr abcdef ABCDEF
```

In Zeile 01 gibt die Befehlsfolge einen Stern aus, bleibt aber mit der Option `-n` in der gleichen Zeile der Ausgabe.

Das zweite Kommando in Zeile 02 ruft mit `$()` eine Subshell auf. Als Eingabe dient in dieser das Klartextpasswort `geheim` ohne Zeilenumbruch (Zeile 02). Danach berechnet in Zeile 03 `sha1sum` den Hash aus der Standardeingabe, den das Kommando durch ein Minuszeichen in der Ausgabe nach dem eigentlichen Hash anzeigt. Das *Awk*-Kommando in Zeile 04 entfernt den Hinweis auf die Eingabedatei, da er nicht von Interesse ist.

Der folgende Schritt ist ein Trick: Der *Sed*-Aufruf `sed 's/../\\x&/g'` in Zeile 05 hängt mit `..` jeweils vor zwei Hexadezimalzeichen den String `\x`. Der Backslash in diesem String muss geschützt werden – durch einen weiteren Backslash. So entsteht `\\`. Der Modifier `g` am Ende des Sed-Komandos wendet die Ersetzung mehrfach an, in diesem Fall insgesamt 20 mal. Somit implementiert die Subshell – zusammen mit der Option `-e` zum Interpretieren des Backslash-Escapes – beim ersten *echo* aus Zeile 02 den binären SHA-1-Aufruf.

Der Binärstrom wandert ab Zeile 07 ein zweites Mal durch `sha1sum`. Diesmal entfernt der Sed-Aufruf aus Zeile 08 den Hinweis auf die Standardausgabe. In Zeile 09 erzeugt `tr` als letzte Stufe der Pipe Großbuchstaben für das MySQL-Format.

## Einfacher mit OpenSSL

Es ist natürlich trickreich und tippaufwändig, die Bash selbst Binärcode mittels `echo -e` und der `\xXX`-Schreibweise in einer Subshell erzeugen zu lassen. Kürzer geht's, wenn das OpenSSL-Paket installiert ist. Dieses kann die erste Runde des SHA-1-Algorithmus' gleich binär ausgeben:

```
01 echo -n \*; \
02 echo -n "geheim" \
03 | openssl sha1 -binary \
04 | sha1sum \
05 | sed 's/\s.*$//' \
06 | tr abcdef ABCDEF
```

Der zweite Aufruf von *echo* in Zeile 02 schickt das Klartextkennwort diesmal ohne Zeilenendezeichen an die Standardeingabe der *sha1*-Funktion von OpenSSL (Zeile 03). Dieses gibt die Hash gleich im Binärformat aus, welches in den Zeilen 04 bis 06 genauso weiter verarbeitet wird, wie in den Zeilen 07 bis 09 des vorigen Beispiels.

## Aufruf: Wer kann's kürzer?

Mit 66 Zeichen ist die PHP-Variante zur Berechnung eines MySQL-Hashes am kürzesten, dicht gefolgt von Perl mit rund 80 Zeichen. Um den Artikel nicht um mehrere Seiten zu verlängern, wurde auf eine Java-Variante verzichtet.

**Aufruf:** Wer kann Implementationen in anderen Programmiersprachen beisteuern oder die bestehenden weiter verkürzen? Einsendungen erfolgen bitte an <redaktion@uptimes.de>. Eine der folgenden UpTimes-Ausgaben zeigt die besten Einsendungen.

**Literatur und Links**

[1] NSA-Programm Stateroom, erklärt in *Der Spiegel*, Ausgabe 35/2013:
http://www.spiegel.de/spiegel/print/d-108794834.html
[2] Info zur Kryptographie-Methode *Salt* in *c't*, Ausgabe 13/2011:
http://www.heise.de/security/artikel/Passwoerter-unknackbar-speichern-1253931.html
[3] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu: Paper „Finding Collisions in the Full SHA-1“ anlässlich Konferenzvortrag auf der *Crypto 2005*: http://people.csail.mit.edu/yiqun/SHA1AttackProceedingVersion.pdf
[4] Kommando *sha1sum* aus den *GNU Coreutils*:
https://www.gnu.org/software/coreutils/manual/html_node/sha1sum-invocation.html

**Über Nils**

Diplom-Informatiker, Perl-Bewunderer und Bash-Basher Nils Magnus hat nach acht Jahren Security-Beratung knapp fünf Jahre als Redakteur für das Linux-Magazin gearbeitet. Seit rund drei Jahren hat ihn das System- und Netzwerk-Handwerk wieder, das er als Senior System Engineer der inovex GmbH ausübt.

# Aufgeschlossen

## Apache-Webzugang mit dem YubiKey Security Token absichern

Zwei-Faktor-Authentifizierung mit dem YubiKey erreicht mit wenig Aufwand mehr Sicherheit beim Zugriff auf Web-Anwendungen wie Fileserver oder Monitoringdienste. Dieser Artikel zeigt wie.

von **Werner Heuser und Frank Hofmann**

Jeder Schlüssel vertritt ein Misstrauen.

Emanuel Wertheimer

Auf viele Dienste greift man heutzutage über eine webbasierte Schnittstelle zu. Die Authentifizierung erfolgt über den Webserver, etwa einen Apache [1], Nginx [2] oder Lighttpd [3]. Diese benötigen meistens ein zusätzliches Modul, welches die Authentifizierung bereitstellt. Zur Anmeldung ist bislang die Verwendung von statischen Zugangsdaten aus Nutzername (Login) und Passwort üblich. Das erfordert, dass beide Seiten für deren Integrität und sichere Verwahrung Sorge tragen (siehe Kasten „Klassische Zugangssicherung“).

Mit Hilfe einer Zwei-Faktor-Authentifizierung sichert man einen Zugang deutlich stärker ab, beispielsweise gegen Passwortknacker: Kommen dynamische Daten oder Einmalpasswörter zum Einsatz, wird Abfangen und Stehlen der Daten auf dem Übertragungsweg nutzlos, insbesondere deren spätere zweckfremde Verwendung. Aktuelle Ereignisse wie Heartbleed [4] zeigen, dass Zwei-Faktor-Authentifizierung immer wichtiger wird. Mit der hier vorgestellten Lösung können auch kleine Unternehmen ihren Webzugang einfach und preiswert absichern.

Auf die YubiKey-Produktfamilie wurden die Autoren 2010 bei der Suche nach einer handhabbaren Zwei-Faktor-Authentifizierung aufmerksam. Beide Autoren überzeugt der praktische Nutzen des YubiKey sowie seine Bereitstellung als Debian-Paket durch den Hersteller. Beide setzen ihn selbst zur Absicherung ihrer Accounts und zunehmend in Kundenprojekten ein. Einer der beiden Autoren, Frank Hofmann, ist seit 2011 außerdem Yubico-Vertriebspartner in Deutschland.

## Zwei-Faktor-Authentifizierung

Im Alltag sind mehrere Strategien anzutreffen, die zur Erhöhung der Sicherheit des Netzzugangs und zur Authentifizierung an den Webanwendungen beitragen. Sensible Systeme nutzen zum Beispiel dynamische Passworte (Einmalpasswörter, engl. *One Time Passwords*, kurz OTP, [10]), die Wiederholungsaktionen durch abgefangene Zugangsdaten (Replay-Attacke) und Phishing verhindern. Insbesondere die so genannte Zwei-Faktor-Authentifizierung nutzt OTPs: Benutzername und Passwort bilden den Faktor 1, das OTP den Faktor 2. Vielen Anwendern ist dieses Verfahren aus dem Online-Banking vertraut: Kontonummer und Passwort bilden Faktor 1, die Transaktionsnummer (TAN) Faktor 2.

Aus technisch-organisatorischer Sicht realisieren Hersteller die Zweistufigkeit über eine Kombination aus Authentifizierungsdienst (Software) und Token (Hardware), mit mehr oder minder großen Aufwendungen. Ziel ist, dass der erhöhten und alltagstauglichen Sicherheit ein überschaubarer Aufwand für Anbieter und Benutzer gegenübersteht. Der vom skandinavischen Hersteller Yubico [11] entwickelte Authentifizierungstoken YubiKey nun (siehe Abbildung 1) macht es Unternehmen möglich, die Zwei-Faktor-Authentifizierung einfach und bezahlbar einzurichten. Auch für Benutzer ist der YubiKey leicht zu handhaben: Er erzeugt ein OTP, den der Benutzer eingibt und den ein Validierungsserver von Yubico auf Gültigkeit prüft.

Im Folgenden steht die Absicherung eines Apache-Webservers auf dem Programm (an anderer Stelle haben die Autoren über die Integration mit einer Secure Shell und PAM oder über mobile Nutzung von Smartphone und dem Passwortspeicher *LastPass* berichtet [12], [13], [14]). Die benötigten Software-Komponenten stehen unter einer freien Lizenz, Dokumentation stellt Hersteller Yubico bereit.

### Exkurs: Klassische Zugangssicherung

Der **Dienstbetreiber** hat die Zugangsberechtigungen sicher zu verwalten. Ebenso hat er zu prüfen und zu klären, dass die Benutzer-Freigaben stets korrekt vergeben sind. Ein weiterer Schritt ist die regelmäßige Prüfung des Passworts auf ausreichende Stärke und Knackbarkeit. In der Regel erfolgt das automatisiert als Hintergrundprozess anhand von Mindestkriterien wie Passwortlänge, aus der Statistik bekannten häufig verwendeten Passworten, darin enthaltenen Zeichenfolgen, sowie anhand eines Abgleichs gegenüber Wörterbüchern. Das dient dazu, verfeinerten und insbesondere verteilten Verfahren zum Aushebeln von Zugangssperren Einhalt zu gebieten. Falls das Passwort eine bestimmte Stärke unterschreitet, erhält der Benutzer die Aufforderung, seine Zugangsdaten zu ändern. Diese Vorgehensweise gilt sowohl für Standardzugänge via http, als auch für sichere Zugänge auf der Basis von SSL via https.

Jeder **Benutzer** ist angehalten, seine Zugangsdaten sicher aufzubewahren. Das kann über einen Schlüsselring oder einen Passwortdienst wie das kommerzielle *LastPass* [5] oder die freie Plattform *KeePass* ([6], [7]) erfolgen. Darüber hinaus ist es üblich geworden, die eigenen E-Mail-Identität als Vorsichtsmaßnahme vor Phishing- und Man-in-the-middle-Attacken zu überprüfen [8] und gegebenenfalls das Passwort zu wechseln. Hilfreich ist das Verständnis dafür, was ein sicheres Passwort auszeichnet [9].

Die Absicherung basiert auf zwei Voraussetzungen – erstens: der Apache-Webserver spricht SSL; zweitens: der Apache-Webserver hat ein zusätzliches Modul namens *libapache2-mod-authn-yubikey* [15] zur Authentifizierung via Webbrowser und YubiKey geladen (siehe Abbildung 2). Beide Schritte stellen wir Stück für Stück vor anhand eines Debian Wheezy mit frisch installiertem Apache-Webserver ohne zusätzliche Konfiguration – also so, wie die Paketverwaltung das Paket einspielt. Wessen Apache bereits über SSL verfügt, überspringt Schritt 1 und fährt bei der Zwischenüberschrift „Schritt 2: YubiKey-Authentifizierungsmodul hinzufügen“ fort.

Abbildung 1: Das YubiKey-Authentifizierungstoken in der Standardausführung.

## Schritt 1: Apache-Webserver SSL beibringen

Damit der Webserver gesicherte Verbindungen via Secure Socket Layer (SSL) zulässt, benötigt er neben einem SSL-Zertifikat das passende SSL-Modul. Das SSL-Zertifikat erwirbt man entweder von einem kommerziellen Anbieter, nutzt eines auf der Basis von CA Cert [16] – oder erzeugt sich kurzerhand ein eigenes. Letzteres ist für Testzwecke immer ganz praktisch, so auch in diesem Artikel.

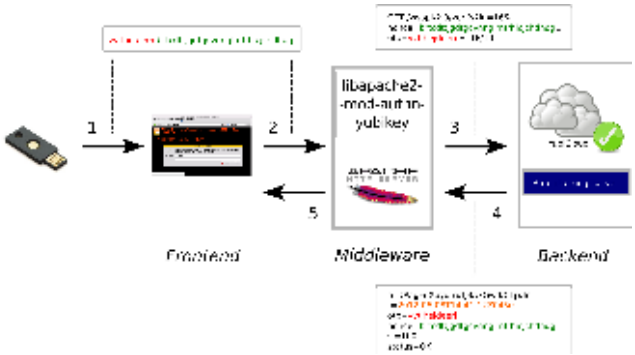



Abbildung 2: Zusammenspiel der Komponenten zur Authentifizierung.

Das X.509-Zertifikat erwartet der Apache-Webserver im Verzeichnis `/etc/apache2/ssl`. Das Werkzeug *openssl* erzeugt es wie folgt:

```
# openssl req -new -x509 -days 365 -nodes \
    -sha256 -out /etc/apache2/ssl/apache.pem \
    -keyout /etc/apache2/ssl/apache.pem
```

Dieses neue X.509-Zertifikat ist 365 Tage gültig (`-days 365`), nutzt SHA-256 (`-sha256`) als sicheres Hash-Verfahren und wird zu der lokalen Datei `/etc/apache2/ssl/apache.pem` gespeichert. Der Dateiname hinter `-out` steht für den Namen der Datei, in die der Public Key (das Zertifikat) wandert, und der Dateiname hinter `-keyout` für den Namen der Datei, in der der Private Key kommt. Das Vorgehen, beide Komponenten in der gleichen Datei nacheinander abzulegen, vereinfacht die Handhabung und ist im Bereich Webserver absolut üblich.

Das generierte Zertifikat holt folgender Aufruf auf das Terminal und sieht aus wie in Abbildung 3:

```
# openssl x509 -in apache.pem -inform pem -noout -text
```

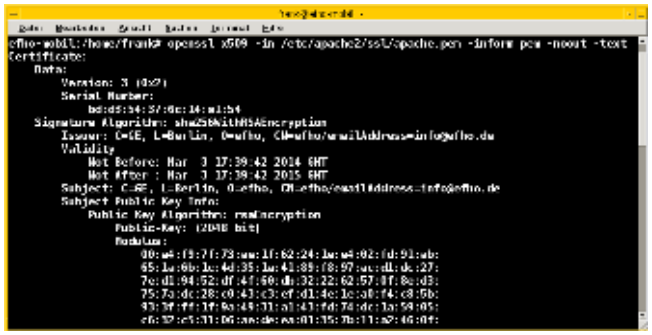


Abbildung 3: Selbst generiertes SSL-Zertifikat.

Nachfolgendes Kommando erzeugt einen symbolischen Link zum frisch erzeugten Zertifikat. Als Dateiname dient der Hash-Wert des Zertifikats, jedoch ergänzt um die Zeichenfolge `.0`:

```
# ln -sf /etc/apache2/ssl/apache.pem \
      /etc/apache2/ssl/`/usr/bin/openssl \
      x509 -noout -hash \
      < /etc/apache2/ssl/apache.pem`.0
```

Jetzt gilt es, die Zugriffsrechte auf die Datei mit dem Zertifikat und dem Private Key auf *600* einzuschränken, sodass nur Root Lese- und Schreibrechte besitzt. Das sieht so aus:

```
# chmod 600 /etc/apache2/ssl/apache.pem
# ls -la /etc/apache2/ssl
insgesamt 12
drwxr-xr-x 2 root root 4096 Mär  3 18:40 .
drwxr-xr-x 8 root root 4096 Mär  3 19:38 ..
lrwxrwxrwx 1 root root   27 Mär  3 18:40 \
  945c2981.0 -> /etc/apache2/ssl/apache.pem
-rw------- 1 root root 2985 Mär  3 18:39 apache.pem
```

Damit der Webserver nun SSL-Verbindungen akzeptiert, gibt man den TCP-Port 443 frei und lädt das entsprechende SSL-Modul für den Apache. Ersteres konfiguriert man in der Datei */etc/apache2/ports.conf*:

```
NameVirtualHost *:80
Listen 80
<IfModule mod_ssl.c>
   Listen 443
</IfModule>
```

Letzteres erledigt der Aufruf `a2enmod`. Er steht als Abkürzung für „Apache 2 Enable Module“:

```
# a2enmod ssl
Enabling module ssl.
...
To activate the new configuration, you need to run:
 service apache2 restart
```

Im nächsten Schritt aktiviert man die Einstellungen, die der Webserver zum Verarbeiten von SSL-Anfragen benötigt. Diese Einstellungen finden sich in der Datei */etc/apache2/sites-available/default-ssl*. Die Aktivierung erfolgt mit dem Kommando `a2ensite`. Es steht als Abkürzung für „Apache 2 Enable Site“:

```
# a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
 service apache2 reload
```

*a2ensite* legt einen symbolischen Link an von der Datei */etc/apache2/sites-available/default-ssl* zu den Einstellungen unter */etc/apache2/sites-enabled*. Nach Neustart des Webservers sind die Änderungen aktiviert. Wer jetzt eine Webseite aufruft – im Beispiel den Localhost – sieht, dass der Webserver jetzt eine verschlüsselte Verbindung öffnet, aber noch keine Zugangsdaten abfragt (Abbildung 4).

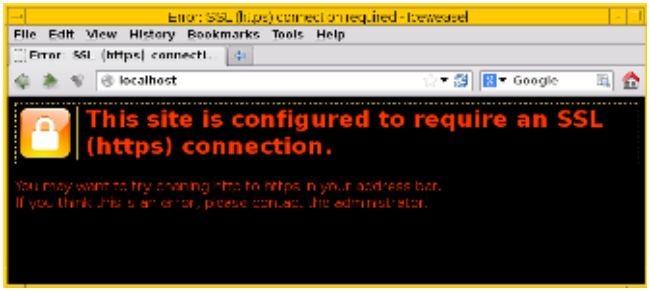



Abbildung 4: Mit aktiviertem Modul *authn_yubikey* begrüßt Apache nur noch über SSL.

## Schritt 2: YubiKey-Authentifizierungsmodul hinzufügen

Das YubiKey-Authentifizierungsmodul für den Apache ist im Debian-Paket *libapache2-mod-authn-yubikey* zu finden [15]. Nach der Installation des Paketes über die Paketverwaltung lädt das Kommando `a2enmod` das Modul:

```
# a2enmod authn_yubikey
Enabling module authn_yubikey.
To activate the new configuration, you need to run:
 service apache2 restart
#
# service apache2 restart
...
Restarting web server: apache2[Thu Mar 06 19:32:22 2014]
. ok
#
```

So weit, so gut. Jetzt fehlt in der Konfiguration des Webservers noch der Hinweis, dass er SSL-Anfragen via YubiKey authentifizieren soll. Diese Informationen ergänzt man im Abschnitt zum *VirtualHost* in der Datei `/etc/apache2/sites-enabled/default-ssl`.

Die Standardkonfiguration *VirtualHost _default_:443* sieht wie folgt aus – hier beispielhaft für das Verzeichnis `/var/www`:

```
01 <Directory /var/www/>
02   Options Indexes FollowSymLinks MultiViews
03   AllowOverride None
04   Order allow,deny
05   allow from all
06
07   LoadModule authn_yubikey_module \
/usr/lib/apache2/modules/mod_authn_yubikey.so
08   #AuthYubiKeyRequireSecure Off
09   AuthType Basic
10   AuthBasicProvider yubikey
11   AuthName "Please Log In using your YubiKey"
12   AuthYubiKeyTimeout 10
13   AuthYubiKeyTmpFile /srv/tmp/ykTmpDb
14   AuthYubiKeyUserFile \
/srv/appli/httpd/yubikeyUsers
15   AuthYubiKeyExternalErrorPage Off
16   Require valid-user
17 </Directory>
```

Relevant für die Authentifizierung sind die Eintragungen nach der Leerzeile: Option *LoadModule* (Zeile 07) lädt die Shared Library zur YubiKey-Authentifizierung von dem angegebenen Pfad. Option *AuthYubiKeyRequireSecure* (Zeile 08) bleibt auskommentiert, um stets eine verschlüsselte Verbindung zu erzwingen. Zeilen 09 bis 12 legen den Authentifizierungstyp *Basic* fest (Zeile 09), setzen als Validierungssystem den YubiKey (Zeile 10), bestimmen die Begrüßungsmeldung (Zeile 11) und begrenzen die Dauer der Gültigkeit, wenn sich ein Benutzer mit seinem YubiKey erfolgreich angemeldet hat (Zeile 12). Bei Letzterem gibt der Zahlenwert die Dauer in Sekunden an. Der Standardwert ist `43200`, das entspricht dem Zeitraum von 12 Stunden bis zur nächsten Authentifizierung.

Zeilen 13 und 14 bezeichnen die Liste der derzeit autorisierten Benutzer – also Nutzer mit einem erfolgreich validierten Zugang – sowie die Nutzerdatenbank. Die angegebenen Verzeichnisse sind frei wählbar, bedürfen aber eines Lese- und Schreibzugriffs durch den Webserver. Ist das nicht gegeben, kann der Apache die mittels YubiKey autorisierten Nutzer nicht wissen. Am Ende der Einstellungen ermöglicht Zeile 15, eine individuelle Fehlermeldung anzuzeigen, sollte die Autorisierung nicht erfolgreich gewesen sein. Der Standard ist Öffünd zeigt keine Fehlermeldung an.

Ohne die Angabe *Require valid-user* in Zeile 16 schließlich sind alle vorher genannten Einträge und generell die Autorisierung nutzlos. Diese Direktive weist den Webserver an, die bei ihm hinterlegten Webseiten nur auszuliefern, wenn sich ein Benutzer vorher erfolgreich autorisiert hat.

## Benutzer und Zugänge festlegen

Damit sich der Benutzer auf einem Webserver via YubiKey anmelden kann, bedarf es noch eines Benutzereintrags in der lokalen Nutzer-Datenbank des Webservers. Die Nutzerdatenbank steuert der Admin über die Angabe *AuthYubiKeyUserFile* in der Konfiguration des Webservers (siehe oben, Zeile 14).

Es bestehen zwei Möglichkeiten zur Authentifizierung – eine einfache Variante, und eine Zwei-Faktor-Authentifizierung. Für erste ist lediglich der Benutzername und die YubiKey-ID erforderlich, für die zweite zusätzlich ein statisches Passwort. Die YubiKey-ID ist die eindeutige Kennung des YubiKey, das heißt die ersten zwölf Zeichen des YubiKey-Strings, welchen ein Druck auf den Knopf des YubiKey ausgibt.

Für die einfache Variante sieht ein Eintrag in der Benutzerdatenbank wie folgt aus – die Reihenfolge zeigt die YubiKey-ID, einen Slash und danach den Benutzernamen, hier *testuser*:

```
# cat /srv/appli/httpd/yubikeyUsers
ccccccbeugur/testuser
```

Für die erweiterte Variante erzeugt das Kommando `htpasswd` einen Eintrag in der Benutzerdatenbank. Hilfreich sind die beiden Schalter `-c` für *create* (neue Datenbankdatei anlegen) und `-s` für *SHA* (verwendet Secure-Hash-Algorithmus anstatt der Funktion *crypt()*). Möglich ist auch `-m` für einen MD5-Hash, von dessen Verwendung die Autoren aber abraten, da MD5 seit 2008 als kryptographisch gebrochen einzuschätzen ist.

Im folgenden Kommandozeilenaufruf folgen nach den Schaltern noch der Dateiname der Benutzerdatenbank und der Benutzername. Der Benutzername ist hier die YubiKey-ID, da sich der Anwender darüber authentifiziert:

```
# htpasswd -cs /srv/appli/httpd/yubikeyUsers \
      ccccccbeugur
New password:
Re-type new password:
Adding password for user ccccccbeugur
# cat /srv/appli/httpd/yubikeyUsers
ccccccbeugur:{SHA}kUlgL50zAznmjF+ZEWWbkIedvh8=
```

Danach lädt man den Apache-Webserver neu, um die Änderungen an der Konfiguration wirksam werden zu lassen.

Als letzten Schritt surft man die Webseite *https://localhost* an und authentifiziert sich via Benutzername und YubiKey-Token (siehe Abbildung 5). Bei der einfachen Variante trägt der Anwender in das obere Feld den Benutzernamen ein, in

das untere Feld den Textstring, welcher der YubiKey durch den Knopfdruck generiert. Bei der Zwei-Faktor-Authentifizierung (Variante 2) gehört in das obere Feld wiederum den Benutzernamen ein. Das untere Feld erwartet das Passwort, gefolgt von dem Textstring. Zwischen den Passwort und dem Textstring darf kein zusätzliches Zeichen stehen.

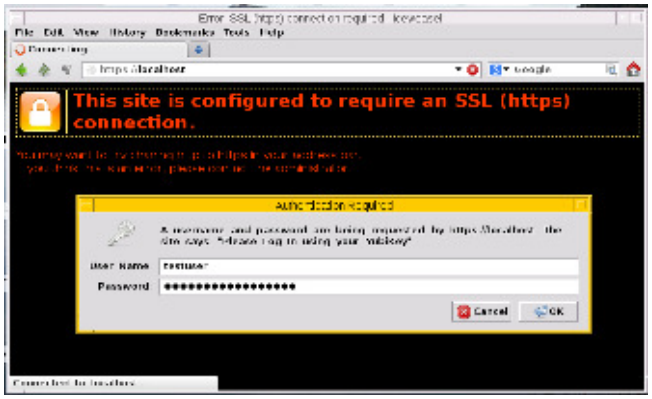



Abbildung 5: Zu den Inhalten der Webseite gelangt man jetzt nur nach vorheriger Authentifizierung via YubiKey.

In beiden Varianten prüft der Webserver zunächst Vorhandensein und die Gültigkeit des Benutzernamens und des Passworts in seiner lokalen Nutzerdatenbank. Ist der Eintrag korrekt, leitet er die Validierung des YubiKey-Textstrings ein. Dazu sendet er diesen an den Validierungsserver von Yubico, welcher ihm als Rückgabewert entweder `OK` oder im Fehlerfall `ERR` zurück liefert. Ist die Authentifizierung erfolgreich, liefert der Webserver die angesurfte Seite aus.

## Vor- und Nachteile

Der YubiKey erscheint praktisch, leicht und insbesondere betriebssystemunabhängig. Im Vergleich zu anderen Autorisierungslösungen – zum Beispiel SecurID [17] – ist er mit etwa 30 Euro in der Basisversion recht preiswert. Zum Lieferumfang zählt ausschließlich der YubiKey. Die zugehörige Software ist quelloffen und als Debian-Paket über die offiziellen Repositories erhältlich.

Der YubiKey selbst ist das Analogon zu einem physischen Schlüssel. Seine Crypto-Engine liegt in der Hardware vor. Wer einen YubiKey stiehlt, kann die Identität des Eigentümers annehmen. Wenn ein Identitätsdieb nicht über den YubiKey verfügt, kann er nur dann auf die gesicherten Ressourcen – hier die Webseiten – zugreifen, wenn er die zufällig generierte YubiKey-Zeichenkette nachahmt. Das dürfte schwer werden, da jeder YubiKey einzigartig ist.

Überdenkenswert ist hingegen das Telefonieren nach Hause, indem das Authentifizierungsmodul den Yubico-Server zur Überprüfung des YubiKey-Tokens anfragt. Diese Überprüfung setzt eine bestehende Verbindung zum Validierungsserver voraus. Ohne diesen Zugang und die positive Rückmeldung von diesem ist die angesurfte Webseite nicht erreichbar – auch wenn diese lokal ist. Zu hinterfragen ist die damit geschaffene Abhängigkeit von einem externen Validierungssystem, welches sich außerhalb von eigener Infrastruktur und eigenem Einflussbereich befindet. Auch eine Firewall kann zum Hindernis werden, die die Erreichbarkeit des Internets (stärker als erwartet) einschränkt.

Die besprochene Lösung hat zwei weitere Haken. Ist das YubiKey-Modul aktiviert, akzeptiert der damit versehene Webserver keine Zugriffe via http und den Port 80 mehr. Eine Kombination aus unsicheren und verschlüsselten Anfragen geht gemäß unseren Tests nicht.

Und: Die Sperrung oder Deaktivierung bestehender Benutzer erreicht man nur dadurch, dass man diese in der Benutzerdatenbank des Webservers per Hand auskommentiert oder austrägt. Immerhin: Um dies mit bestehenden Nutzerdatenbanken und Radius/SSO zu kombinieren, besteht das YubiX-Projekt [18]. Dessen Evaluierung steht unsererseits jedoch aus.

## Für wen ist die Lösung geeignet

Die hier vorgestellte Lösung über das zusätzliche Modul für den Apache-Webserver bündelt verschiedene, komplexe Technologien und Protokolle miteinander. Sie macht sie auch für Benutzer handhabbar, die weniger technikaffin sind. Der Sicherheitsgewinn ist beträchtlich und mit vergleichsweise wenig Aufwand in kurzer Zeit erreichbar. Ein ähnliches Modul steht auch für Nginx zur Verfügung [19], wurde aber von uns noch nicht getestet.

Die Autoren halten die Lösung insbesondere bei Infrastrukturen für geeignet, die für kleine Nutzerkreise bestehen und eine zusätzliche Absicherung benötigen. Das betrifft zum Beispiel von außen erreichbare Dienste und Systeme wie Fileserver und Blogs, aber auch Status- und Zustandsdienste wie Nagios.

Die physische Aufteilung in verschiedene Komponenten – Webserver und Validierungsdienst – wirkt sich bei der Nutzung mobiler Geräte wie Notebooks und Smartphones positiv aus. Diese Geräte haben den größten Teil des Tages

eine Internetverbindung, und die Kommunikation erfolgt von wechselnden Standorten aus über sich ändernde Kanäle verschiedener Dienstanbieter. Wer bestehende Infrastrukturen nutzt, auf die man als Benutzer keinen Einfluss hat, akzeptiert stets auch schwach oder unzureichend abgesicherte Zugänge und Verbindungen. Der YubiKey als zweiter Faktor ergänzt die Kommunikation mit wenigen Schritten um eine zusätzliche und vor allem wirksame Sicherheitsebene.

## Danksagung

Unser Dank gilt Wolfram Eifler, Thomas Osterried und Michal Bielicki für deren kritische Anmerkungen und Kommentare im Vorfeld dieses Beitrags.

**Quellen**

[1] Webserver Apache: http://projects.apache.org/projects/http_server.html
[2] Webserver Nginx: http://nginx.org
[3] Lighttpd: http://www.lighttpd.net
[4] Heartbleed: http://heartbleed.com/
[5] LastPass: https://lastpass.com
[6] Keepass2Android: http://keepass2android.codeplex.com
[7] KeePass: http://keepass.info
[8] Sicherheitstests des BSI: https://www.sicherheitstest.bsi.de/
[9] Stefan Schumacher: Sichere Passwörter. Vortrag auf dem Brandenburger Linux-Infotag (BLIT), Potsdam 2011:
http://blit.org/2011/zeitplan/attachments/74_HS02_2011-11-05_15_Starke-Passwoerter-vortrag.pdf
[10] Frank Hofmann: Sichere Benutzer-Authentifikation an sensiblen IT-Systemen. In: Stefan Schumacher und Jörg Sambleben, Informationstechnologie und Sicherheitspolitik – Wird der dritte Weltkrieg im Internet ausgetragen? Magdeburger Institutit für Sicherheitsforschung (MIS), 4. Ausgabe, Band 2/2012, S. 271 ff.:// http://www.sicherheitsforschung-magdeburg.de/uploads/journal/MJS-017.pdf
[11] Yubico: http://www.yubico.com
[12] Thomas Osterried, Frank Hofmann: Ausbuchstabiert – Sichere Authentifikation mit dem YubiKey, Teil 1 (erschienen in LinuxUser 09/2012):
http://www.linux-community.de/Internal/Artikel/Print-Artikel/LinuxUser/2012/09/Sichere-Authentifizierung-mit-dem-YubiKey
[13] Thomas Osterried, Frank Hofmann: Ausbuchstabiert – Sichere Authentifikation mit dem YubiKey, Teil 2 (erschienen in LinuxUser 10/2012):
http://www.linux-community.de/Internal/Artikel/Print-Artikel/LinuxUser/2012/10/Sichere-Authentifizierung-mit-dem-YubiKey-Teil-2
[14] Werner Heuser, Frank Hofmann: Praxistest YubiKey Neo als NFC-Authentifizierungs-Tag in Verbindung mit einem Smartphone. Vortrag im Berliner Büro 2.0:
https://www.buero20.org/veranstaltungen/b20-knowledge-space-yubikey-nfc/
[15] Paket *libapache2-mod-authn-yubikey* für Debian Wheezy:
https://packages.debian.org/wheezy/httpd/libapache2-mod-authn-yubikey
[16] CA Cert: http://www.cacert.org
[17] SecurID bei Wikipedia: http://de.wikipedia.org/wiki/SecurID
[18] YubiX Software Stack: https://www.yubico.com/develop/open-source-software/yubix/
[19] YubiKey-Modul für Nginx: https://github.com/sanderv32/ngx_http_auth_yubikey_module

**Über Werner und Frank**

Werner Heuser ist seit 15 Jahren freiberuflicher EDV-Sachverständiger für Laptops und Handys. Er ist Autor des Buches *Linux on the Road – A Guide for Laptops and Mobile Devices* und Technical Editor des Buches *Upgrading and Repairing Laptops* von Scott Mueller. 1999 gründete er die Firma *Xtops.DE* (http://xtops.de) und war der erste Anbieter von Laptops mit vorinstalliertem Linux-Betriebssystem. Sein aktueller Arbeitsschwerpunkt ist der professionelle Umgang mit mobilen Geräten wie Laptops, Tablets und Handys bei *Sentinel4Mobile* (http://sentinel4mobile.de).
Frank Hofmann hat Informatik an der Technischen Universität Chemnitz studiert. Derzeit arbeitet er in Berlin im Büro 2.0 (http://www.buero20.org), einem Open-Source-Expertennetzwerk, als Dienstleister mit Spezialisierung auf Druck und Satz (http://www.efho.de). Er ist Mitgründer des Schulungsunternehmens *Wizards of FOSS* (http://www.wizards-of-foss.de). Seit 2008 koordiniert er das Regionaltreffen der Linux User Groups aus der Region Berlin-Brandenburg.

## Ich kann Dich sehen
### Remote-Webcam an Linux-Rechner für Standbilder

Eine Kamera überträgt Linux-gesteuert Standbilder per Internet zu einem Server – ohne GUI und anderen Schnickschnack, also kein Skype, kein VLC. Nur die Kommandozeile. Natürlich ein bisschen Hardware. Und ein paar Haken und Ösen. Viel Spaß!

von **Jürgen Plate**

Die ersten 10.000 Aufnahmen
sind die schlechtesten.

Helmut Newton, Fotograf

Das Projekt: Die Kamera sollte die Wand einer Berghütte beziehen und alle Viertelstunden eine Bild der Umgebung schießen. Die Hütte war sehr spartanisch eingerichtet, es gab weder Internet noch fließend Wasser, und der Strom kam von einem Solarpanel. Ein kleiner Industrie-PC mit Atom-Prozessor sollte das Kamerabild einfangen, mit einigen Umwelt-Daten aufpeppen und per Surfstick in die weite Welt schicken. Da es teils eine private Liebhaberei und teils ein Hochschulprojekt war, zog sie sich länger hin – das Projekt wurde halt immer wieder mal aufgegriffen.

Es lohnt sich, einige Erkenntnisse aus der Vorauswahl von Kamera und aus dem Herumprobieren mit Tools zu schildern. Als Randbedingung spielte natürlich der Stromverbrauch des Gesamtsystems eine Rolle. Womit ich beim Thema bin. Denn man muss generell zwei Sorten von Kameras unterscheiden: die extrem preiswerten und einfachen USB-Webcams, die direkt am PC angeschlossen werden, und die nicht so billigen Netzwerkkameras, die per Twisted-Pair-Kabel am Netz hängen und ihren eigenen kleinen Web- und Streamingserver enthalten.

## Tauglichkeitstests: USB-Webcams

Für USB-Kameras gilt Ähnliches wie für TV-Karten: Ihre Nutzung erfordert die entsprechenden Treiber, bei Linux die Video4Linux-Treiber, kurz V4L2. Unterschiedliche Kameramodelle benötigen unterschiedliche Kernel-Module. Neben den USB-Modulen (`usbcore`, `usb-uhci`, `usb-ohci`), die fast immer schon laufen, braucht man noch das Input-Modul (`input`) und *Video for Linux 2* (`videodev`). Dann benötigt die Kamera noch das passende Modul (`ibmcam`, `cpia_usb`, `ov511`, `dc2xx`). Die Kernel-Module erzeugen die entsprechenden Einträge im Device-Verzeichnis in der Regel automatisch (meistens in `/dev/video0`).

Manch Linux-Urgestein wird sich noch an das Treiber-Theater mit proprietären CD-ROM-Laufwerken und später mit Scannern erinnern. Ähnliches Ungemach gab es anfangs mit den Webcams, und teilweise gibt es immer noch Kameras, bei denen zusätzliche Treiber nötig sind. Es empfiehlt sich eine WWW-Recherche, denn der Markt bei USB-Webcams ist so groß, dass man sich das nicht mehr antun muss.

Zur Identifikation der von der Webcam verwendeten Treiber holen nach dem Einstecken an einen Linux-Rechner *lsusb* und `dmesg` nützliche Informationen. Eine Ausgabe von `dmesg` sieht auszugsweise aus wie im folgenden Listing – in unserem Fall sind es zwei verschiedene Kameras, aber bei beiden der Treiber *uvcvideo*.

```
[ 12.621009] Linux video capture interface: v2.00
[ 12.685253] uvcvideo: Found UVC 1.00 device USB 2.0 Camera \
  (0c45:6340)
[ 12.712522] input: USB 2.0 Camera as /devices/pci0000:00/\
  0000:00:1d.7/usb1/1-1/1-1:1.0/input/input8
[ 12.713513] uvcvideo: Found UVC 1.00 device HD 720P Webcam \
  (0603:8f01)
[ 12.718307] input: HD 720P Webcam as /devices/pci0000:00/\
  0000:00:1d.7/usb1/1-4/1-4:1.0/input/input9
[ 12.718700] usbcore: registered new interface driver uvcvideo
[ 12.718709] USB Video Class driver (1.1.1)
```

Falls der Befehl *dmesg* keine hilfreiche Meldung zeigt, kann man mittels *lsmod* die Liste aller geladenen Module ansehen und dort den zu der Webcam passenden Treiber herausfinden. Dazu die Webcam abstecken und `lsmod` aufrufen. Dann die Webcam anstecken und nochmals `lsmod` aufrufen. Das neu aufgetauchte Modul ist der Treiber der Webcam.

Achtung: Falls man einen Laptop mit eingebauter Webcam besitzt, sind eventuell zwei Module für Webcams geladen (außer, die interne und die externe Webcam nutzen den gleiche Chipsatz). Sieht man etwas Änliches wie oben, kann man

zufrieden sein – insbesondere scheint UVC inzwischen zuverlässig zu funktionieren – und es wird kein spezieller Treiber benötigt. Der Rechner erkennt dann nämlich eine Kamera und bindet sie über `input0` an. Das trifft inzwischen auf eine ganze Reihe von Modellen zu. Normalerweise wird auch das Device `/dev/video0` installiert (oder auch `/dev/video1` und so weiter), über das die Kamera angesprochen wird. Zur Sicherheit (oder etwa vor einem Kauf) sollte man den Namen der Webcam googeln und nachsehen, ob in den technischen Daten etwas von *UVC* steht.

## Von der USB-Webcam auf den Rechner

Ist die Webcam soweit in Ordnung, gehören auf das System jetzt die Webcam-Anwendung *fswebcam* [1] sowie zwei Bildbetrachter, *gpicview* für X [2] und *fbi* [3] für die Konsole. Damit lässt sich probeweise ein Bild einfangen. Bei Debian-Linux und dessen Abkömmlingen erledigen beides folgende Anweisungen:

```
apt-get install fswebcam gpicview fbi
fswebcam -v -r "640x480" test.jpg
gpicview test.jpg
```

Im Beispiel entsteht ein Bild namens *test.jpg* im aktuellen Verzeichnis und wird anschließend angezeigt – 640 mal 480 Pixel sollte jede Kamera können. Wer keinen X-Server und kein grafisches Interface laufen hat, betrachtet das Bild mit dem *Frame Buffer Imager* (`fbi`) – allerdings geht dies natürlich nur auf einer realen Konsole am Rechner, nicht per SSH:

```
fbi test.jpg
```

Unterstützt der Rechner V4L/V4L2 und eben UVC, so gibt es eine Reihe von weiteren Tools für die Weiterverarbeitung der Bilder, etwa das Paket *motion* oder auch `luvcview`. Mein aktuelles Projekt überträgt jedoch nur Einzelbilder zu bestimmten Zeiten. `ffmpeg` für einen Stream benötige ich daher nicht.

Das eingesetzte Tool *fswebcam* ist ein kleines und ausgefeiltes Webcam-Programm. Es kann eine Anzahl von Frames von den meisten Geräten lesen, die V4L2-kompatibel sind, dann deren Mittelwert bilden, um das Rauschen zu reduzieren, und schließlich einen Titel auf das Resultat zeichnen. Dazu verendet es die Grafikbibliothek *GD*, die auch die Kompression des Bildes in PNG oder JPEG vornimmt. Letztere ist bei den meisten Distributionen standardmäßig installiert.

Das Tool nimmt für die Snapshots eine Reihe von Einstellungen entgegen. Bei Außenaufnahmen können auch proprietäre Parameter der Kamera mittels der Option `-s` durchgereicht werden. Für Innenaufnahmen ist die Parameterzahl überschaubarer. Ein Aufruf könnte etwa lauten:

```
fswebcam -v -S 5 -F 2 -r "1280x720" \
  -d /dev/video0 --no-banner snap.jpg
```

Was die Optionen dieses Befehls bewirken, zeigt Tabelle 1 zeigt.

| | |
|---|---|
| `-v` | detaillierte Ausgabe („verbose mode“) |
| `-S 5` | fünf Frames überspringen und erst die folgenden speichern (die Kameras müssen oft sozusagen erst einrasten) |
| `-F 2` | zwei Frames speichern |
| `-r "1280x720"` | legt die Auflösung des Bildes fest, hier HD 1280 x 720 Pixel |
| `-d /dev/video0` | gibt an, welches Device die Bilder aufnehmen soll |
| `-no-banner` | unterdrückt Einblendungen wie die Streifchen im Fernsehen, auf denen steht, wer gerade ins Mikrofon stottert. Fswebcam kann das auch und setzt einen Standardbanner, welch selbiger sich unterdrücken lässt. („verbose mode“) |

Tabelle 1: Hier verwendete Optionen des Kommandos `fswebcam`

Zur Option `-S` sei bemerkt, dass die ersten vorbeisausenden Frames manchmal nicht akzeptabel sind, da eine kurz zuvor eingestellte Kameraempfindlichkeit noch nicht gegriffen hat. Deswegen ist es sinnvoll, einige Frames zu überspringen. Es ist auch möglich, mittels `-D` *n* die Wartezeit von n Sekunden einzuschieben.

Wenn nicht die Option `-r` eingestellt ist, nimmt das Programm die Standardauflösung – das ist meistens nicht die optimale, in der Regel 352 mal 288 Pixel. Man erhält aber viel bessere Bilder, wenn man als Auflösung zum Beispiel 800 mal 600 Pixel angibt. Oft ist die maximal mögliche Auflösung – etwa 1200 mal 1600 Pixel – auch nicht optimal, daher fährt man mit einer etwas geringeren Auflösung besser.

Das Prográmmchen *fswebcam* kann fast alles selbst, man muss es nur konfigurieren. Dazu gehört beispielsweise auch das Einfangen eines Bildes in regelmäßigen Zeitabständen. Persönlich neige ich dazu, solche Aufgaben dem *cron*-Dämon zu überlassen. Interessanter ist die Möglichkeit, das Bild automatisch mit einer Beschriftung am unteren oder oberen Rand zu verse-

hen, wobei Hintergrund- und Schriftfarbe, Transparenz, Schriftgröße usw. frei wählbar sind. Standardmäßig erfolgt die Beschriftung am unteren Bildrand. Die Optionen für die Beschriftung zeigt Tabelle 2.

| | |
|---|---|
| `-title "xxx"` | Titel (1. Zeile links) |
| `-subtitle "xxx"` | Untertitel (2. Zeile links) |
| `-timestamp "xxx"` | Datum/Uhrzeit (1. Zeile rechts, Beispiel: "%d.%m.%Y %H:%M") |
| `-info "xxx"` | Zusatzinfo (2. Zeile rechts) |

Tabelle 2: Optionen für die automatische Bildbeschriftung.

Die Timestamp-Option beherrscht alle Formen der Zeitformatierung, die von der C-Funktion `strftime()` unterstützt werden. Meine Optionen-Auswahl sieht wie folgt aus:

```
fswebcam -D 3 -S 10 -F 10 -r 1280x720 \
  -d /dev/video0 --title "Kamera 1" \
  --subtitle "http://www.netzmafia.de" \
  --timestamp "%d.%m.%Y %H:%M" \
  --info "Blick auf den Monte Podice" $FILE
```

Für Titel, Untertitel, Info etc. könne auch variable Werte verwendet werden, beispielsweise durch Übergabe in Shell-Variablen. Wenn es keine variablen Parameter gibt, die sich bei jedem Aufruf ändern, kann die gesamte Konfiguration auch in einer Datei untergebracht werden, was den Aufruf auf der Kommandozeile entsprechend verkürzt. Wenn *fswebcam* sich beschwert, dass es keinen Font findet, übergibt diesen die Option `--font "`*Pfad zum Font*`"`.

Die Option `--flip` erlaubt das horizontale oder vertikale Spiegeln des Bildes – etwa bei Deckenmontage der Kamera – zum Beispiel in der Form `--flip v` oder `--flip h,v`. `--rotate` *www* rotiert das Bild um den angegebenen Winkel *www*, wobei jedoch nur die Werte 90, 180 und 270 möglich sind. Die Option `--crop` *groesse,offset* erlaubt Bildausschnitte an verschiedenen Positionen. So extrahiert `--crop 320x240` einen zentralen Ausschnitt von 320 mal 240 Pixel Größe. `--crop 100x100,0x0` produziert dagegen einen 100 mal 100 Pixel großen Ausschnitt der linken oberen Ecke. Mittels `--scale` *groesse* lässt sich das Bild auch auf eine gewünschte Größe skalieren: `--scale 640x480` wäre ein Beispiel.

## Tuning: Kamerainformationen abfragen

*fswebcam* erlaubt auch die Abfrage der Kamera in Hinblick auf die Möglichkeit, kameraspezifische Steuerungsvariablen wie Kontrast oder Helligkeit zu ändern. Mit dem Parameter `--list-controls` werden alle proprietären Werte abgefragt, die der Treiber verarbeiten kann. Abbildung 1 zeigt eine solche Abfrage.

Die numerische Spannweite der Werte kann von Kamera zu Kamera sehr unterschiedlich sein – zum Beispiel reicht die Helligkeit bei einem Modell von -64 bis +64 und bei einem anderen von -128 bis +127. Insofern ist Vorsicht angesagt, wenn man Werte für die Controls an die Kamera sendet. Um der Gefahr zu entgehen, gegebenenfalls falsche Werte zu übergeben, erlaubt *fswebcam* Prozentangaben. Wird etwa die Helligkeit mit 20 Prozent angegeben, rechnet das Programm den entsprechenden Zahlenwert aus (gerundet auf ganze Zahlen). Die 20 Prozent ergäben einmal -51, bei einem anderen Modell -77. Die Einstellung erfolgt über den Parameter `-s` und immer in der Form *Bezeichnung=Wert*, zum Beispiel:

```
-s "Backlight Compensation=75%"
-s "Brightness=35%"
-s "Contrast=45%"
```

```
fswebcam -d /dev/video0 --list-controls

--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
Available Controls      Current Value  Range
------------------      -------------  -----
Brightness              10 (57%)       -64 - 64
Contrast                21 (32%)       0 - 64
Saturation              64 (49%)       1 - 128
Hue                     0 (50%)        -40 - 40
Gamma                   72 (0%)        72 - 500
Gain                    0 (0%)         0 - 100
Power Line Frequency    50 Hz          Disabled
                                       | 50 Hz
                                       | 60 Hz
Sharpness               4              0 - 6
Backlight Compensation  1              0 - 2
Adjusting resolution from 384x288 to 352x288.
--- Capturing frame...
Captured frame in 0.00 seconds.
--- Processing captured image...
There are unsaved changes to the image.
```

Abbildung 1: Abfrage, die die proprietären Werte eines Devices anzeigt.

Im Test befanden sich Kameras unterschiedlicher Hersteller und im Preisbereich von zehn bis 30 Euro. Unter anderem wurden zwei preiswerte Kameras bei Pollin erworben, eine Logilink UA0155 [4] und eine HAMA Digital Eye II [5] (Abbildung 2), die beide sofort unter Linux liefen und ordentliche Bilder lieferten. Bei beiden muss man aber die Schärfe von Hand einstellen.


Abbildung 2: Unter anderem wurden die Kameras Logilink UA0155 und HAMA Digital Eye II getestet.

Etwas teuerer waren die Logitech C270 USB Webcam [6] und die Microsoft LifeCam HD-3000 [7], die in Bild 3 zu sehen sind. Beide Kameras liefen ebenfalls sofort unter Linux, obwohl in den technischen Daten (wie erwartet) immer nur Windows als Betriebssystem angegeben war. Beide Kameras haben HD-Auflösung (1280 mal 720 Pixel) und ein integriertes Mikrofon mit Rauschunterdrückung. Die Logitech C270 erlaubt daneben noch Fotos, die per Software-Interpolation etwa einer 3-Megapixel-Kamera entsprechen (Windows-Software-Version). Beim Einfangen von Bildern meldete *fswebcam* zwar eine kleine Unsauberkeit bei den angelieferten Daten, konnte aber ein korrektes JPG-Bild speichern. Bei schwacher Beleuchtung wirken die Bilder allerdings etwas flau.

Abbildung 3: Die getesteten Kameras Logitech C270 und Microsoft LifeCam HD-3000.

Eine Überraschung bot die Microsoft LifeCam HD-3000. Abgesehen davon, dass sie – obwohl von Microsoft – unter Linux das Gewünschte leistete, lieferte sie auch noch die besten Bilder von allen vier USB-Kameras. Außerdem kann sie auch noch 16:9-Breitbild für Videoaufzeichnungen im Kinoformat. Des Weiteren darf jegliche Helligkeitsanpassung entfallen, denn die sogenannte Truecolor-Technologie sorgt automatisch für kräftige Farben bei nahezu allen Lichtverhältnissen. Die Software-Interpolation entspricht bei der HD-300 sogar einer 4-Megapixel-Kamera.

## Tauglichkeitstests: Netzwerk-Kameras

Die oben beschriebenen Methoden lassen sich fast ohne Änderung auch auf Kameras anwenden, die über Netzwerkkabel oder WLAN erreichbar sind. Eine Netzwerk-Kamera liefert ebenfalls Einzelbilder oder Videoströme. Der Vorteil solcher Kameras besteht einerseits darin, dass die Kamera an einer beliebigen Stelle im Netz stehen kann und nicht an die Nähe eines Rechners gebunden ist. Andererseits werden bereits fertige JPG-Bilder oder MPEG-Videoströme über ein Webinterface geliefert, man hat also keinerlei Treiberprobleme. Nachteilig ist der wesentlich höhere Preis der Kameras, der sich auf das Drei- bis Vierfache einer guten USB-Kamera beläuft. Auch der Energiebedarf (Stromaufnahme) ist beträchtlich höher – schließlich birgt die Kamera in der Regel einen kompletten Computer.

Für meine Tests habe ich vier Kameras verwendet: Die ältere Axis 207 [8] stellt in vielen Bereichen einen Standard dar, nur ihre Bildauflösung ist heute nicht mehr up-to-date. Die Trendnet TV-IP100 [9] gab es vor Jahren recht preiswert, ebenso wie die Logilink WC0040 [10]. Die Grandtec Megapixel IP Camera [11] bietet eine hohe Bildauflösung.

Die ersten drei Kameras liefern ein Farbbild von maximal 640 x 480 Pixeln, sind eben nicht mehr die neuesten Modelle. Die Grandtec schafft 1280 x 1024 Pixel. An dieser Stelle kommt es jedoch nicht so sehr auf die Auflösung oder andere Kameraeigenschaften an, sondern die vier Modelle sollen einfach die Möglichkeiten illustrieren, wie auf solche Kameras zugegriffen wird. Oft lassen sich die Methoden dann auf andere Modelle übertragen. Die Axis-Kamera hat ein Fixfocus-Objektiv, bei den anderen muss man die Schärfe manuell regulieren (Abbildung 4).

Abbildung 4: Alt und jung: die Kameras von Grandtech und Axis.

Über ein Webinterface oder ein mitgeliefertes Windows-Tool sind nicht nur der Videostrom oder Einzelbilder abrufbar, sondern es lassen sich

auch beliebige Einstellungen vornehmen. Bei Inbetriebnahme muss die Kamera zunächst eine IP-Adresse erhalten. Bei der Gelegenheit kann man auch gleich die anderen Grundeinstellungen vornehmen. Beim Abgreifen der Bilder wird die Kamera über das Netzwerk angesprochen.

Zum Herunterladen der Bilder können auf der Kommandozeile die Programme *wget* oder *curl* verwendet werden. Bei der Trendnet TV-IP100 war etwas Forschungsarbeit nötig, da per Default das Abliefern der Bilder nur per E-Mail oder FTP offeriert wird. Es ging dann doch recht einfach: Im Webinterface beim Browser unter *Ansicht* den Seitenquelltext aufrufen und im HTML-Wust nach einem IMG-Tag suchen. Er präsentiert sich hübsch verpackt in einer Tabelle, aber mit etwas seltsamer Dateiangabe:

```
<IMG SRC=\
  "IMAGE.JPG?cidx=2009117184335312242" BORDER="0">
```

Der Schwanz `?cidx=200911...`“ ist eigentlich unnötig: Es handelt sich um einen Trick der Kamerasoftware, um die Browser auszutricksen. Da die Webseite immer gleich aussieht, würde ein schlauer Browser beim Klicken auf *Reload* nicht das aktuelle Bild der Kamera abrufen, sondern das im Browser-Cache. Durch das Anhängen einer Pseudo-Formulareingabe mit sich ändernden Werten wird dafür gesorgt, dass immer das aktuelle Bild geholt wird. Bei der Trendnet TV-IP100 erhält man bei folgendem Aufruf ein Bild und nichts sonst:

```
curl -o snap.jpg http://192.168.2.100/IMAGE.JPG
```

Bei der Logilink WC0040 funktioniert der Abruf eines Standbildes genauso einfach, im Beispiel mit User- und Passwortangabe – was die Kamera immer haben will:

```
curl -o snap.jpg \
  http://user:pass@192.168.2.102/snapshot.jpg
```

Die Axis-Kamera macht etwas Ähnliches wie die Trendnet. Sie liefert das Bild scheinbar über ein CGI-Script aus, was ebenfalls dafür sorgt, dass sich der Browser immer ein frisches Bild holt. Bei der Axis 207 können Parameter wie Helligkeit, Bildgröße usw. mitgegeben werden. Im folgenden Beispiel gebe ich die Bildgröße vor (die URL wurde aus drucktechnischen Gründen umbrochen, sie muss in einer Zeile stehen):

```
curl -o snap.jpg \
  http://192.168.2.101/axis-cgi/jpg/\
  image.cgi?resolution=640x480
```

Es steht zwar nicht im Handbuch, aber wenn man nur ein Bild ohne Einstellungen will, genügt auch:

```
curl -o snap.jpg \
  http://192.168.2.101/axis-cgi/jpg/image.jpg
```

Für Video steht in den letzten beiden Feldern der URL stattdessen `/mjpg/video.mjpg`.

## Dompteur für Netzwerk-Cam unter Linux

Die Grandtec Megapixel IP Camera sträubte sich gegen die Zusammenarbeit, obwohl die Reklame ausdrücklich Linux-Kompatibilität aufführte. Ihre Windows-Software – zu Linux gab es wieder mal nix – kommt mit massenhaft Active X und derlei Gefrickel. Etwas Forschungsarbeit ließ aber den Verdacht aufkommen, dass unter der Haube ein ARM-Linux werkelt – wenn auch mit gewöhnungsbedürftigem Interface. Innerhalb des Mini-Webservers, der auf der Kamera lief, fand sich nämlich das Verzeichnis */cgi-bin* mit einigen vielversprechenden Kommandos. Leider zeigte sich, dass auch ein Kommando mit dem Namen *still.cgi* keineswegs ein Standbild lieferte, sondern einen MJPEG-Stream. Immerhin ohne Active X oder anderen bösen Dingen, aber mit Abfrage von Username (immer *root*) und Passwort.

Darauf ließ sich jedoch schon einmal aufbauen. Mit dem Programm *curl* kann das Video geladen und als JPG-Datei gespeichert werden. *curl* wurde eingesetzt, weil sich hier im Gegensatz zu *wget* die Downloadzeit begrenzen lässt. Also fischt man sich eine kurze Videosequenz heraus (Zeile 01), aus der dann *avconv*, das früher *ffmpeg* hieß, ein Frame extrahiert (Zeile 02), und schon hat man ein Standbild, die Videosequenz kann weg (Zeile 03) – es gibt immer eine Möglichkeit, eine störrische Netzwerk-Kamera zu zähmen:

```
01 curl -o video.jpg -m 3 \
  http://root:xxx@192.168.2.103/cgi-bin/still.cgi
02 avconv -i video.jpg -s 1280x1024 snap.jpg
03 rm video.jpg
```

Nach dem Einfangen des Bildes in der Datei `snap.jpg` wird noch etwas Kosmetik betrieben, was natürlich bei allen Kameras auf die gleiche Weise erfolgen kann. Bei den Netzwerkkameras fehlen uns jedoch die Features von *fswebcam* zum Einfügen von Beschriftungen im Bild. Da hilft das in vielen Linux-Distributionen enthaltene Imagemagick-Paket weiter, insbesondere das darin enthaltene Programm *convert*. ImageMagick [12] ist ein freies Softwarepaket zur Erstellung und Bearbeitung von Pixelgrafiken, es kann nahezu alle üblichen Bildformate lesen, verändern und schreiben, außerdem lassen sich Bilder dynamisch

erzeugen (Tabelle 3). Für Perl-Programmierer gibt es sogar passende Module.

| Tool | Beschreibung |
|---|---|
| *convert* | Liest, bearbeitet und speichert Bilder. |
| *mogrify* | Wie *convert*, nur dass die Eingabedatei durch die Ausgabe ersetzt wird. |
| *animate* | Spielt mehrere Bilder schnell hintereinander ab. |
| *compare* | Vergleicht zwei Bilder und gibt den Unterschied als Bilddatei aus. |
| *composite* | Überlagert mehrere Bilder zu einem einzigen Bild. |
| *conjure* | Führt Skripte in der Skriptsprache von ImageMagick aus. |
| *display* | Stellt Bilder auf einem X-Server dar. |
| *identify* | Gibt Eigenschaften wie Dateiformat oder Maße von Bilddateien aus. |
| *import* | Macht Bildschirmfotos. |
| *montage* | Fasst mehrere Bilder zu einem großen Einzelbild zusammen. |
| *stream* | Liest Teile von Bilddateien und gibt sie als Rohdaten, Gleitpunktzahlen oder Ähnlichem aus. |

Tabelle 3: *ImageMagick* enthält diese Tools mit je eigener Manpage, die alle auf einen gemeinsamen Satz von Bibliotheken zugreifen.

Für einen eindeutigen Dateinamen eines jeden Bildes werden Datum und Uhrzeit herangezogen (Zeile 01). Dann wird auch noch das Bild mit Datum und Uhrzeit beschriftet (Zeile 02). Die Beschriftung erfolgt je nach Bildhelligkeit in weiß oder schwarz (ab Zeile 03). Am Ende wird die Urspungsdatei nicht mehr benötigt (Zeile 10):

```
01 FILE=$(date "+%d%m%Y%H%M")
02 TS=$(date "+%d.%m.%Y   %H:%M")
03 col=$(Brightness snap.jpg)
04 if [ $col -gt 50 ]
05 then
06   convert -fill white \
             -gravity SouthEast \
             -pointsize 20 \
             -draw "text 15,15 '$TS'" \
               snap.jpg ${FILE}.jpg
07 else
08   convert -fill black \
             -gravity SouthEast \
             -pointsize 20 \
             -draw "text 15,15 '$TS'" \
               snap.jpg ${FILE}.jpg
09 fi
10 rm snap.jpg
```

Die Größe eines Bildes beträgt, je nach Auflösung, um die 30 bis 150 KByte. Ein Tag hat 86.400 Sekunden, ein 30-Tage-Monat 2.592.000 Sekunden. Wenn Sie alle 10 Sekunden ein Bild speichern, um später einen Zeitraffer-Film daraus zu machen, sind das ungefähr 7,8 bis 39 GByte pro Monat. Also auf den Plattenplatz achten und rechtzeitig alte Dateien löschen.

Wenn die Kamera bei höchst unterschiedlichen Lichtverhältnissen ein halbwegs brauchbares Bild liefern soll, muss oftmals die Helligkeit justiert werden – sofern die Kamera das nicht selbst schon erledigt. Dazu muss das Bild zweimal kurz hintereinander aufgenommen werden. Das erste Bild (Zeile 01) dient nur der Lichtbestimmung (02), das zweite Bild wird dann mit der ermittelten Helligkeit aufgenommen – Beispiel für eine USB-Webcam:

```
01 fswebcam -q -D 3 -S 10 -F 10 -r 1280x720 \
        -d $CAM \
        --no-banner $TMP/$FILE
02 HELL=$(Brightness $TMP/$FILE)
03 fswebcam -D 3 -S 10 -F 10 -r 1280x720 \
        -d $CAM \
        -s "Brightness=${HELL}\%" \
        --no-banner $TMP/$FILE
```

Die Shell-Option *Brightness* liefert den Lichtwert in Prozent und ergibt sich aus der Differenz von 100 und der ermittelten Helligkeit in Prozent. Ist das Bild dunkel, wird der Wert für den Kamera-Parameter *Brightness* also hoch, ist es hell, verkleinert sich der korrigierende Lichtwert. Wer sich dafür interessiert, wie man eigentlich die Helligkeit eines Bildes ermittelt, findet die Antwort im Kasten „Exkurs: Helligkeit eines Bildes bestimmen“. Zum Glück kann man die Arbeit dem Universaltool ImageMagick überlassen und man muss nicht selbst programmieren.

Das Programm *convert* kann nicht nur alle möglichen Bildmanipulationen vornehmen, sondern auch diverse Infos über ein Bild liefern. Das Problem ist eher, aus dem Datenwust den richtigen Wert herauszufischen. Dazu wird das Bild erst in Graustufen umgewandelt (Zeile 03) – damit hat schon jeder Bildpunkt nicht mehr drei, sondern nur noch einen Helligkeitswert. Dann wird mittels *sed* der Helligkeitswert *mean* herausgefischt. Dieser Wert liegt zwischen 0 und 1 und muss mit 100 multipliziert werden, um eine Prozentangabe zu bekommen (Zeile 04). Der Prozentwert wird dann noch von 100 subtrahiert, was der Kommandozeilenrechner *bc* erledigt (Zeile 05). Zum Schluss werden mit dem Sed-Kommando noch die Nachkommastellen abgeschnitten:

```
01 Brightness()
02   {
03   local data=`convert $1 \
       -colorspace gray \
       -verbose info: `
04   local mean=`echo "$data" \
       | sed -n '/^.*[Mm]ean:.*[(]\([0-9.]*\).*$/\
       { s//\1/; p; q; }'`
05   echo "100-$mean*100" | bc \
       | sed -e 's/\..*$//'
06   }
```

### Exkurs: Helligkeit eines Bildes bestimmen

Mathematisch kann der Lichtwert eines Grauwertbildes als Mittelwert aller Grauwerte, der Kontrast als Varianz aller Grauwerte definiert werden. Die einfachste Möglichkeit zur Bestimmung der Helligkeit eines Pixels – also dessen Grauwert – ergibt sich daher mit der Formel *(R+G+B)/3* (Zeile 12), wobei R, G und B die Werte für die drei Grundfarben Rot, Grün und Blau sind. Da jede Farbe einen Wert zwischen 0 und 255 annehmen kann, ergibt sich der gleiche Bereich für den Lichtwert (Zeile 11). Aus dem Lichtwert aller x- und y-Bildpunkte (Zeile 16) errechnet sich am Ende die Gesamthelligkeit des Bildes `avg` (Zeile 17). In folgendem C-Programmfragment ist `t_img` (Zeile 01) einfach irgendeine Struktur, die alle Bildinfo enthält.

```
01 int brightness(t_img *bild)
02   {
03   int  x, y, br, avg;
04   t_color val;
05   double  sum;
06
07   for(x = 0; x < bild->xsize; x++)
08     {
09     for(y = 0; y < bild->ysize; y++)
10       {
11       getpixel(bild, x, y, &val);
12       grau = (val.R + val.G + val.B)/3;
13       sum += grau;
14       }
15     }
16   avg = sum / (bild->xsize * bild->ysize);
17   return avg;
18   }
```

Nur lässt diese Form der Helligkeitsbestimmung die Wahrnehmungseigenschaften unsers Auges außer Acht: Wir nehmen den gelb-grünen Farbbereich wesentlich heller wahr als die anderen Farben. Das wird in anderen Farbmodellen berücksichtigt. So könnten Sie anstelle der linearen Umrechnung auch den RGB-Farbraum in das YUV-Farbmodell umrechnen. YUV verwendet zur Darstellung der Farbinformation zwei Komponenten: Die Luminanz Y beschreibt den Lichtanteil. Die Chrominanz beschreibt den Farbanteil und besteht aus den zwei Unterkomponenten U und V, die den Rot- und Blau-Lichtpegel reduzieren:

```
Y = R x 0.299 + G x 0.587 + B x 0.114
U = (B - Y) x 0.493
V = (R - Y) x 0.877
```

Für die Helligkeitsbestimmung wird nur der Y-Wert benötigt.

*convert* ist ein echter Tausendsassa. So kann man das Bild auch gleich noch verkleinern und mit höherer Kompressionsrate (aber geringerer Qualität) speichern, um Platz zu sparen. Oder man macht Graustufen-Bilder daraus. Oder, oder, oder.

## Und Action – Bewegung erkennen

Wer Bewegungserkennung von Hand implementieren möchte, wendet sich dem Kasten „Exkurs: Bewegung erkennen“ zu. Ausgefeilter kann das alles aber ein Programm zur Motion Detection (MD). Bei manchen Kameras ist MD bereits implementiert. Oft wird auch ein Tool für Windows mit der Webcam geliefert. Natürlich gibt es dergleichen auch für Linux. Einer der bekanntesten Vertreter dieser Programmgattung kann auch wieder über die Kommandozeile gestartet werden.

Das Programm *motion* ([13],[14]) empfängt nonstop Bilder von beliebig vielen Webcams oder Netzwerkkameras. Ändert sich dabei von einem Bild zum anderen eine definierte Anzahl Pixel, schlussfolgert das Programm, dass sich etwas im zu überwachenden Bereich bewegt. In diesem Fall nimmt es mit Hilfe von *ffmpeg* einen Videostream oder eine Serie von Einzelbildern auf und speichert sie auf einem Server. Weiter ist es möglich, Bereiche in Bildern zu maskieren, um Bewegungen darin zu ignorieren. Letzteres hilft vor allem dann, wenn sich beispielsweise Bäume oder Sträucher im Erfassungsbereich befinden, die im Wind schaukeln – oder eben die Katze, die ihr Revier kontrolliert.

### Exkurs: Bewegung erkennen

Eine sehr einfache Methode Bewegungen zu erkennen ist, zwei zeitlich aufeinander folgende Bilder zu vergleichen, um die Änderung von Pixeln zu erkennen. Überschreitet die Zahl der Änderungen einen Schwellenwert, wird beispielsweise ein Alarm ausgelöst. Bei statischen Motiven und konstanter Beleuchtung genügt eventuell auch der Vergleich mit einem Referenzbild. Wenn man in diesem Fall das aktuelle Bild vom Referenzbild subtrahiert, liefern alle Pixel, die sich nicht verändern haben, das Ergebnis Null zurück (einen schwarzen Pixel). Dieses Verfahren funktioniert ganz gut, jedoch stößt es bei Beleuchtungsänderung und bei kleinen Veränderungen im Bild an seine Grenzen.

Hebt sich das bewegende Objekt gut vom Hintergrund ab, etwa mit einer weißen Wand dahinter, ist auch die langsamste Bewegung detektierbar. Variiert der Hintergrund stark und erfolgt die Bewegung langsam, kann man die Bewegungserkennung austricksen. Bei Außenaufnahmen kommen noch Effekte wie vom Wind bewegte Äste oder die Katze vom Nachbarn hinzu. Der Stubentiger soll ja im Gegensatz zu einem Einbrecher keinen Alarm auslösen. In solchen Fällen ist es günstig, nur einen Teilausschnitt des Bildes zu untersuchen.

Das Prinzip des Differenz-Algorithmus zeigt das folgende Programmfragment: Die Funktion *changed* liefert die Anzahl der Pixel zurück, die sich jeweils in beiden Bildern um mehr als den Wert *diff* unterscheiden.

`t_img` ist wieder Struktur, die alle Bildinfo enthält. Den Zähler für die Unterschiede setzt Zeile 06 initial auf Null. Die For-Schleifen der Zeilen 08 und 10 iterieren den Pixelvergleich über alle Pixel x und y. Der Pixelvergleich startet mit dem Einlesen aller aktuellen Pixel zweier Bilder (Zeilen 12 und 13). Jetzt gilt es, den Grauwert jedes Bildes zu ermitteln (Zeilen 15 und 16). Unterscheiden sich die Werte, soll der Zähler inkrementieren (Zeile 18 bis 19).

```
01 int changed(t_img *bild1, t_img *bild2, int diff)
02   {
03   int y, z, chan, diffcount;
04   t_color val1, val2;
05
06   diffcount = 0;
07
08   for(x = 0; x < bild1->xsize; x++)
09     {
10     for(y = 0; y < bild1->ysize; y++)
11       {
12       getpixel(bild1, x, y, &val1);
13       getpixel(bild2, x, y, &val2);
14
15       grau1 = (val1.R + val1.G + val1.B)/3;
16       grau2 = (val2.R + val2.G + val2.B)/3;
17
18       if(abs(grau1 - grau2) > diff)
19         diffcount++;
20       }
21     }
22   return diffcount;
23   }
```

Vergessen Sie auch nicht, dass sich die Bilder mit den ImageMagick-Tools *convert* und *mogrify* vor- und nachbereiten lassen. Neben diversen Filterfunktionen lassen sich auch Ober- und Untergrenzen für die Helligkeit angeben, bei deren Über- bzw. Unterschreitung die Pixel weiß bzw. schwarz eingefärbt werden.

Neben der Funktion als Bewegungsmelder eignet sich *motion* auch, um innerhalb bestimmter Intervalle Schnappschüsse zu speichern oder fortlaufend Videos aufzunehmen. Das Werkzeug verfügt zudem über eine Möglichkeit, die aktuell empfangenen Bilder von überall mit einem Browser abzurufen. Bei der Installation von *motion* wird auch eine Gruppe gleichen Namens erzeugt. Alle Nutzer, die *motion* verwenden, müssen zu dieser Gruppe hinzugefügt werden, damit die Konfigurationsdatei von *motion* gelesen werden kann.

In der Datei `/etc/motion/motion.conf` lassen sich zahlreiche Einstellungen vornehmen. Die wichtigsten für erste Versuche zeigt Tabelle 4.

Nach dem Anpassen der Konfigurationsdatei können Sie diese durch den Aufruf `motion -n` schon mal testen. Funktioniert alles nach Wunsch, starten Sie `motion` als Daemon. Bewegt sich etwas vor der Kamera, sollten nun Bilder aufgenommen und gespeichert werden. Später kann der Daemon

in die Startskripte aufgenommen werden. Sollten zuviele Bilder ohne erkennbare Bewegung aufgenommen werden, können Sie die Empfindlichkeit über den *threshold*-Wert einstellen.

| *motion*-Einstellung | Beschreibung |
|---|---|
| *videodevice* | Gerätedatei der Webcam. |
| *width* | Breite der aufzunehmenden Bilder in Pixel. |
| *height* | Höhe der aufzunehmenden Bilder in Pixel. |
| *framerate* | Anzahl aufzunehmende Bilder pro Sekunde. Alternative: *minimum_frame_time* |
| *minimum_frame_time* | Wartezeit (s) zwischen zwei Bildern. Alternative: *framerate* |
| *threshold* | Anzahl der Pixel, die sich ändern müssen, damit `motion` auslöst. Voreinstellung: 1500. |
| *quality* | Qualität der aufzunehmenden Bilder. |
| *ffmpeg_video_codec* | Dateityp des aus den Bildern zu generierenden Videos. |
| *target_dir* | Speicherort der Bilder und Videos. Falls keine Daten gespeichert werden sollen, hilft der allseits beliebte Papierkorb `/dev/null`. |
| *start_motion_daemon* | Für den Daemon-Betrieb auf `YES` zu setzen. |

Tabelle 4: Wichtigste Einstellungen für das ImageMagick-Tool *motion* in der Konfigurationsdatei `/etc/motion/motion.conf`.

Falls das nicht erfolgreich ist, lässt sich der Bildausschnitt maskieren. Dazu erzeugen Sie mit dem Grafikprogramm Ihrer Wahl ein Bild von der Größe des Kamerabildes laut Angabe von *width* und *height*. Das Bild hat nur schwarze und weiße Flächen. Alles was schwarz ist, wird ausgeblendet. Im einfachsten Fall wäre dies also ein einfacher schwarzer Rahmen. Die Datei wird dann im Format *pgm* (Portable Gray Map) gespeichert. In der Konfigurationsdatei tragen Sie in diesem Fall eine weitere Zeile ein:

`mask_file` *`/pfad/zur/maskendatei`*`.pgm`

Ein weiteres Feature von `motion` ist, dass man auf bestimmte Ereignisse reagieren kann. Mit der Konfigurationsoption *on_event_start* wird ein Programm oder Skript angegeben, das `motion` ausführt, sobald eine Bewegung erkannt wurde. Auf diese Weise kann man sich dann Nachricht per SMS, Twitter oder E-Mail zusenden lassen. Für erste Tests können Sie beispielsweise einen Zeitstempel an eine Datei anhängen:

```
on_event_start 'date "+%d%m%Y%H%M" \
   >> /home/testuser/event'
```

Als Alternative zu *motion* würde sich noch *zoneminder* anbieten, das aus etlichen Modulen besteht und über ein Web-Interface bedient wird ([15],[16]). Neben Video4Linux für die Unterstützung der Kameras werden ein Apache-Webserver, MySQL, PHP und Perl benötigt. Für die Aufnahme von Stand- und Bewegtbildern kommen noch die die Ffmpeg- und Libjpeg-Pakete hinzu. Der Aufwand gegenüber Motion ist also beträchtlich. Um die wegen der vielen Pakete und Codecs etwas umständliche Installation und Konfiguration zu erleichtern, hält das Zoneminder-Forum ein Skript bereit [17].

## Nachtblind? Infrarot-Scheinwerfer löten!

Ist die Kamera nachtblind? Dann hilft ein kleiner Infrarot-Scheinwerfer. Manche Kameras, etwa die des Herstellers Trendnet, haben sogar schon eine kleine Infrarot-Beleuchtung eingebaut. Bei der Trendnet-Kamera sorgt eine Photozelle dafür, das die sechs Infrarot-LEDs nur bei Dunkelheit mit Strom versorgt werden. Aber nicht alle elektronischen Kameras sind für infrarotes Licht empfindlich, manche haben einen Infrarotfilter in der Optik. Da hilft dann nur Ausprobieren.

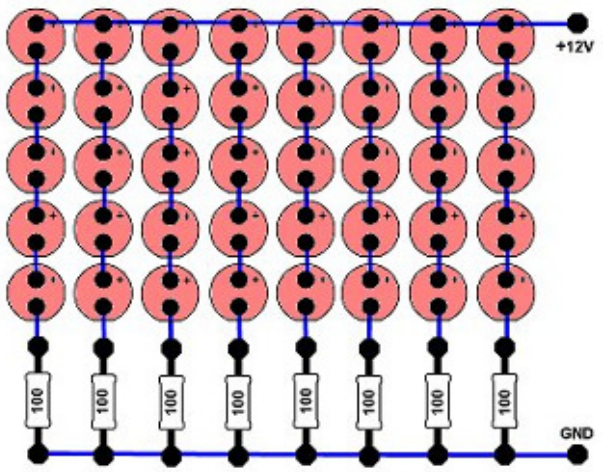



Abbildung 5: Die Verdrahtung des Infrarot-Scheinwerfers.

Der hier vorgestellte Eigenbau-Infrarot-Scheinwerfer besteht aus 40 Infrarot-LEDs und acht Widerständen. Er lässt sich auch vom Lötanfänger ganz einfach auf einer Lochrasterplatte aufbauen. Profis machen sich natürlich eine Platine, insbesondere wenn mehr als ein Scheinwerfer gebraucht wird. Bei 12 V Versorgungsspannung nimmt der Scheinwerfer ca. 200 mA auf, er sollte also ggf. vom Computer aus ein- und ausgeschaltet werden können (per Relais oder MOSFET-Schalttransistor). Die Schaltung des Scheinwerfers

ist so einfach, das Bild 5 nur die Verdrahtung zeigt. Es werden jeweils fünf LEDs und ein 100-Ohm-Widerstand in Reihe geschaltet. Für den Außeneinsatz muss dann noch ein passendes Gehäuse mit Klarsichtdeckel spendiert werden.

Kameras, die im Freien hängen, müssen Temperaturschwankungen, Regen, Sturm und Schnee widerstehen. Die Gehäuse sollten wasserdicht sein, sonst ziehen sie Feuchtigkeit, es beschlagen die Linsen oder die Elektronik im Inneren korrodiert. Entsprechend geschützte Modelle stecken in Metallgehäusen, ihre Linsen sind verdeckt von abgedichteten Glasscheiben. Diese Netzwerkkameras, beispielsweise von *Mobotix*, kosten um die 500 Euro, und wetterfeste Kamera-Gehäuse kosten zwischen 80 und 300 Euro. Dafür sind sie von innen beheizt (230 V) und können so weder einfrieren noch beschlagen.

Die Alternative ist ein (unbeheiztes) Low-Cost-Gehäuse für die USB-Webcam. Hier dient als Wetterschutzgehäuse ein 100-W-Halogenstrahler, erleichtert um sein Innenleben (Abbildung 6. Das Gehäuse bietet genügend Platz für die Kamera. Für größere Kameras muss man sich dann das jeweils passende Gehäuse suchen (150- oder 500-Watt-Lampe).

Abbildung 6: Ein Halogenstahler ohne Innenleben liefert ein preiswertes Wetterschutzgehäuse.

Bei der Logilink UA0155 wurde der hintere Teil der Halterung abgesägt, wobei der Teil mit dem Kugelgelenk unversehrt blieb. Dann kann man die Kamera einfach ins Gehäuse kleben, wobei die Möglichkeit, die Kamera über das Kugelgelenk auszurichten, erhalten bleibt. Um den USB-Stecker aus dem Gehäuse herausführen zu können, musste der kleine Anschlusskasten entfernt und die Kabeldurchführung mit einer Rundfeile etwas vergrößert werden. Anschließend wird die Öffnung wieder verschlossen und mit Silikon abgedichtet.

Gegebenenfalls kann das Gehäuse mit einem Heizwiderstand ausgerüstet werden, der im Winter das Vereisen der Glasscheibe verhindert. Dafür sollte aber immer eine getrennte Stromversorgung verwendet werden. Einerseits muss ja nur bei Minustemperaturen geheizt werden und andererseits liefert die USB-Schnittstelle nicht wirklich genügend Strom zum Heizen.

## Letzter Streich: Storage optimieren

Nachdem der Kamera-Server nicht viel zu tun hat, bietet es sich an, als Thinclient einen Barbone oder Raspberry Pi zu verwenden. Damit es keine beweglichen Teile gibt (Verschleiß) kann eine Solid State Disk (SSD), eine interne Flash-Speicherkarte oder eine SD-Karte als Massenspeicher dienen. Das Logging lässt sich abschalten (entweder ganz oder durch Umleitung nach */dev/null*).

Dabei ergibt sich allerdings ein schwerwiegender Nachteil: Die interne Flash-Speicherkarte, SD-Karte oder SSD des Rechners wird belastet, da die Bilder erst auf der Platte zwischengespeichert werden, bevor sie auf den Webserver oder einen anderen fernen Server übertragen werden. Mit der oben geschilderten Helligkeitsanpassung eventuell sogar zweimal. Das ständige Löschen und Wiederbeschreiben ist trotz aller schlauer Algorithmen der Controller-Bausteine Gift für die Lebensdauer der Solid-State-Speicher.

Deshalb sollte für alle Dateien eine RAM-Disk eingerichtet werden – bei der sogar ein noch schnellerer Zugriff als Nebeneffekt hinzukommt. Dazu gibt es zwei Möglichkeiten. Das ***tmpfs-Dateisystem*** als erste Möglichkeit ist eigentlich kein reines RAM-Dateisystem, sondern die Daten landen im Festplatten-Swap, sobald der Speicherplatz im RAM knapp wird. Folgender Shell-Befehl macht das Verzeichnis */root/tmp* zur RAM-Disk:

```
# mount -t tmpfs none /root/tmp
```

Falls die Größe festgelegt werden soll, hilft:

```
# mount -t tmpfs -o size=20M none /root/tmp
```

Es werden jedoch dynamisch immer so viele Ressourcen abgezweigt, wie gerade benötigt werden, auch wenn eine Größe angegeben wurde. Ist das Laufwerk also leer, belegt es auch keinen Platz im RAM. Es ist möglich, diese Partition standardmäßig beim Systemstart einzubinden, indem man eine Zeile in die Datei `/etc/fstab` einfügt:

```
tmpfs /root/tmp tmpfs defaults,size=20M 0 0
```

Das ***ramfs***-*Dateisystem* als zweite Möglichkeit lagert im Gegensatz zum *tmpfs* keine Daten in den Swap aus, ist also ein reines RAM-Dateisystem. Das Einrichtungskommandos ist fast identisch:

```
sudo mount -t ramfs ramfs /root/tmp
```

Damit erhält man eine RAM-Disk, die sich ebenfalls dynamisch der benötigten Größe anpasst. Um die Partition beim Systemstart automatisch einzuhängen, fügt folgende Zeile in der Datei `/etc/fstab` hinzu:

```
ramfs /root/tmp ramfs defaults 0 0
```

Das *ramfs*-Dateisystem erlaubt im Gegensatz zu *tmpfs* keine Mountoptionen und bietet somit auch keine Möglichkeit, die Größe zu limitieren. Eventuell hat das System dann keinen freien Hauptspeicher mehr zur Verfügung und kann nur noch auf die Festplatte auslagern.

## Finale: Ab auf die Hütte

Noch im Herbst letzten Jahres sind wir auf die Hütte hoch – nicht nur mit Kameras und IPC, sondern zusätzlich mit Kabel, Monitor, Tastatur und natürlich einer g'scheiten Brotzeit. Der preiswert gebraucht erworbene 12-Zoll-Monitor gehörte wohl mal zu einem Kassenterminal und läuft mit den 12 V aus der Solaranlage.

Oben angekommen: Super Wetter, alles bestens! Na ja, fast. Am geplanten Rechnerstandort im Speicher hatten sich Wespen häuslich eingerichtet – keine Chance, irgendetwas einzubauen.

Aber ganz ohne Erfolg wollten wir doch nicht wieder gehen. Also wurde ein Tisch vor die Hütte gestellt und mal ein Probeaufbau in Betrieb genommen. So konnten wir schon mal sehen, ob der Surfstick ins Netz kommt und wie die Kamerabilder so aussehen. Die Internet-Verbindung klappte, jedoch zeigte es sich, dass der Stick wohl mehrere Talstationen sah, weshalb er sich nicht gleich entscheiden konnte, mit welcher er nun Kontakt aufnimmt. Ihm half, in der Software den Timeout kräftig hoch zu setzen. Danach lief alles wie geplant. Und natürlich mussten wir jedem vorbeikommenden Wanderer erklären, dass wir keinen Lauschposten für die NSA installieren.

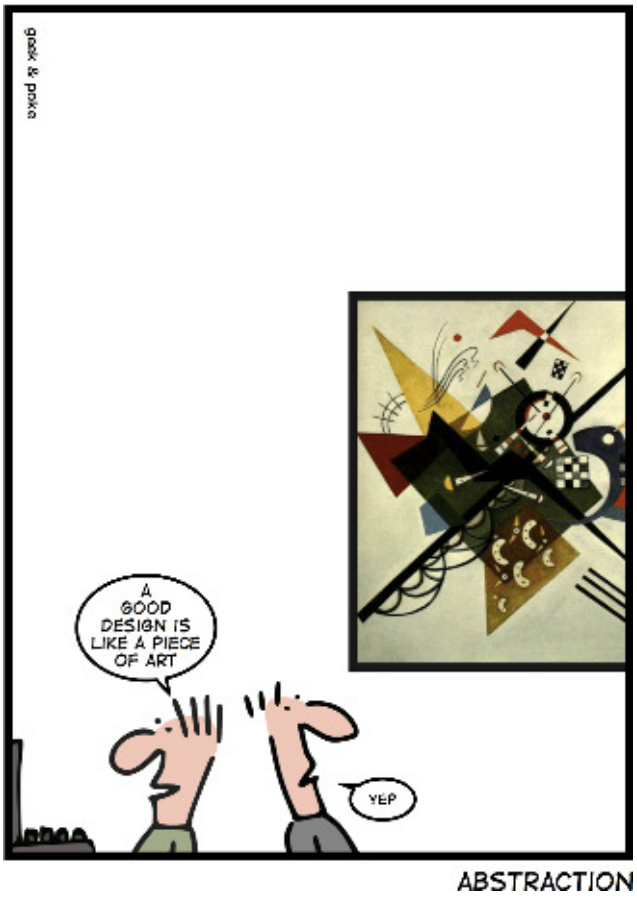
Der Festeinbau findet in diesem Frühsommer 2014 statt, wenn der Schnee geschmolzen ist und die Fluglöcher der Wespen verstopft sind. Mir kam gerade noch eine Idee: Wenn der Industrie-PC schon am Internet hängt, könnte ich eigentlich noch ein Kabel mit Netzwerkdose runter in die Hütte ziehen und auf dem PC Routing aktivieren – für den Nerd-Faktor in der Hütte.

**Weitere Infos**

[1] Fswebcam: http://www.firestorm.cx/fswebcam/
[2] Bildbetrachter *gpicview* für X: http://lxde.sourceforge.net/gpicview/
[3] Bildbetrachter *fbi* für die Konsole: http://www.kraxel.org/blog/linux/fbida/
[4] Logilink UA0155: http://www.logilink.eu/showproduct/UA0155.htm
[5] HAMA Digital Eye II: https://de.hama.com/00053953/hama-hd-webcam-digital-eye-ii
[6] Logitech C270 USB Webcam: http://www.logitech.com/de-de/product/hd-webcam-c270
[7] Microsoft LifeCam HD-3000: http://www.microsoft.com/hardware/de-de/p/lifecam-hd-3000
[8] Axis 207: http://www.axis.com/products/
[9] Trendnet TV-IP100: http://www.trendnet.com/products/
[10] Logilink WC0040: http://www.logilink.eu/showproduct/WC0040.htm
[11] Grandtec Megapixel IP Camera: http://www.grand.com.tw/sur_mega_pixel.php
[12] ImageMagick: http://www.imagemagick.org
[13] Homepage von *motion*: http://www.lavrsen.dk/foswiki/bin/view/Motion
[14] Motion-Artikel im LinuxUser 12/2010:
http://www.linux-community.de/Internal/Artikel/Print-Artikel/LinuxUser/2010/12/Objekte-mit-Motion-per-Video-ueberwachen
[15] Zoneminder-Homepage: http://www.zoneminder.com
[16] Zoneminder Artikel in LinuxUser 09/2011:
http://www.linux-community.de/Internal/Artikel/Print-Artikel/LinuxUser/2011/09/Videoueberwachung-mit-Zoneminder/
[17] Zoneminder-Installscript: http://www.zoneminder.com/forums/viewtopic.php?t=16628

## Über Jürgen

Jürgen Plate ist Professor für Elektro- und Informationstechnik an der Hochschule München. Er beschäftigt sich seit 1980 mit Datenfernübertragung und war, bevor der Internetanschluss für Privatpersonen möglich wurde, in der Mailboxszene aktiv. Unter anderem hat er eine der ersten öffentlichen Mailboxen – TEDAS der mc-Redaktion – programmiert und 1984 in Betrieb genommen.



## Gewusst, wie!
## Das IT-Weiterbildungssystem

Seit 1997 existieren die neu geordneten IT-Ausbildungsberufe. Im Jahr 2002 kam die Verordnung über Fortbildungen im ITK-Bereich hinzu. Der Artikel zeigt, wie die Ausbildungen im deutschen dualen Berufsbildungssystem implementiert sind und welche Qualifizierungsmöglichkeiten für IT-Facharbeiter bestehen.

von **Stefan Schumacher**

Es gibt nur eins,
was auf Dauer teurer ist als Bildung:
keine Bildung.

John F. Kennedy

Waren die schrankwandgroßen Computer vor einigen Jahrzehnten noch ausschließlich von entsprechend ausgebildeten Elektrotechnikern oder Facharbeitern für elektronische Datenverarbeitung zu bedienen, haben heute nahezu alle Unternehmen Informationstechnik eingeführt: Sei es, um die elektronische Steuererklärung durchzuführen, eine eigene Webseite zu betreiben oder per E-Mail mit Kunden in Kontakt zu bleiben.

Üblicherweise übernehmen Informatiker oder Hochschulabsolventen aus benachbarter MINT-Gebieten die Systemadministration. Kleinere und mittlere Unternehmen (KMU) sind jedoch oft nicht in der Lage, studierte Informatiker zu vergüten. Außerdem erscheint es unwahrscheinlich, dass ein Handwerksbetrieb mit einem Meister an der Spitze einen Diplom-Informatiker einstellt. Um auch die Ausbildung auf Facharbeiter-Niveau zu ermöglichen, regelt die *Verordnung über die Berufsausbildung im Bereich der Informations- und Telekommunikationstechnik* seit 1997 die Ausbildungsberufe Fachinformatiker, Informatikkaufmann, IT-Systemkaufmann und IT-Systemelektroniker [1].

### Ausbildung und Rahmenlehrplan für Fachinformatiker

Das deutsche Berufsbildungssystem ist mit der dualen Berufsausbildung und den darauf aufbauenden Weiterbildungen zum Meister, Fachwirt oder Techniker stark formalisiert. Ordnungspolitisch erreichen dies das Berufsbildungsgesetz (BBiG), das Gesetz zur Ordnung des Handwerks (kurz: Handwerksordnung, HwO) sowie die darauf aufbauenden Ausbildungsordnungen (AO). Einhergehend mit der starken Regularisierung haben formale Zertifikate („Abschlüsse“) hohe Bedeutung: Beispielsweise gibt es in einigen Gewerken noch den *Großen Befähigungsnachweis* (auch Meisterzwang genannt).

Im Jahre 1997 wurde also der Ausbildungsberuf des Fachinformatikers eingeführt. Die Zahl der neu geschlossenen Ausbildungsverträge stieg laut Berufsbildungsbericht 2013 [2] von 1.779 im Jahr 1997 auf 9.480 im Jahre 2000 und pendelt seither um die 9.000 herum. 2011 war dieser Ausbildungsberuf mit 9.843 neugeschlossenen Ausbildungsverträgen der beliebteste von allen seit 1996 neu geschaffenen Ausbildungsberufen, noch vor dem Mechatroniker, der mit 7.653 Neuabschlüssen an zweiter Stelle folgt.

Von den 2011 neuen rund 10.000 Auszubildenden zum Fachinformatiker verfügten knapp 6.000 über eine Studienberechtigung, also rund 60 Prozent. Das entspricht 4,4 Prozent sämtlicher Azubis mit Studienberechtigung. Von den zehn Ausbildungsberufen mit der höchsten Zahl an studienberechtigten Auszubildenden ist Fachinformatiker der einzige technische Beruf. Die anderen neun stammen alle aus dem kaufmännischen Bereich.

Die Ausbildung erfolgt an den Lernorten Berufsschule und Ausbildungsbetrieb, die zuständige Stelle ist die Industrie- und Handelskammer (IHK) des jeweiligen Ausbildungsbetriebes. Sie nimmt über ihren Prüfungsausschuss auch die Abschlussprüfung vor und verleiht bei Bestehen den Abschluss als Fachinformatiker. Der Ausbildungsbetrieb vermittelt während der Ausbildung vorwiegend praktische Fertigkeiten, während die Berufsschule vorwiegend theoretische Kenntnisse vermitteln soll. Der Ausbildungsbetrieb hat daher die Möglichkeit, große Teile des Ausbildungsinhaltes im Rahmen der Ausbildungsordnung festzulegen und auf die eigenen Handlungsprozesse abzustimmen.

Der Rahmenlehrplan für Fachinformatiker definiert elf Lernfelder mit insgesamt 880 Unterrichts-

stunden. Es existieren die zwei Fachrichtungen Anwendungsentwicklung und Systemintegration, die sich nur im Zeitumfang einiger Lernfelder unterscheiden. Tabelle 1 zeigt die Lernfelder.

| | |
|---|---|
| 1 | Der Betrieb und sein Umfeld |
| 2 | Geschäftsprozesse und betriebliche Organisation |
| 3 | Informationsquellen und Arbeitsmethoden |
| 4 | Einfache IT-Systeme |
| 5 | Fachliches Englisch |
| 6 | Entwickeln und Bereitstellen von Anwendungssystemen |
| 7 | Vernetzte IT-Systeme |
| 8 | Markt und Kundenbeziehungen |
| 9 | Öffentliche Netze und Dienste |
| 10 | Betreuung von IT-Systemen |
| 11 | Rechnungswesen und Controlling |

Tabelle 1: Die elf Lernfelder im Ausbildungsberuf Fachinformatiker

Der Rahmenlehrplan gibt in den Lernfeldern die Zielformulierung vor. Die Lernfelder selbst entstehen aus den beruflichen Handlungsfeldern. Dabei handelt es sich um berufliche Aufgabenstellungen, die ein Facharbeiter beherrschen soll. Diese sind in der Regel prozess- oder auftragsorientiert entwickelt. Ein typisches Handlungsfeld für einen Fachinformatiker wäre beispielsweise die Datensicherung in vernetzten Systemen.

Der Lehrer entwickelt in den Lernfeldern konkrete Lernarrangements in sogenannten Lernsituationen. Lernsituationen sind die kleinsten Lehrplaneinheiten der Berufsbildung. Reinhard Bader führt in „Lernfelder konstruieren – Lernsituationen entwickeln“ konkrete Schritte auf, um vom beruflichen Handlungsfeld zur Lernsituation zu kommen [3]. Da die Lernfelder im Rahmenlehrplan bereits vorgegeben sind, ist nur noch der Schritt vom Lernfeld zur Lernsituation zu vollziehen.

Für IT-Sicherheit beispielsweise existiert aber gar kein Lernfeld. Das Thema ist in den Lernfeldern 4, 7, 9 und 10 integriert. Da die Lernfelder keine Angaben zu den Lehrinhalten machen (etwa: Verschlüsselung von Dateien mit einem bestimmten Algorithmus), bleibt es den Berufsschullehrern überlassen, notwendige Handlungskompetenzen auszuwählen und zu vermitteln. Dies ermöglicht zwar die im dualen Berufsbildungssystem notwendige Flexibilität und Anpassung an die jeweiligen Ausbildungsbetriebe. Das verhindert aber auch eine einheitliche Grundausbildung mit IT-Sicherheitsthemen für jeden auszubildenden Fachinformatiker, wie diese beispielsweise für die Elektrofachkräfte existiert.

Da die Ausbildung zum Fachinformatiker im dualen System durchgeführt wird, verbringt der Auszubildende seine Lehrzeit nicht nur in der berufsbildenden Schule, sondern auch im Ausbildungsbetrieb. Dieser bringt dem Auszubildenden die für notwendig erachteten Fertigkeiten und Kenntnissen bei, sofern diese eine notwendige berufliche Handlungsfähigkeit nach dem Berufsbildungsgesetz darstellen [4]. Es steht dem Ausbildungsbetrieb damit frei, Handlungsfelder der IT-Sicherheit zu vermitteln und selbst zu überprüfen. Über das qualifizierte Zeugnis des Ausbildungsbetriebes erfolgt aber keine formale und anerkannte Zertifizierung.

## Das IT-Weiterbildungssystem

Im Jahr 2002 wurde die IT-Fortbildungsverordnung [5] erlassen, welche das IT-Weiterbildungssystem einführte. Dafür bildet es die aus dem Handwerk bekannten Hierarchieebenen der Aufstiegsfortbildung ab: Es besteht aus drei aufeinander aufbauenden Qualifizierungsebenen, wie Abbildung 1 zeigt.

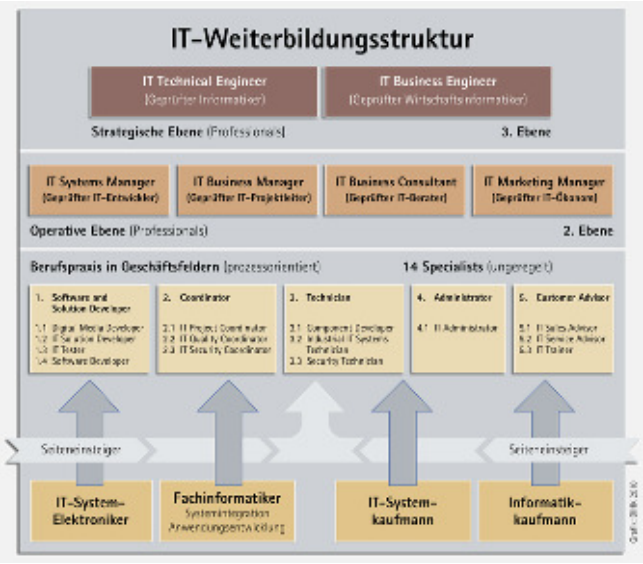



Abbildung 1: Das IT-Weiterbildungssystem nach dem Deutschen Industrie- und Handelskammertag (DIHK).

Auf der untersten Ebene befinden sich die Facharbeiter der Informations- und Telekommunikationstechnik, zum Beispiel Fachinformatiker, IT-Kaufleute oder auch Quereinsteiger. Deren erster Qualifikationsschritt erfolgt im Arbeitsprozess, also berufsbegleitend, und orientiert sich an den Arbeitsprozessen der Praxis. Das Ergebnis besteht auf Ebene 1 aus sogenannten **Specialists**. Ebene 2 zertifiziert dann **operative Professionals** und Ebene 3 schließlich die **strategischen Professionals**.

Als Zulassungsvoraussetzung zur Fortbildungsebene 1 – **Specialist** – zählen laut dem IT-Sektorkommitee, das damals zur Einführung von Personalzertifizierungen im Rahmen des IT-Weiterbildungssystems gegründet wurde [6]:

- eine berufsqualifizierender Bildungsabschluss in einem Beruf des IT-Bereichs, oder ein Bachelor- oder Master-Abschluss aus dem IT-Bereich;
- ersatzweise einen berufsqualifizierenden Bildungsabschluss in einem sonstigen Beruf und danach eine mindestens einjährige Berufspraxis im IT-Bereich,
- oder ersatzweise eine mindestens vierjährige Berufspraxis im IT-Bereich;
- oder ersatzweise durch Zeugnisse oder auf andere Weise glaubhaft gemachte Qualifikationen, die die Zulassung zur Zertifizierung rechtfertigen.

Die zu zertifizierende Person kann sich in einem der folgenden 29 Spezialistenprofile zertifizieren lassen:

- Softwareentwickler (Software Developer)
  - IT Systems Analyst
  - IT Systems Developer
  - Software Developer
  - Database Developer
  - User Interface Developer
  - Multimedia Developer
- Techniker (Technician)
  - Component Technician
  - Industrial IT Systems Technician
  - Security Technician
- Lösungsbetreuer (Administrator)
  - Network Administrator
  - IT Systems Administrator
  - Database Administrator
  - Web Administrator
  - Business Systems Administrator
- Produkt- und Kundenbetreuer (Advisor)
  - Service Advisor
  - IT Trainer
  - IT Product Coordinator
  - IT Sales Advisor
- Entwicklungsbetreuer (Coordinator)
  - IT Project Coordinator
  - IT Configuration Coordinator
  - IT Quality Management Coordinator
  - IT Test Coordinator
  - IT Technical Writer
- Lösungsentwickler (Solutions Developer)
  - Business Systems Advisor
  - E-Marketing Developer
  - E-Logistic Developer
  - Knowledge Management Systems Developer
  - IT Security Coordinator
  - Network Developer

Mit dem Antrag zur Zertifizierung legt der Prüfling eine Projektskizze für ein zu bearbeitendes Projekt vor. Das Projekt soll dem betrieblichen Arbeitszusammenhang entsprechen, in das Profil des Prüflings passen und von der zertifizierenden Stelle als fachlich geeignet eingestuft werden. Die Zertifizierungsstellen sind privatwirtschaftlich organisiert und müssen bei der Deutschen Akkreditierungsstelle (DAkkS) überprüft worden sein, wofür sie bestimmte DIN-Normen einhalten müssen.

Für die Betreuung des Prüflings ist mindestens ein Fachberater und ein Lernprozessbegleiter zu benennen. Der Fachberater sollte aus dem betrieblichen Umfeld stammen (Kollege, Vorgesetzter), der Lernprozessbegleiter kann sowohl aus dem Umfeld stammen, oder als externer Berater engagiert werden. Der Lernprozessbegleiter führt mindestens vier mal jährlich ein Reflexionsgespräch mit dem Prüfling durch. Dieser soll insbesondere die eigene Kompetenzentwicklung reflektieren und in der Dokumentation beschreiben. Nach Abgabe der Dokumentation wird diese durch die zertifizierende Stelle begutachtet und der Prüfling zu einer Präsentation seines Projektes sowie einem Fachgespräch geladen. Absolviert er die Aufgabenstellung erfolgreich, erhält er sein unbenotetes Zertifikat als **IT-Specialist** im jeweiligen Profil.

Die Zertifizierung endet hiermit aber nicht: Der frisch gebackene Spezialist hat nun sicherzustellen, dass er weiterhin qualifiziert bleibt. Dazu hat er 18 bis 24 Monate nach der Zertifizierung berufliche Tätigkeiten, Projekte und Weiterbildungen im Profil durch Projektskizzen nachzuweisen. Nach fünf Jahren erfolgt eine Re-Zertifizierung durch ein 60-minütiges Fachgespräch über erneut eingereichte Projektblätter.

Ebene 2 zertifiziert die **operativen Professionals** in den vier Profilen *Geprüfter Entwickler*, *Geprüfter IT-Projektleiter*, *Geprüfter IT-Berater* und *Geprüfter IT-Ökonom*. Ebene 3 zertifiziert **strategische**

**Professionals** in den zwei Profilen *Geprüfter Informatiker* und *Geprüfter Wirtschaftsinformatiker*. Zugelassen zur Prüfung wird, wer die jeweils darunter liegende Qualifikationsstufe erreicht hat.

Die Qualifizierungen erfolgen wieder berufsbegleitend im Arbeitsprozess. Die Zertifizierung selbst ist aber staatlich geregelt und wird von den Industrie- und Handelskammern abgenommen. Dabei müssen operative Professionals ein betriebliches IT-Projekt durchführen und dokumentieren sowie Kenntnisse aus dem Bereich Mitarbeiterführung und Personalmanagement nachweisen. Strategische Professionals bearbeiten einen strategischen IT-Prozess als Fallstudie sowie Projekt- und Geschäftsbeziehungen im schriftlichen Test.

## Zertifizierungen außerhalb des Berufsbildungssystems

Auch außerhalb des formalen Berufsbildungssystems bestehen Weiterbildungsmöglichkeiten, die in der Regel der Anbieter selbst zertifiziert. Diese Zertifikate sind zwar nicht staatlich anerkannt und enthalten damit keinen formalen Tauschwert etwa als Zugangsberechtigung zu Hochschulen. Sie erhöhen aber in der Regel die Chancen bei der Arbeitsplatzsuche und zertifizieren produkt- oder herstellerabhängige Fachkenntnisse.

Diese Zertifizierungen haben ihren Ursprung in den USA, wo das IT-Unternehmen Novell 1989 den *CNE – Certified Netware Engineer* einführte. Der Hintergrund für die Einführung einer derartigen Zertifizierung liegt im amerikanischen Berufsbildungssystem. Da es keine starke Formalisierung und in der Regel auch keine unabhängige und vergleichbare Prüfung von Ausbildungsinhalten gibt, sollte dieses Zertifikat einheitlich sicherstellen, dass der Prüfling die vom Hersteller Novell ausgewählten Prüfungsinhalte beherrscht. Damit wurde eine Vergleichbarkeit von Fachkräften geschaffen, die im deutschen Berufsbildungssystem allerdings schon durch die Abschlussprüfungen der Ausbildungen gewährleistet ist.

Problematisch ist bei diesen Zertifizierungen der Fokus auf einen bestimmten Hersteller oder ein Produkt. Ein Microsoft- oder Apple-Zertifikat ist eben nur für Microsoft- oder Apple-Produkte gedacht und soll keine allgemeine berufliche Handlungsfähigkeit vermitteln, wie das die deutsche Berufsausbildung tut.

Daraus ergibt sich ein weiteres Problem. In den USA sind standardisierte Tests wie die Hochschulzugangstests SAT oder ACT [7] weit verbreitet und anerkannt, was sich auch auf die Akzeptanz weiterer standardisierter Tests auswirkt. Die erste Generation der Microsoft-Zertifikat bestand lediglich aus einem Katalog von Multiple-Choice-Fragen, von denen einige zufällig ausgewählt und abgefragt wurden. Der vollständige Fragenkatalog war als Buch verfügbar – mehrere Prüflinge lernten den Katalog auswendig. Sie bestanden damit die Prüfung und erhielten das Zertifikat, waren aber im beruflichen Umfeld nur bedingt handlungsfähig.

In Deutschland sind solche Zertifikate weniger stark verbreitet als in den USA. Sie werden auch nicht formal anerkannt, etwa bei der Beantragung einer EU-Blue-Card für Einwanderer, oder bei der Zulassung zu einem Studienplatz.

## Innerbetriebliche Fortbildungsmaßnahmen

In den letzten Jahren ist insbesondere mit der Diskussion um Kompetenzentwicklung und *Lebenslanges Lernen* das nicht-formale Lernen in den Fokus der Forschung geraten. Derartige Initiativen untersuchen, wie Lernerfolge aus nicht-formalen sowie informellen Kontexten anerkannt werden können.

Das Bundesministerium für Bildung und Forschung definiert nicht-formales Lernen als Lernen, welches außerhalb der Hauptsysteme der allgemeinen und beruflichen Bildung stattfindet und nicht zwingend zu einem formalen Abschluss führt [8]. Nicht-formales Lernen kann demnach am Arbeitsplatz oder in Organisationen der Zivilgesellschaft stattfinden. Davon unterscheidet es informelles Lernen: Dieses sei die

> natürliche Begleiterscheinung des täglichen Lebens. Anders als beim formalen und nicht-formalen Lernen handelt es sich beim informellen Lernen nicht notwendigerweise um ein intentionales Lernen, weshalb es auch von den Lernenden selbst unter Umständen gar nicht als Erweiterung ihres Wissens und ihrer Fähigkeiten wahrgenommen wird.

Im Rahmen von Weiterbildungsmaßnahmen können explizite und implizite Lernprozesse zertifiziert werden. So ist es in der Wirtschaft üblich, formale Lern-Settings wie Lehrgänge, Trainings oder Workshops zu schaffen und die Maßnahme durch ein Zertifikat zu bescheinigen – egal, ob die

Maßnahme didaktisch fundiert und ihr Erfolg evaluiert ist, oder ob der Dozent über irgendeine formale Qualifikation besitzt – oder überhaupt ein Konzept hat.

Daher ist die Anerkennung derartiger Zertifikate äußerst problematisch, wenn sie nicht durch eine Industrie- oder Handwerkskammer staatlich reguliert und damit geschützt sind, oder wenn sich nicht der Anbieter selbst einen entsprechenden Ruf über die Qualität seiner Zertifikate erworben hat. Anrechenbar auf eine reguläre Aufstiegsfortbildung sind derartige Zertifikate in der Regel nicht, es sei denn, die zertifizierende Stelle gestattet dies durch Einzelfallprüfungen.

Bei der Konzeption von innerbetrieblichen Weiterbildungen ist daher davon auszugehen, dass diese bei den regulären Aufstiegsfortbildungen im Rahmen des IT-Weiterbildungssystems nicht anerkannt werden. Eine Kooperation mit der zuständigen IHK im Vorfeld ist zwar möglich, allerdings nur schwer im gegebenen Rahmen umzusetzen. Die Zertifizierungen erfolgen grundsätzlich im Arbeitsprozess: Der zukünftige **Specialist** (Ebene 1) hat dafür ein Projekt zu bearbeiten und zu dokumentieren sowie in einem Fachgespräch das Ergebnis zu verteidigen. Da das Projekt im Arbeitsprozess durchgeführt und bearbeitet wird, wird es automatisch in die Prozesse des Unternehmens integriert. Eine explizite formale Schulung ist im IT-Weiterbildungssystem nicht vorgesehen und wird daher weder verlangt noch formal anerkannt.

Möchte ein Unternehmen Weiterbildungen durchführen, die eine formale Aufstiegsfortbildung darstellen, ist dies nur im Rahmen des IT-Weiterbildungssystems möglich. Die jeweiligen Zertifikate sind durch die bei der *DAkkS* akkreditierten zertifizierenden Stelle zu verleihen und werden nur so formal anerkannt. Es ist allerdings möglich, eigene Weiterbildungsmaßnahmen durchzuführen und das zu bearbeitende Projekt des Prüflings dort zu integrieren. Interne Zertifikate werden aber in der Regel nicht anerkannt.

## Durchlässigkeit des IT-Weiterbildungssystems

Die Durchlässigkeit des IT-Weiterbildungssystems ist sowohl in der horizontalen als auch vertikalen Richtung als hoch einzustufen. Die horizontale Durchlässigkeit bezeichnet die Anerkennung der formalen Qualifikation in den einzelnen Bundesländern sowie im Ausland. Da es sich bei den Zertifikaten des IT-Weiterbildungssystems um staatlich anerkannte Zertifikate und entsprechend regulierte und akkreditierte Prozesse handelt, sind die Zertifikate geschützt und werden bundesweit anerkannt. Es ist also egal, an welcher IHK man eine Zertifizierung beispielsweise zum **Operative Specialist** erworben hat: Jede andere IHK wird dieses Zertifikat bei der Zulassung zum **Strategic Specialist** anerkennen.

Die vertikale Durchlässigkeit bezeichnet die Übergänge zwischen den Qualifizierungsstufen Schule - Ausbildung - Hochschule. Sie ist im IT-Weiterbildungssystem formal geregelt und damit sowohl rechtssicher als auch hoch angesehen. Die drei definierten Ebenen bauen aufeinander auf und ermöglichen so den weiteren formalen Aufstieg. Den Zugang zur ersten Ebene (Specialist) ermöglicht entweder eine Berufsausbildung, oder die Berufserfahrung eines Quereinsteigers. Damit ist formal der Zugang auch für Menschen ohne abgeschlossene Berufsausbildung möglich.

Weiterhin ist auch der Wechsel an eine Hochschule möglich, da die Ebene 2 und 3 im Niveau 6 und 7 des *Europäischen Qualifikationsrahmens* [9] eingeordnet wurden. Allerdings ist der Zugang zu Hochschulen formal nicht einheitlich, sondern in den Hochschulgesetzen der Länder geregelt. Diese wiederum übertragen Teile der Zulassungskompetenz an die aufnehmende Hochschule, sodass sich die Zulassungsregeln von Land zu Land und von Hochschule zu Hochschule unterscheiden können.

Durch die Umsetzung des Europäischen Qualifikationsrahmens und Einordnung der Zertifizierungen in den *Deutschen Qualifikationsrahmen* ist die horizontale und vertikale Durchlässigkeit in der EU gewährleistet – sofern das Zielland den Europäischen Rahmen umsetzt.

**Links**

[1] Verordnung über die Berufsausbildung im Bereich der Informations- und Telekommunikationstechnik vom 10. Juli 1997 (BGBl. I S. 1741):
http://www.gesetze-im-internet.de/bundesrecht/itktausbv/gesamt.pdf
[2] Bundesinstitut für Berufsbildung: Datenreport zum Berufsbildungsbericht 2013; Informationen und

Analysen zur Entwicklung der beruflichen Bildung, Bonn 2013, S. 147:
http://datenreport.bibb.de/html/dr2013.html
[3] Reinhard Bader: Lernfelder konstruieren – Lernsituationen entwickeln; Eine Handreichung zur Erarbeitung didaktischer Jahresplanungen für die Berufsschule. In: Die Berufsbildende Schule, Ausgabe 7/8 2003, S. 210-217.
[4] § 14 Abs. 1 BBiG „Berufsausbildung“: http://www.gesetze-im-internet.de/bbig_2005/__14.html
[5] *Verordnung über die berufliche Fortbildung im Bereich der Informations- und Telekommunikationstechnik (IT-Fortbildungsverordnung)* vom 3. Mai 2002, zuletzt geändert durch die *Fünfte Verordnung zur Änderung von Fortbildungsprüfungsverordnungen* vom 26. März 2014 (BGBl. I S. 274):
http://www.gesetze-im-internet.de/it-fortbv/BJNR154700002.html
[6] IT-Sektorkomitee, Zertifizierung von IT- Spezialisten:
http://wis.ihk.de/fileadmin/inhalte/Dateien/IT-Weiterbildung/Normatives_Dokument_Version_2_mg.pdf
[7] Standardisierte Zugangstests SAT und ACT:
http://www.in-usa-studieren.de/usastudium/bewerbung_studienplatz_usa/standardisierte_zugangstests_usa.html
[8] Bundesministerium für Bildung und Forschung: Stand der Anerkennung non-formalen und informellen Lernens in Deutschland, im Rahmen der OECD Aktivität *Recognition of non-formal and informal Learning*, 2008:
http://www.bmbf.de/pub/non-formales_u_informelles_lernen_ind_deutschland.pdf
[9] Europäischer Qualifikationsrahmen (EQR): http://www.eu-bildungspolitik.de/eqr__33.html
[10] Deutscher Qualifikationsrahmen (DQR): http://www.deutscherqualifikationsrahmen.de

**Über Stefan**

Stefan Schumacher ist geschäftsführender Direktor des Magdeburger Instituts für Sicherheitsforschung und gibt das Magdeburger Journal zur Sicherheitsforschung heraus. Er befasst sich seit knapp 20 Jahren als Hacker mit Fragen der Informations- und Unternehmenssicherheit und erforscht Sicherheitsfragen aus pädagogisch-psychologischer Sicht. Seine Forschungsergebnisse stellt er auf Fachkongressen und in Publikationen vor. Seine Schwerpunkte liegen auf Social Engineering, Security Awareness, Organisationssicherheit, internationale Cyber-Security und Mensch-Maschine-Interaktion. Derzeit forscht er an einer Didaktik der Sicherheit und daran, IT-Sicherheit im Berufsbildungssystem zu verankern.

## Shellskripte mit Aha-Effekt III
### Shell-Shuffle für die Ohren

Hintergrund dieses Shellskripts ist eine Linux-Anwendung im Labor, die auf Tastendruck eine Aktion des Servers auslöst. Am Server sind aber weder Tastatur noch Monitor für Input und Output angeschlossen. Der Tastendruck wird also behelfsweise vollzogen. Und die akustische Quittung dieses Tastendruckes wird mit dem folgenden Skript lustig.

von **Jürgen Plate**

Den Zufall gibt die Vorsehung –
zum Zwecke muss ihn
der Mensch gestalten.

Friedrich Schiller, aus: Don Carlos

Um dem Server eine Reaktion zu entlocken, wanderte zunächst eine schöne große, rote Pilztaste an eine Statusleitung der seriellen Schnittstelle – die Sorte, die auch bei Fernsehshows beliebt ist. Es wird also nicht etwa eine Tastatur simuliert, sondern die serielle Schnittstelle abgefragt (ja, Server haben noch so etwas). Ein kleines C-Programm bewachte die Statusleitungen. Sobald die Taste einige Sekunden gedrückt wurde, startete es das hier gezeigte Shellskript, das die eigentliche Arbeit verrichtet.

Denn was fehlte, war eine Rückmeldung der Form „Tastendruck akzeptiert“. Anfangs geschah das durch Ausgabe eines Quittungstons über die Onboard-Soundkarte. Irgendwann war mir das zu langweilig, und es entstand folgendes Skript, das aus einer Menge von Sound-Dateien eine zufällig auswählt und den Namen an das aufrufende Skript übergibt. Das Skript zeigt, wie weit man mit einigen Shellkommandos kommen kann:

```
01 DIR="/home/local/sounds"
02 TMP="/tmp/"$(date +\%s)$$
03 RND=$(od -N2 -An -d /dev/random)
04
05 cd $DIR
06 ls *.wav > $TMP
07 MAX=`cat $TMP | wc -l`
08
09 NUM=$(( $RND \% $MAX + 1 ))
10 FILE=`head -$NUM $TMP | tail -1`
11
12 echo $FILE
13 rm $TMP
```

Nach Festlegen des Quellverzeichnisses für die Geräusche (Zeile 01) sind eine Temporärdatei (Zeile 02) und eine Zufallszahl zu erzeugen (Zeile 03), indem man sich einen 16-Bit-Integerwert aus */dev/random* erzeugt. Die Zufallszahl sorgt später für die Auswahl des Sounds. Alternativ ließe sich zum Beispiel auch mit */dev/urandom* spielen.

Jetzt wechselt das Skript an den Ort der Sounds (Zeile 05), merkt sich eine Liste aller Sound-Dateien (Zeile 06) und ermittelt ihre Anzahl (Zeile 07). Zeile 09 ermittelt für die Variable *NUM* einen Wert, der zwischen Untergrenze *1* und Obergrenze *MAX* liegt. Aus der Dateiliste wählt Zeile 10 mittels der Zufallszahl eine Zeile und damit einen Dateinamen aus, der schließlich das Ohr erreicht (Zeile 12). Nach Aufräumen der zu Beginn erzeugten Temporärdatei (Zeile 13) ist alles erledigt.

Damit das übergeordnete C-Programm, das die rote Taste bewacht, nun den neuen Shuffle-Sound statt des statischen Quittungstons abruft, muss in ihm das Shuffle-Sound-Skript *$(/home/local/zufall)* den statischen Dateinamen des Quittungstons ersetzen, zum Beispiel mittels `/usr/bin/play -V0 -v5 $(/home/local/zufall)`.

Tipp: Das Quellverzeichnis für den Shuffle-Sound mit möglichst abartigen Geräuschen und Musikfetzen füllen.

## Fosdem-Pickings
### *Sunxi*, *PicoTCP*, *Mageec* und Tux-Platinchen

Auf der diesjährigen *FOSDEM* (Free and Open Source Developers' European Meeting) Anfang Februar in Brüssel fielen in der Unzahl an Impulsen vier auf, die besondere Erwähnung verdienen. Natürlich gab es noch viel mehr. Aber man kann ja nicht alles haben.

von **Anika Kehrer**

Wer jede Entscheidung zu schwer nimmt, kommt zu keiner.

Harold Macmillan

Es handelt sich einmal um ein seit einigen Jahren entwickelten, ARM-basierten System-on-Chip des chinesischen Herstellers *Allwinner*, die nach Angaben des begeisterten Fosdem-Speakers mit Vollgas eine Community um sich sammeln. Als zweites hat sich eine belgische Firma hingesetzt und – das „Internet of Things" im Blick – TCP/IP quasi verkleinert. Als Drittes hat sich als Aussteller, Speaker und Devroom-Operator ein extrem stromsparendes Compiler-Projekt vorgestellt, das auf Automatisierung und lernende Algorithmen setzt. Schließlich hat sich die bulgarische Firma Olimex eine Art Hack ausgedacht, um die umher wuselnden Software-Entwickler zum Löten zu bewegen.

### *sunxi*

Mit dem niedrigpreisigen *sunxi*-SoC A10 habe Hersteller Allwinner den Markt 2011 überrascht, erklärte der Referent in seinem Vortrag *ARM: Allwinner sunxi SoCs and the community behind it* [1]. Seitdem finden sich die Nachfolgermodelle zunehmend in Billig-Tablets und TV-Sticks. Das habe schnell Hacker und Enthusiasten angelockt, sich mit der A-Serie zu befassen. Im Community-Projekt *Linux sunxi* [2] arbeiten sie daran, diese Chips zu portieren und zum Teil auch zu reverse-engineeren. Viele Treiber liegen quelloffen vor (Abbildung 1), doch einige Blobs betreffen zum Beispiel die Mali-GPU oder den GPS-Unterstützung. Immerhin sind eine Reihe offene Hardware-Boards erhältlich (Olimex, Cubietech).

Derzeit tummeln sich etwa 20 Entwickler und rund 600 Mailinglisten-User in dem Projekt. An Betriebssystemen sind Linux, FreeBSD, Tizen, Firefox OS und Minix in Arbeit. Auch Distributionen unterstützen den Allwinner-SoC: Der Referent zählte Fedora 18 und 19, Linaro, Arch Linux und Gentoo, Mer und Kali auf. Als besonders attraktiv an diesem Projekt stellte der Referent die breite Menge an Konsumentengeräten heraus (Tablets, Sticks, Settopboxen): Allwinners Marktanteil habe im Jahr 2013 nach Apple und vor Nvidia bei etwa 15 Prozent gelegen.


Abbildung 1: Open-Source-Unterstützung zählt *sunxi*-Hersteller Allwinner als eine seiner drei Marketingstärken, führte der Referent beeindruckt an. (Quelle: Vortragsfolien)

### *PicoTCP*

Die belgische Tech-Firma *tass* bietet eigentlich Beratung, Training und externen Service an. An „Connected Devices„ hat sie aber einen Narren gefressen. Also beschloss sie, einen TCP/IP-Stack für das „Internet der Dinge" zu kreieren, der sich an den IETF-Standard hält (Abbildung 2). Der davon berichtende Fosdem-Vortrag *PicoTCP* [3] gipfelte in eine Demo, wo der auf einem kleinen E-Piano erzeugte Ton über WLAN auf einem Laptop ausgegeben wurde (Abbildung 3).

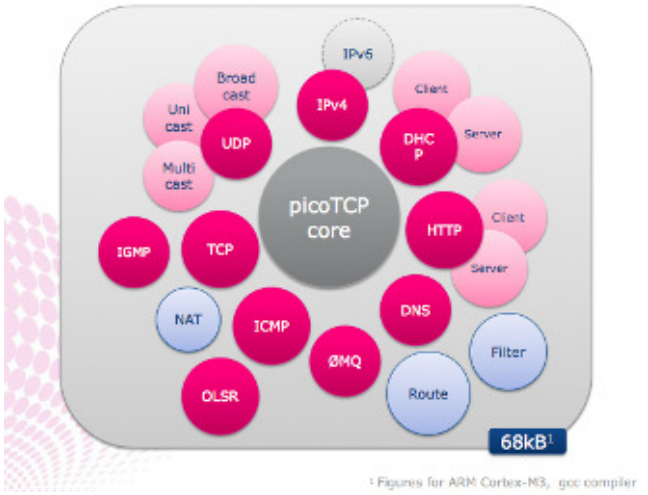



Abbildung 2: Modularität, Portabilität und Performance sind Hauptanliegen der quelloffen verfügbaren Referenzimplementierung des Mini-TCP/IP-Stacks *PicoTCP*. (Quelle: Vortragsfolien

Der erste Commit landete im Oktober 2012 auf Github. Das Repository ist auf der Unternehmensprojektseite [4] verlinkt. An Lizenzen sind GPLv2 sowie proprietäre Lizenzen verfügbar, wenn Letzteres für jemanden erforderlich ist. Unterstützte Architekturen sind derzeit zum Beispiel die 32-bittigen ARM-Varianten LPC1768 und LPC4357 von NXP sowie der STM32 von STMicroelectronics. Für 16 Bit funktioniert der MSP430-Microcontroller von Texas Instruments, für 8 Bit der AVR ATmega128. An Echtzeit-Betriebssystemen zählt die Feature-List FreeRTOS, mbed-RTOS und Linux auf.

Abbildung 3: Der PicoTCP-Vortrag enthielt eine Demo, bei dem ein am E-Piano erzeugter Midi-Sound drahtlos auf dem Laptop ausgegeben wurde. (Foto: Anika Kehrer)

Stolz führte der Referent die Projektbeschreibung des Ohloh-Verzeichnisses von PicoTCP an [5]: Demnach besitze das Projekt eine „junge, aber etablierte Code-Basis, gewartet von einem großen Entwicklerteam“. Wer nun ebenfalls Interesse hat, sich mit einem abgespeckten Mini-Internetprotokoll zu befassen, startet mit der rund 50-seitigen Benutzerdokumentation inklusive API-Beschreibungen [6].

### *Mageec*

Die „Machine Guided Energy Efficient Compilation“ (Mageec) ([7], Abbildung 4) ist ein gemeinsames Projekt der Firma Embecosm und der Universität Bristol, das finanzielle Trägerschaft der britischen Regierung genießt. Es geht darum, maschinelle Intelligenz zu nutzen, um Kompilierprozesse möglichst energieeffizient zu machen. Vielleicht hat vor einigen Jahren jemand von dem Compiler-Forschungsprojekt *Milepost* (Machine Learning for Embedded Programs Optimization) gehört, das von 2006 bis 2009 als EU-Projekt lief [8]. Das Mageec-Projekt nutzt nach eigenen Angaben viel davon.

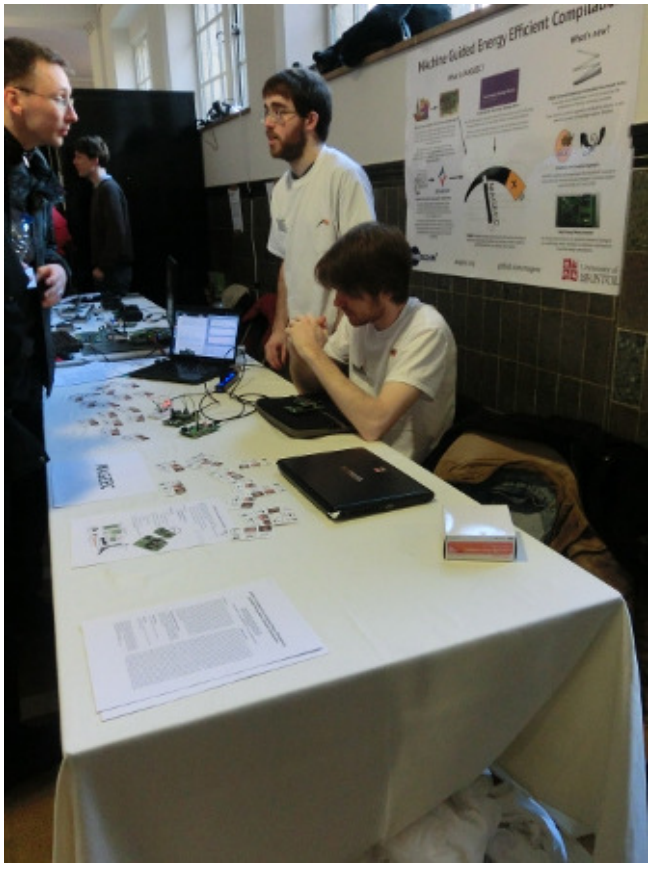


Abbildung 4: Gut, dass das Projekt einen Stand besetzte, wo man Fragen loswerden konnte: Die „Machine Guided Energy Efficient Compilation“ (Mageec) ist alles andere als trivial. (Foto: Anika Kehrer)

Hauptanliegen ist, Energie zu sparen. Zu dem Projekt gehören daher zum Beispiel Open-Hardware-Boards und Software zum Messen von

Energieverbrauch. Alle im Projekt entwickelten Komponenten sind bei Github [9] quelloffen verfügbar oder dokumentiert. Das Projekt startete im Juni 2013. Ziel ist, bis Ende 2014 die LLVM-Compiler-Infrastruktur [10] sowie die GNU Compiler Collection zu unterstützen.

## Zu guter Letzt: Mini-Pingi-Board

Wie bringt man Hardware-scheue Software-Entwickler zum Löten? Mit herzigen Mini-Boards. Diesen Proof-of-Concept erbrachte zumindest die bulgarische Firma Olimex, die einen kleinen Löttisch aufgebaut hatte. Passende Anwendungsfälle für ein vier mal fünf Zentimeter messendes Platinchen waren ebenfalls zu sehen.

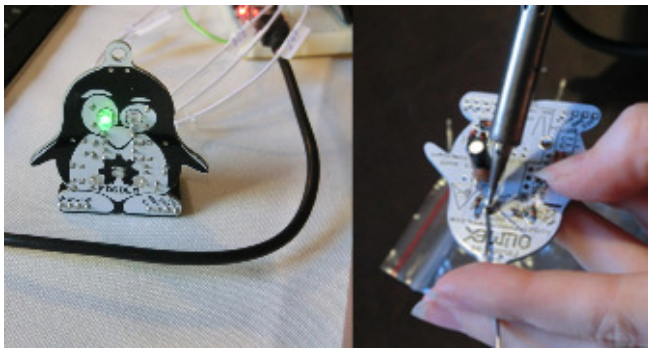


Abbildung 5: Mit herzigen Mini-Boards bringt man Hardware-scheue Software-Entwickler zum Löten. (Fotos: Anika Kehrer)

Die Firma Olimex baut und vertreibt eine Reihe von Open-Source-Hardware und fertigt Platinen in Wunschform an. Der extra für die Fosdem entstandene Bausatz *FOSDEM-85* ist auch jetzt noch für knapp neun Euro inklusive Versand auf der Firmen-Webseite erhältlich [11].

**Links**

[1] Fosdem-Vortrag *ARM: Allwinner sunxi SoC's and the community behind it* mit Slides:
https://fosdem.org/2014/schedule/event/arm_allwinner_sunxi_socs/
[2] Community-Projekt *Linux sunxi*: http://linux-sunxi.org
[3] Fosdem-Vortrag *PicoTCP* mit Slides:
https://fosdem.org/2014/schedule/event/deviot03/
[4] Projektseite: http://www.tass.be/en-gb/expertise/connected-devices/picotcp/
[5] Ohloh-Seite: http://www.ohloh.net/p/picotcp
[6] PicoTCP User Documentation:
http://mbed.org/media/uploads/daniele/user_doc.pdf
[7] Compiler-Forschungsprojekt *Mageec*: http://mageec.org
[8] Vorgänger-Forschungsprojekt *Milepost*:
http://www.linux-magazin.de/NEWS/Milepost-Neue-Intelligenz-fuer-die-Compiler-Sammlung-GCC
[9] *Mageec* bei Github: https://github.com/mageec/
[10] : LLVM-Compiler-Infrastruktur: http://llvm.org
[11] Pinguin-Platinchen:
https://www.olimex.com/Products/Duino/AVR/FOSDEM-85/open-source-hardware

**Über Anika**

Anika Kehrer ist freie Technikjournalistin und seit der Wiederauflage der UpTimes im Sommer 2012 als Chefredakteurin an Bord. In dieser Rolle hält sie die Fäden in der Redaktion zusammen und akquiriert und betreut die Artikel der redaktionellen Sommer- und Winterausgaben. Was das heißt? Die UpTimes als Fachmagazin und als Vereinszeitschrift der GUUG anspruchsvoll, transparent und partizipativ gestalten, in Zusammenarbeit mit dem Redaktionsteam aus GUUG-Mitgliedern.

## Hilfreiches für alle Beteiligten
### Autorenrichtlinien

Selbst etwas für die UpTimes schreiben? Gern! Als Thema ist willkommen, was ein GUUG-Mitglied interessiert und im Themenbereich der GUUG liegt. Was sonst noch zu beachten ist, steht in diesen Autorenrichtlinien.

Der Schriftsteller ragt zu den Sternen empor,
Mit ausgefranstem T-Shirt.
Er raunt seiner Zeit ihre Wonnen ins Ohr,
Mit ausgefranstem T-Shirt.

Frei nach Frank Wedekind, Die Schriftstellerhymne

Wir sind an Beiträgen interessiert. Wir, das ist diejenige Gruppe innerhalb der GUUG, die dafür sorgt, dass die UpTimes entsteht. Dieser Prozess steht jedem GUUG-Mitglied offen. Der Ort dafür ist die Mailingliste <redaktion@uptimes.de>.

## Welche Themen und Beitragsarten kann ich einsenden?

Die UpTimes richtet sich als Vereinszeitschrift der GUUG an Leser, die sich meistens beruflich mit Computernetzwerken, IT-Sicherheit, Unix-Systemadministration und artverwandten Themen auseinandersetzen. Technologische Diskussionen, Methodenbeschreibungen und Einführungen in neue Themen sind für dieses Zielpublikum interessant, Basiswissen im Stil von *Einführung in die Bourne Shell* hingegen eher nicht. Wer sich nicht sicher ist, ob sein Thema für die UpTimes von Interesse ist, kann uns gern eine E-Mail an <redaktion@uptimes.de> schicken.

Neben Fachbeiträgen sind Berichte aus dem Vereinsleben, Buchrezensionen, Konferenzberichte, humoristische Formen und natürlich Leserbriefe interessant. Wer nicht gleich mehrseitige Artikel schreiben möchte, beginnt also mit einem kleineren Beitrag.

Fachbeiträge sind sachbezogen, verwenden fachsprachliches Vokabular und anspruchsvolle Erläuterungen, besitzen technische Tiefe und ggf. auch Exkurse. Berichte aus dem Vereinsleben greifen aktuelle Themen auf oder legen Gedankengänge rund um die GUUG und ihre Community dar. Konferenzberichte zeigen, welche Veranstaltungen jemand besucht hat, was er/sie dort erfahren hat und ob die Veranstaltung nach Meinung des Autoren beachtenswert oder verzichtbar war. Unterhaltsame Formen können ein Essay oder eine Glosse sein, aber auch Mischformen mit Fachartikeln (Beispiel: der "Winter-Krimiin Ausgabe 2013-3). Auch unterhaltsame Formen besitzen jedoch inhaltlichen Anspruch. Denn die UpTimes ist und bleibt die Mitgliederzeitschrift eines Fachvereines.

In der UpTimes legen wir daher auch Wert auf professionelle publizistische Gepflogenheiten und einheitliche Schreibweisen. Dafür sorgt zum Beispiel ein einheitliches Layout der Artikel, oder etwa die grundsätzliche Vermeidung von Worten in Großbuchstaben (entspricht typografisches Schreien) oder von Worten in Anführungsstrichen zum Zeichen der Uneigentlichkeit (entspricht Distanzierung von den eigenen Worten). Wichtig sind außerdem beispielsweise Quellenangaben bei Zitaten, Kenntlichmachung fremder Gedanken, Nachvollziehbarkeit der Argumentation sowie Informationen zum Autor nach dem Artikel.

## In welchem Format soll ich meinen Artikel einsenden?

**ASCII:** Am liebsten blanke UTF8-Texte. Gern mit beschreibenden Anmerkungen oder Hinweisen (zum Beispiel mit Prozentzeichen zum Kenntlichmachen der Meta-Ebene).

**LaTeX:** Wir setzen die UpTimes mit LaTeX. Weil wir – wie es sich beim Publizieren gehört – mehrspaltig setzen und ein homogenes Erscheinungsbild anstreben, verwenden wir für die UpTimes bestimmte Formatierungen. Es ist nicht erwünscht, eigene Layoutanweisungen einzusenden. Wir behalten uns vor, Texte für die Veröffentlichung in der UpTimes umzuformatieren. Eine Vorlage mit den von uns verwendeten Auszeichnungen für Tabellen, Kästen und Abbildungen gibt es unter http://www.guug.de/uptimes/artikelvorlage.tex.

**Listings:** Der mehrspaltige Druck erlaubt maximal 45 Zeichen Breite für Code-Beispiele, inklusive 1 Leerzeichen und einem Zeichen für den Zeilenumbruch innerhalb einer Code-Zeile (Backslash). Breitere Listings formatieren wir um, verkleinern die Schriftgröße oder setzen sie als separate Abbildung.

**Bilder:** Wir verarbeiten gängige Bildformate, soweit ImageMagick sie verdaut und sie hochauflösend sind. Am besten eignen sich PNG- oder PDF-Bilddateien. Plant bei längeren Artikeln mit 1 Abbildung pro 3000 Zeichen. Das müssen nicht Bilder sein, sondern auch Tabellen, Listings oder ein Exkurskasten sind möglich. Verseht Eure Bilder nicht mit Rahmen oder Verzierungen, weil die Redaktion diese im UpTimes-Stil selbst vornimmt.

## Wie lang kann mein Artikel sein?

Ein einseitiger Artikel hat mit zwei Zwischentiteln um die 2.700 Anschläge. Mit etwa 15.000 Anschlägen – inklusive 3 Abbildungen – landet man auf rund vier Seiten. Wir nehmen gern auch achtseitige Artikel, achten dabei aber darauf, dass der Zusammenhang erhalten bleibt und dass es genug Bilder gibt, damit keine Textwüsten enstehen.

Wer Interesse hat, für die UpTimes zu schreiben, macht sich am besten um die Zeichenzahl nicht so viele Gedanken – auch für kurze oder lange Formate finden wir einen Platz. Die Redaktion ist bei der konkreten Ideenentwicklung gern behilflich. Für eine Artikelidee an <redaktion@uptimes.de> reicht es, wenn Ihr ein bestimmtes Thema behandeln wollt.

## Wohin mit meinem Manuskript?

Am einfachsten per E-Mail an <redaktion@uptimes.de> schicken. Das ist jederzeit möglich, spätestens jedoch vier Wochen vor dem Erscheinen der nächsten UpTimes. Zum Manuskript ist ein kleiner Infotext zum Autor wichtig, ein Bild wünschenswert.

Nützlich ist, wenn der Text vor Einsendung durch eine Rechtschreibkorrektur gelaufen ist. `aspell`, `ispell` oder `flyspell` für Textdateien sowie die von LibreOffice bieten sich an. Wenn Ihr Euren Text an die Redaktion schickt, solltet Ihr also weitestmöglich bereits auf die Rechtschreibung geachtet haben: Nach der Einschickung ist Rechtschreibung und Typo-Korrektur Aufgabe der Redaktion. Die Texte in der UpTimes folgen der neuen deutschen Rechtschreibung.

## Wie verlaufen Redaktion und Satz?

Wir behalten uns vor, Texte für die Veröffentlichung in der UpTimes zu kürzen und zu redigieren. Das bedeutet, dafür zu sorgen, dass der Artikel nicht ausufert, versehentliche Leeraussagen wegfallen, Syntax und Satzanschlüsse geglättet werden, dass Passiva und Substantivierungen verringert und Unklarheiten beseitigt werden (die zum Beispiel Fragen offen lassen oder aus Passivkonstruktionen resultieren, ohne dass der Schreibende das merkt). Manchmal ist dieser Prozess mit Nachfragen an den Autoren verbunden.

Die endgültige Textversion geht jedem Autoren am Ende zur Kontrolle zu. Dabei geht es um die inhaltliche Kontrolle, ob sich durch den Redaktionsprozess Missverständnisse oder Falschaussagen entstanden sind. Danach setzt die Redaktion die Artikel. Wenn der Satz weitgehend gediehen ist – also ein *Release Candidate* als PDF vorliegt – erhalten die Autoren als erste diesen RC. Danach wird die UpTimes dann veröffentlicht.

## Gibt es Rechtliches zu beachten?

Die Inhalte der UpTimes stehen ab Veröffentlichung unter der CC-BY-SA-Lizenz, damit jeder Leser die Artikel und Bilder bei Nennung der Quelle weiterverbreiten und auch weiterverarbeiten darf. Bei allen eingereichten Manuskripten gehen wir davon aus, dass der Autor sie selbst geschrieben hat und der UpTimes ein nichtexklusives, aber zeitlich und räumlich unbegrenztes Nutzungs- und Bearbeitungsrecht unter der CC-BY-SA einräumt.

Bei Fotos oder Abbildungen Dritter ist es rechtlich unabdingbar, dass der Autor sich bei dem Urheber die Erlaubnis zu dieser Nutzung einholt, und fragt, wie die Quelle genannt zu werden wünscht. Die Frage nach der CC-BY-SA ist hierbei besonders wichtig.

An Exklusivrechten, wie sie bei kommerziellen Fachzeitschriften üblich sind, hat die UpTimes kein Interesse. Es ist den Autoren freigestellt, ihre Artikel noch anderweitig nach Belieben zu veröffentlichen.

## Bekomme ich ein Autorenhonorar?

Für Fach- und literarische Beiträge zahlt die GUUG dem Autor nach Aufforderung durch die Redaktion und Rechnungstellung durch den Autor pro Seite 50 € zuzüglich eventuell anfallender USt. Beiträge für die Rubrik „Vereinsleben“,

Buchrezensionen und Artikel bezahlter Redakteure sind davon ausgenommen. Gleiches gilt für Paper, wenn die UpTimes die Proceedings der Konferenz enthält.

**Nächste redaktionelle Ausgabe:**
**UpTimes 2014-2, Winterausgabe**

- Redaktionsschluss: Sonntag, 26. Oktober 2014.
- Erscheinung: Sonntag, 30. November 2014.
- Gesuchte Inhalte: Fachbeiträge über Unix und verwandte Themen, Veranstaltungsberichte, Rezensionen, Beiträge aus dem Vereinsleben.
- Fragen, Artikelideen und Manuskripte an: <redaktion@uptimes.de>
- LaTeX-Template für UpTimes-Artikel: http://www.guug.de/uptimes/artikel-vorlage.tex

# Über die GUUG
## German Unix User Group e.V.

Vereinigung deutscher Unix-Benutzer

Die Vereinigung Deutscher Unix-Benutzer hat gegenwärtig rund 700 Mitglieder, davon etwa 90 Firmen und Institutionen.

Im Mittelpunkt der Aktivitäten der GUUG stehen Konferenzen. Ein großes viertägiges Event der GUUG hat eine besondere Tradition und fachliche Bedeutung: In der ersten Jahreshälfte treffen sich diejenigen, die ihren beruflichen Schwerpunkt im Bereich der IT-Sicherheit, der System- oder Netzwerkadministration haben, beim *GUUG-Frühjahrsfachgespräch* (FFG).

Seit Oktober 2002 erscheint mit der *Uptimes* – die Sie gerade lesen – eine Vereinszeitung. Seit 2012 erscheint die Uptimes einerseits zu jedem FFG in Form einer gedruckten Proceedings-Ausgabe (ISBN), und andererseits im Rest des Jahres als digitale Redaktionsausgabe (ISSN). Daneben erhalten GUUG-Mitglieder zur Zeit die Zeitschrift *LANline* aus dem Konradin-Verlag kostenlos im Rahmen ihrer Mitgliedschaft.

Schließlich gibt es noch eine Reihe regionaler Treffen (http://www.guug.de/lokal): im Rhein-Ruhr- und im Rhein-Main-Gebiet sowie in Berlin, Hamburg, Karlsruhe und München.

## Warum GUUG-Mitglied werden?

Die GUUG setzt sich für eine lebendige und professionelle Weiterentwicklung im Open Source-Bereich und für alle Belange der System-, Netzwerkadministration und IT-Sicherheit ein. Wir freuen uns besonders über diejenigen, die bereit sind, sich aktiv in der GUUG zu engagieren. Da die Mitgliedschaft mit jährlichen Kosten

| | |
|---|---|
| Fördermitglied | 350 € |
| persönliches Mitglied | 90 € |
| in der Ausbildung | 30 € |

verbunden ist, stellt sich die Frage, welche Vorteile damit verbunden sind?

Neben der Unterstützung der erwähnten Ziele der GUUG profitieren Mitglieder auch finanziell davon, insbesondere durch die ermäßigten Gebühren bei den Konferenzen der GUUG und denen anderer europäischer UUGs. Mitglieder bekommen außerdem *c't* und *iX* zum reduzierten Abopreis.

## Wie GUUG-Mitglied werden?

Füllen Sie einfach das umseitige Anmeldeformular aus und schicken Sie es per Fax oder Post an die unten auf dem Formular angegebene Adresse. Falls Sie die Seite nicht herausreißen wollen: Sie können den Mitgliedsantrag als PDF herunterladen, siehe URL auf dem Mitgliedsantrag.

## Impressum

Uptimes – Mitgliederzeitschrift der
German Unix User Group (GUUG) e.V.
**Herausgeber:** GUUG e.V.
Murschall 5
D-84529 Tittmoning
Tel.: +49-(89)-380 125 95 0
Fax: +49-(89)-380 125 95 9
E-Mail: <redaktion@uptimes.de>
Internet: http://www.guug.de/uptimes/

**Autoren dieser Ausgabe:** Wolfgang Stief, Programmteam FFG, Mathias Weidner, Nils Magnus, Werner Heuser, Frank Hofmann, Jürgen Plate, Stefan Schumacher, Anika Kehrer
**V.i.S.d.P:** Wolfgang Stief, Vorstandsvorsitzender, Anschrift siehe Herausgeber
**Chefredaktion:** Anika Kehrer
**Redaktion:** Mathias Weidner
**LaTeX-Layout (PDF):** Robin Schröder
**XHTML-Layout (ePub):** Mathias Weidner
**Titelbild:** Hella Breitkopf
**Titelgestaltung:** Hella Breitkopf
**Bildnachweis:** Comicreihe *geek & poke* CC-by-sa, mit freundlicher Genehmigung von Oliver Widder. Andere Quellennachweise am jeweiligen Bild.
**Verlag:** Lehmanns Media GmbH, Hardenbergstraße 5, 10623 Berlin
**ISSN:** 2195-0016

Wenn Sie Interesse an Anzeigen in der Uptimes haben,
wenden Sie sich bitte an <werbung@guug.de>.