KaffeeKlatsch

Das Magazin rund um Software-Entwicklung

ISSN 1865-682X

04/2016

Jahrgang 9

KaffeeKlatsch

Das Magazin rund um Software-Entwicklung

Sie können die elektronische Form des KaffeeKlatschs
monatlich, kostenlos und unverbindlich
durch eine E-Mail an

abo@bookware.de

abonnieren.

Ihre E-Mail-Adresse wird ausschließlich für den Versand
des KaffeeKlatschs verwendet.

Editorial

Was lange währt …

Wenn man fragt, was eigentlich unixoide Systeme ausmacht, dann bekommt man recht unterschiedliche Antworten. Was aber diese Systeme einzigartig macht, sind seine kleinen Werkzeuge, die nach Belieben miteinander kombiniert werden können.

Unix an und für sich ist ja schon interessant genug. Schon lange bevor die Objektorientierung Ruhm und Ehre erlangte, wurden derartige Konzepte schon im Kern verwendet. Und erst das Dateisystem, oder das Speichermanagement – unschlagbar gut. Aber das sind ja alles Dinge, die im Geheimen – sozusagen „unter der Haube" – glänzen. Der eigentliche Gewinn für Entwickler ist die Shell.

Die unzählige Vielfalt an kleinen, nützlichen Werkzeugen, die meist textbasiert ganz einfach mit Hilfe des Pipe-Symbols miteinander kombiniert werden können, machen ein unixoides System konkurrenzlos. Will man etwa alle Java-Klassen zählen, so bedarf es zwar einiger Kenntnis über die Befehle; aber ein Einzeiler genügt um das gewünschte Ergebnis zu bekommen. Oder eine grobe Abschätzung darüber, wie viele Zeilen Code (ohne Kommentare) ein Projekt hat, lässt sich genau so einfach herausfinden, wie ein Bild mit Hilfe von ASCII-Zeichen darzustellen.

Aber warum erzähle ich Ihnen das?

Das alles gibt es jetzt auch für Windows. Was eigentlich schon für Windows NT (zumindest theoretisch) vorgesehen war, gibt es jetzt endlich auch aus dem Hause Microsoft für das neue Windows. User-Code-Linux (Ubuntu) läuft – ohne virtuelle Maschine – direkt auf dem Windows-Kern. Und damit läuft auch die bash-Shell direkt unter Windows. Endlich!

Allerdings stellt sich nun eine Frage, die bereits früher schon einmal gestellt wurde. Als IBM 1987 das neue, bessere OS/2 vorstellte, warben sie auch damit, dass Windows-Code uneingeschränkt darauf laufen würde. Der große Erfolg blieb dann leider auch aus, denn warum sollten die Hersteller und Anwender zu OS/2 wechseln, wenn sie Windows-Programme auch „nur" unter Windows zum Laufen bekommen.

Und jetzt stellt sich doch die analoge Frage: Wenn Linux uneingeschränkt unter Windows läuft, wozu brauche ich dann noch Windows?

Herzlichst
Ihr Michael Wiedeking
Herausgeber

Referenzen

[1] Welch, Chris *Microsoft is adding the Linux command line to Windows 10* The Verge, 30. März 2016 http://www.theverge.com/2016/3/30/11331014/microsoft-windows-linux-ubuntu-bash

Beitragsinformation

Der KaffeeKlatsch dient Entwicklern, Architekten, Projektleitern und Entscheidern als Kommunikationsplattform. Er soll neben dem Know-how-Transfer von Technologien (insbesondere Java und .NET) auch auf einfache Weise die Publikation von Projekt- und Erfahrungsberichten ermöglichen.

Beiträge

Um einen Beitrag im KaffeeKlatsch veröffentlichen zu können, müssen Sie prüfen, ob Ihr Beitrag den folgenden Mindestanforderungen genügt:

- Ist das Thema von Interesse für Entwickler, Architekten, Projektleiter oder Entscheider, speziell wenn sich diese mit der Java- oder .NET-Technologie beschäftigen?
- Ist der Artikel für diese Zielgruppe bei der Arbeit mit Java oder .NET relevant oder hilfreich?
- Genügt die Arbeit den üblichen professionellen Standards für Artikel in Bezug auf Sprache und Erscheinungsbild?

Wenn Sie uns einen solchen Artikel, um ihn in diesem Medium zu veröffentlichen, zukommen lassen, dann übertragen Sie Bookware unwiderruflich das nicht exklusive, weltweit geltende Recht

- diesen Artikel bei Annahme durch die Redaktion im KaffeeKlatsch zu veröffentlichen
- diesen Artikel nach Belieben in elektronischer oder gedruckter Form zu verbreiten
- diesen Artikel in der Bookware-Bibliothek zu veröffentlichen
- den Nutzern zu erlauben diesen Artikel für nicht-kommerzielle Zwecke, insbesondere für Weiterbildung und Forschung, zu kopieren und zu verteilen.

Wir möchten deshalb keine Artikel veröffentlichen, die bereits in anderen Print- oder Online-Medien veröffentlicht worden sind.

Selbstverständlich bleibt das Copyright auch bei Ihnen und Bookware wird jede Anfrage für eine kommerzielle Nutzung direkt an Sie weiterleiten.

Die Beiträge sollten in elektronischer Form via E-Mail an redaktion@bookware.de geschickt werden.

Auf Wunsch stellen wir dem Autor seinen Artikel als unveränderlichen PDF-Nachdruck in der kanonischen KaffeeKlatsch-Form zur Verfügung, für den er ein unwiderrufliches, nicht-exklusives Nutzungsrecht erhält.

Leserbriefe

Leserbriefe werden nur dann akzeptiert, wenn sie mit vollständigem Namen, Anschrift und E-Mail-Adresse versehen sind. Die Redaktion behält sich vor, Leserbriefe – auch gekürzt – zu veröffentlichen, wenn dem nicht explizit widersprochen wurde.

Sobald ein Leserbrief (oder auch Artikel) als direkte Kritik zu einem bereits veröffentlichten Beitrag aufgefasst werden kann, behält sich die Redaktion vor, die Veröffentlichung jener Beiträge zu verzögern, so dass der Kritisierte die Möglichkeit hat, auf die Kritik in der selben Ausgabe zu reagieren.

Leserbriefe schicken Sie bitte an leserbrief@bookware.de. Für Fragen und Wünsche zu Nachdrucken, Kopien von Berichten oder Referenzen wenden Sie sich bitte direkt an die Autoren.

Werbung ist Information

Firmen haben die Möglichkeit Werbung im KaffeeKlatsch unterzubringen. Der Werbeteil ist in drei Teile gegliedert:

- Stellenanzeigen
- Seminaranzeigen
- Produktinformation und -werbung

Die Werbeflächen werden als Vielfaches von Sechsteln und Vierteln einer DIN-A4-Seite zur Verfügung gestellt.

Der Werbeplatz kann bei Frau Natalia Wilhelm via E-Mail an anzeigen@bookware.de oder telefonisch unter 09131/8903-90 gebucht werden.

Abonnement

Der KaffeeKlatsch erscheint zur Zeit monatlich. Die jeweils aktuelle Version wird nur via E-Mail als PDF-Dokument versandt. Sie können den KaffeeKlatsch via E-Mail an abo@bookware.de oder über das Internet unter www.bookware.de/abo bestellen. Selbstverständlich können Sie das Abo jederzeit und ohne Angabe von Gründen sowohl via E-Mail als auch übers Internet kündigen.

Ältere Versionen können einfach über das Internet als Download unter www.bookware.de/archiv bezogen werden.

Auf Wunsch schicken wir Ihnen auch ein gedrucktes Exemplar. Da es sich dabei um einzelne Exemplare handelt, erkundigen Sie sich bitte wegen der Preise und Versandkosten bei Natalia Wilhelm via E-Mail unter natalia.wilhelm@bookware.de oder telefonisch unter 09131/8903-90.

Copyright

Impressum

KaffeeKlatsch Jahrgang 9, Nummer 4, April 2016
ISSN 1865-682X
Bookware – eine Initiative der
MATHEMA Verwaltungs- und Service-Gesellschaft mbH

Henkestraße 91, 91052 Erlangen
Telefon: 0 91 31 / 89 03-90
Telefax: 0 91 31 / 89 03-99
E-Mail: redaktion@bookware.de
Internet: www.bookware.de

Herausgeber/Redakteur: Michael Wiedeking
Anzeigen: Natalia Wilhelm
Grafik: Nicole Delong-Buchanan

Inhalt

Artikel

Daten sichern – aber sicher

von Dr. Frank Lauterwald

Verschlüsselung ist in letzter Zeit ein beliebtes Thema. Sie dient dazu, Daten vor den Augen Unbefugter zu schützen. Das ist nicht einmal sonderlich schwierig. Schwieriger ist da schon das Zusammenspiel mit Backups, Synchronisation zwischen Rechnern und Erreichbarkeit von Unterwegs.

Kolumne

Bedingte Bit-Setzerei (exklusiv verbessert)

von Michael Wiedeking

Die letzte Kolumne bot eine ziemlich effiziente Methode, um Bits (ohne ein *if* zu verwenden) bedingt zu setzen oder zu löschen. Dort wurde auch behauptet, dass jene fünf Instruktionen noch zu überbieten wären. Was hier geschehen soll.

Herbstcampus

Wissenstransfer par excellence

Daten sichern – aber sicher

Wie verheiratet man Verschlüsselung, Datensicherung und Synchronisation?

von Dr. Frank Lauterwald

Verschlüsselung ist in letzter Zeit ein beliebtes Thema. Sie dient dazu, Daten vor den Augen Unbefugter zu schützen. Das ist nicht einmal sonderlich schwierig. Schwieriger ist da schon das Zusammenspiel mit Backups, Synchronisation zwischen Rechnern und Erreichbarkeit von Unterwegs.

Der Autor betrieb seinen ersten Web-Server um die Jahrtausendwende an einer DSL-Leitung[1] aus dem Keller seiner Eltern heraus. Eines Tages war die Hauptseite durch einen Redirect auf eine Seite im digitalen Rotlichtmilieu ersetzt. Wer das unauffällige Ziel übernommen hat, ist bis heute unklar, ein echter Schaden ist dadurch glücklicherweise nicht entstanden.

Ein Bekannter des Autors wurde einmal Ziel einer Hausdurchsuchung. Der Grund: er hatte als Jugendlicher einen Joint geraucht und sich dabei erwischen lassen. Viele Jahre später wurde ein Mann gleichen Namens wegen eines deutlich schwerwiegenderen Vergehens gesucht. Dieser war allerdings unbekannt verzogen. Es lag also nahe, bereits auffällig gewordene Personen mit gleichem Namen und bekanntem Aufenthaltsort aufzusuchen.

Diese Anekdoten zeigen, dass nicht viel dazugehört, um ins Visier staatlicher oder privater Stellen zu geraten. Man muss dazu durchaus kein Terrorist sein. Dennoch dürfte jede(r) etwas zu verbergen haben – und sei es nur vor bestimmten Personen oder Kreisen. Wie man Mails verschlüsselt, hat mein Kollege Thomas Bertz bereits im KaffeeKlatsch vorgestellt [1]. Dieser Artikel handelt davon, wie man Bestandsdaten schützt.

Verschlüsselung alleine macht jedoch nicht glücklich. Sie ist nur ein Mittel, um Unbefugten den Zugriff auf unsere Daten zu verwehren. Wäre das das einzige Ziel, so wäre Datensparsamkeit das Mittel der Wahl: Einfach in den digitalen Reißwolf mit den Daten und wir wären alle Sorgen los.

Stattdessen wollen wir den Zugriff auf einige Daten selbst nicht aufgeben. Meist geht es dabei nicht nur um Archive, sondern um Dateien, mit denen wir tagtäglich arbeiten. Die Verschlüsselung soll daher das normale Arbeiten möglichst wenig beeinträchtigen. Für das einfache Lesen und Schreiben von Dateien lässt sich das problemlos bewerkstelligen: Vor dem ersten Zugriff muss ein Passwort eingegeben werden, das spätestens mit dem Abschalten des Rechners wieder aus dem Speicher verschwindet.

Schwierig wird es, wenn man die Verschlüsselung in komplexe bestehende Prozesse integrieren will. Ein wichtiger Punkt sind hier Backups – denn das Ziel ist ja gerade, dass wir weiterhin Zugriff auf unsere Daten haben. Falls das bereits durch eine kaputte Festplatte verhindert wird, haben wir unser Ziel verfehlt. Noch interessanter wird es, wenn wir unsere Daten auf verschiedenen Rechnern verwenden wollen. Benutzen Sie mehrere Rechner? Falls ja: haben Sie sich in den letzten Monaten zumindest einmal gefragt, auf welchem die aktuelle Version des Quartalsberichts liegt?

Alles hat einen Rahmen ...

In diesem Artikel wird eine mögliche Lösung für die genannten Probleme aufgezeigt. Wie so oft geht das leider nicht ohne Voraussetzungen.

[1] Mit – für damalige Verhältnisse – beachtlichen 128kbit/s Upstream. Wenn Sie heute nach der Durchschnittsgröße von aktuellen Webseiten suchen, erhalten Sie – je nach Quelle – Werte um 2 MB. Die Ladezeit einer heute durchschnittlichen Seite hätte damals also etwas über zwei Minuten betragen – falls Sie die Leitung für sich gehabt hätten.

Die wichtigste: Ihr Aufgabenprofil passt. Das heißt: Sie wollen Ihre Daten verschlüsselt halten, haben mindestens zwei Rechner zur Verfügung, und wollen die Daten zwischen diesen Rechnern synchron halten.

Die zweitwichtigste: Sie verwenden Linux. Nein, das war gelogen. Dieser Artikel behandelt zwar vor allem Linux, mit Windows oder OS X geht es aber genauso. Sogar ein Mischen verschiedener Betriebssysteme ist kein Problem. Hilfreich (aber nicht zwingend) ist es, wenn Sie ein wenig mit einer Kommandozeile Ihrer Wahl vertraut sind. Außerdem sollten Sie zumindest einen Rechner mit einem SSH-Server[2] Ihr Eigen nennen. Für unixoide Systeme (Linux, OSX, Solaris, *BSD, …) ist das vermutlich sowieso der Fall; im Falle von Windows lässt es sich durch Installieren der richtigen Software lösen.

Synchronisation und Backups mit *git*

Lassen wir die Verschlüsselung noch einen Moment außen vor und betrachten zunächst Synchronisation und Backup.

Ein für diese Zwecke bisher eher vernachlässigtes Werkzeug ist das verteilte Versionsverwaltungssystem git. Es wurde eigentlich zur Verwaltung von Quellcode entwickelt, eignet sich jedoch auch für beliebige andere Dokumente. git verwaltet und synchronisiert verschiedene Versionen von Dateien zwischen mehreren Rechnern.

Warum nun plötzlich verschiedene Versionen? Wenn Sie nur einmal monatlich Ihren aktuellen Kontoauszug speichern, können Sie diesen Punkt ignorieren und einfach den aktuellen Stand Ihres Dokumenten-Verzeichnisses auf ein Backup-Medium kopieren. Wie viele Backups bewahren Sie aber auf? Und prüfen Sie den aktuellen Stand rigoros, bevor Sie ein altes Backup überschreiben? Stellen Sie sich vor, Sie fangen sich einmal unbemerkt einen Verschlüsselungstrojaner ein und überschreiben später Ihr Backup mit einer verschlüsselten Fassung, auf die Sie selbst keinen Zugriff mehr haben. Oder Ihr Office-Programm schreibt eine korrupte Datei, die sich nicht mehr öffnen lässt. Oder Ihre Festplatte oder Ihr Arbeitsspeicher greifen manchmal daneben.[3]
In diesen Fällen zahlt es sich aus, wenn Sie den vorherigen Stand Ihrer Daten aufbewahrt haben. Die erste Empfehlung lautet also: halten Sie Ihre wichtigen Daten in *git*-Repositories. Ein leeres Repository erzeugen Sie mit dem Befehl

```
git init
```

Hierdurch passiert noch nicht viel. *git* legt lediglich ein Verzeichnis *.git* an und speichert dort einige Metadaten. Nun können Sie mittels

```
git add <Dateiname>
```

beliebige Dateien unter Versionsverwaltung stellen. Durch

```
git commit
```

wird der aktuelle Stand der versionierten Dateien gespeichert. Auf diesen Stand können Sie später nach Belieben wieder zugreifen. Spätere Änderungen sichern Sie analog mittels git add und git commit. Für eine tiefere Beschäftigung mit git sei auf die vielen Quellen im Web, z. B. [2] verwiesen.

Die Datenhaltung in einem Repository schützt Sie bereits gegen einfache Schäden wie versehentliches Löschen oder einzelne beschädigte Dateien. Diese Sicherheit bezahlen Sie mit einem Mehrverbrauch an Plattenplatz. In einem kleinen Versuch ergab sich folgendes Bild: Nach dem Initialisieren eines Repositories mittels git init belegte dieses vernachlässigbare 100 KB. Nachdem eine ca. 300 MB große Videodatei dazukopiert wurde, stieg der Platzbedarf wenig verwunderlich auf 316316 KB. *git add* verdoppelte diesen Bedarf annähernd auf 632564 KB, während git commit mit einer Steigerung auf 632608 KB wieder nur marginalen Einfluss hatte. Rechnen Sie also grob mit einem Platzbedarf, der der Summe der aktuellen Dateien plus der gespeicherten Versionen entspricht, wobei sich bei mehreren Versionen noch Vorteile durch die Kompression ergeben. Wenn möglich werden nämlich nur die Änderungen gespeichert und nicht der vollständige Stand.

Für Ihre Filmsammlung kommt Sie dieses Vorgehen also möglicherweise teuer zu stehen; für Ihre Kontoauszüge und die Korrespondenz mit Ihrer Krankenkasse müssen Sie sich eher keine Gedanken machen. Der Autor verwaltet seine Dokumente seit ca. 2009 (inklusive älterer eingescannter Unterlagen) auf diese Weise und die Größe des Repositories liegt aktuell bei ca. 6 GB. Das sind die Dinge, die wirklich wichtig sind; weniger relevante Daten behandelt man möglicherweise anders.

[2] Secure Shell Server.

[3] Der Rechner des Autors litt vor vielen Jahren einmal an einem RAM-Fehler, der im laufenden Betrieb unbemerkt blieb. Der betroffene Rechner lief ohne Absturz über zwei Wochen, allerdings waren alle mit *bzip2* komprimierten Backups korrumpiert und nicht mehr lesbar. Bei einer genaueren Analyse benötigte sogar *memtest86* über eine Stunde, um den Speicherfehler zu bemerken. Das Problem darf also als recht subtil bezeichnet werden; dennoch war es ausreichend, um alle Backups unlesbar zu machen..Als Lösung blieb damals nur das Herauspatchen der Prüfsummen-Checks aus *bzip2*, um zumindest an die unbeschädigten Teile zu kommen.

Synchronisation

Dieser Zwischenschritt war nötig, denn nun wenden wir uns der Synchronisation zwischen verschiedenen Rechnern zu. Damit verfolgen wir zwei Ziele: erstens schützt sie uns vor größeren Schäden an einem Rechner, z. B. einem Plattenausfall oder dem Verlust des ganzen Repositories.[4] Zweitens können Sie an mehreren Rechnern mit den gleichen Dateien arbeiten, ohne sich Gedanken darüber machen zu müssen, auf welchem nun gerade der aktuelle Stand welcher Datei liegt. Diese Synchronisation bekommen Sie mit git nahezu geschenkt. Mittels

```
git clone <ip-addresse>:<pfad>
```

erzeugen Sie eine komplette Kopie eines Repositories von einem entfernten Rechner. Spätere Änderungen holen Sie mit

```
git pull
```

von dort ab und integrieren sie in Ihre Kopie. Ihre lokalen Änderungen propagieren Sie mittels

```
git push
```

wieder zurück. Am einfachsten ist es, wenn Sie einen Rechner zum Master erklären und alle anderen Rechner gegen diesen synchronisieren. Diesen Master bezeichnen wir im Folgenden als Server, es kann jedoch ein beliebiger Rechner sein, den Sie ganz normal für sonstige Aufgaben verwenden können. Lediglich die IP-Adresse sollte sich für die Synchronisation nicht ändern. Das bewerkstelligen Sie einfach durch eine feste IP-Zuweisung in Ihrem Router.

Erreichbarkeit

Soll der Server von außerhalb der eigenen vier Wände erreichbar sein, hat man verschiedene Möglichkeiten.

Am einfachsten ist es, einen dedizierten Server in einem Rechenzentrum anzumieten. Begnügt man sich mit einem virtuellen Server, bekommt man bei den großen Discountern mehrere hundert Gigabyte Plattenplatz für weniger als 10 Euro im Monat. Allerdings hat man das Gerät nicht physisch unter Kontrolle, was für die Sicherheit nicht gerade ein Vorteil ist.

Im Gegenzug bekommt man implizit ein Offsite-Backup, ist also gegen Schäden im eigenen Haus abgesichert, die alle dortigen Rechner betreffen.

Die Alternative ist ein dynamischer DNS-Dienst. [3] Hierbei betreiben Sie selbst einen Server von zu Hause, den Sie von außen ansprechen können. Dazu müssen Sie einerseits ihrem Router beibringen, die benötigten Ports (für git über ssh: Port 22) an diesen Rechner weiterzuleiten. Zum zweiten benötigen Sie einen DNS-Namen, über den Sie auf Ihren Rechner zugreifen, denn Ihre öffentliche IP-Adresse ändert sich bei den meisten Telefonanbietern häufiger. Ein DynDNS-Dienst stellt Ihnen nun einen DNS-Eintrag zur Verfügung, den Sie immer auf ihre gerade gültige IP-Adresse aktualisieren können. Eine (etwas veraltete) Übersicht an DynDNS-Anbietern findet sich in [4]. Beim Autor dieses Artikels ist der kostenlose Dienst von http://freedns.afraid.org/ im Einsatz.

Die Aktualisierung soll natürlich automatisch erfolgen. Idealerweise können Sie dazu Ihren Router nutzen; die meisten Modelle bringen die nötige Software bereits mit. Viele Router bieten jedoch nur eine geringe Auswahl an Anbietern, darunter keine kostenlosen.[5]

Dies ist insofern kein Problem, als für den Remote-Zugriff sowieso ein Rechner laufen muss. Dieser kann sich gleich noch um die DynDNS-Aktualisierung kümmern. Auf Unix-artigen Systemen verwenden Sie dazu das Programm ddclient, das durch eine einfache Textdatei konfiguriert wird:

```
# Configuration file for ddclient # /etc/ddclient.conf
# check every xxx seconds
daemon=120
syslog=yes # log update msgs to syslog
pid=/var/run/ddclient.pid # pidfile
protocol=freedns
use=web
server=freedns.afraid.org
login=<login>
password=<passwort>
<hostname>
```

Viele Linux-Distributionen (z.B. Debian) bringen außerdem Konfigurationswerkzeuge mit, die Ihnen das Schreiben dieser Textdatei abnehmen. In der Beispielkonfiguration prüft der Rechner nun alle zwei Minuten, ob sich die öffentliche IP-Adresse geändert hat und aktualisiert gegebenenfalls den DNS-Eintrag.

Soll ein Rechner zu Hause dauerhaft laufen, sind die Stromkosten ein nicht zu unterschätzender Faktor. Ein Watt im Dauerbetrieb schlägt mit knapp 2,50 EUR im Jahr zu Buche (bei Stromkosten von 28 ct/kWh).[6]

[4] Gegen einen Festplattenschaden können Sie sich natürlich auch mit einem RAID-System absichern. Gegen den schon erwähnten Verschlüsselungstrojaner oder versehentliches Löschen hilft Ihnen das jedoch nichts. Diese unerwünschten Änderungen werden sofort auf alle Platten gespiegelt.

[5] So z.B. die EasyBox des Autors, ein Zwangskauf von Vodafone. Diese hat allerdings vor wenigen Wochen das Zeitliche gesegnet.

[6] Nach einer Rechnung von *http://www.energiesparen-blog.net/strom-sparen/stromkosten-im-standby-selbst-errechnen-was-kostet-1-watt-pro-jahr/*. Der angenommene Preis passt zufällig sehr präzise zu dem der hiesigen Stadtwerke.

Läuft der Rechner also dauerhaft, so verursacht er ab einem Verbrauch von 48 Watt alleine durch den Strombedarf höhere Kosten als der oben diskutierte virtuelle Server in einem Rechenzentrum.

Für derartige always-on-Szenarien bieten sich daher zwei Lösungen an:

1. der Server wird erst dann hochgefahren, wenn man ihn benötigt. Dieser Punkt wird hier nicht näher beleuchtet; zwar können viele Rechner über Wake-on-LAN gestartet werden; die nötigen Datenpakete von außen durch einen Router zu bekommen, rechtfertigt aber vermutlich einen eigenen Artikel.
2. man verwendet einen extrem stromsparenden Server. Hier bieten sich Kleincomputer wie der Raspberry Pi an. Deren Einrichtung ist jedoch etwas hakelig, weshalb wir ihnen einen eigenen Artikel in einer kommenden Ausgabe widmen werden. Das gilt besonders für die im nächsten Abschnitt beschriebene Verschlüsselung – denn den Systemen, die mit vielen Kleincomputern ausgeliefert werden, fehlt es dazu an erforderlichen Kernel-Erweiterungen.

Verschlüsselung

Nun können wir uns endlich dem eigentlichen Thema zuwenden – der Verschlüsselung zum Schutz gegen unbefugten Zugriff. Hier ist zunächst die Granularität zu klären. Verschlüsseln können Sie zum Beispiel einzelne Dateien, ganze Festplatten oder sogenannte Container-Dateien.

Wir beschäftigen uns hier mit Containern. Dabei wird eine Datei erzeugt, die wiederum ein vollständiges verschlüsseltes Dateisystem enthält. Lange Zeit war das Werkzeug der Wahl hierzu TrueCrypt [5], das jedoch im Mai 2014 von seinen Entwicklern aufgegeben wurde. Ein Vorteil von TrueCrypt war, dass es für alle gängigen Betriebssysteme verfügbar war. Mit VeraCrypt steht zwar eine Art Nachfolger zur Verfügung; wir konzentrieren uns hier jedoch auf die Linux-Bordmittel, mit denen Sie dasselbe erreichen können. Eine knappe, aber verständliche Anleitung finden Sie unter [6].

Zunächst erzeugen Sie eine Container-Datei:

```
dd if=/dev/urandom of=priv.luks bs=1M count=250
```

Statt */dev/urandom* können Sie auch */dev/zero* verwenden, was deutlich schneller ist; dann sind die leeren Bereiche des Containers allerdings mit Nullen befüllt, woran ein Angreifer zumindest den Füllstand ablesen kann.

Sodann wird der Container an ein sogenanntes Loop-Device gebunden:

```
losetup /dev/loop0 priv.luks
```

Aus Sicht des Betriebssystems verhält sich der Container nun wie eine Festplatte, die über den Gerätenamen */dev/loop0* angesprochen wird.

Für die nächsten Schritte dient das Werkzeug *cryptsetup*:

```
cryptsetup -c aes-xts-plain -y -s 512 \ luksFormat /dev/loop0
cryptsetup luksOpen /dev/loop0 privat
```

Der erste Befehl richtet die eigentliche Verschlüsselung ein. Mit -c geben Sie den zu verwendenden Algorithmus an, mit -s die Schlüssellänge.

Der zweite Befehl erzeugt eine weitere Indirektion: das Gerät /dev/loop0 verhält sich wie eine verschlüsselte Festplatte. Durch den cryptsetup-Aufruf wird es zusätzlich in entschlüsselter Form unter dem Namen privat bereitgestellt. Der Zugriff erfolgt dann über das virtuelle Gerät /dev/mapper/privat. Dieses können wir nun wie eine normale Festplatte benutzen, d.h. ein Dateisystem anlegen und das Gerät einbinden.

```
mkfs.btrfs /dev/mapper/privat
mount /dev/mapper/privat /mnt
```

Unter /mnt finden Sie nun den Inhalt Ihres Containers. Um diesen nach einem Neustart erneut einzubinden, verwenden Sie die Zeilen

```
losetup /dev/loop0 priv.luks
cryptsetup luksOpen …
mount /dev/mapper/privat /mnt
```

genau wie oben. Zur Vereinfachung verwenden Sie am besten ein Skript wie das folgende:

```
#/bin/bash
# $1: loopdevice, e.g. /dev/loop0
# $2: container, e.g. container.tc
# $3: mapped name (used internally)
# $4: mountpoint, e.g. /mnt/container
# $5: password hint
cr_mount() {
  loopdevice=$1
  container=$2
  mapper=$3
  mountpoint=$4
  echo "mounting $container"
  echo "hint: $5"
  echo calling losetup $loopdevice $container
```

```
    losetup $loopdevice $container cryptsetup luksOpen \
      $loopdevice $mapper mount /dev/mapper/$mapper \
        $mountpoint
}
# $1 mountpoint, $2 mapped name, $3 device
cr_umount() {
  umount $1
  cryptsetup luksClose $2
  losetup -d $3
}
find_loop_device() {
  echo `losetup -f`
}
mount_privat() {
  dev=$(find_loop_device)
  cr_mount $dev $CONTAINER_BASE/privat.luks \
    privat /mnt/privat "Hint: Password Gelb"
}

mount_all() {
  mount_privat
...
}

CONTAINER_BASE=/pfad/zuden/Containern
```

Kern sind die Funktionen *cr_mount()* und *cr_unmount()*, die – mit den richtigen Parametern aufgerufen – jeweils einen Container ein- oder ausbinden. Um diese Parameter nicht jedes Mal neu angeben zu müssen, dienen Funktionen wie *mount_privat()*. Sie bindet den als „Privat“ benannten Container ein und enthält alle dazu nötigen Parameter. Als weitere Erleichterung können Sie eine Funktion *mount_all()* schreiben, die einfach die Funktionen zum Einbinden aller Ihrer Container aufruft. Um dieses Skript zu verwenden, müssen Sie es lediglich in einer (*Root*-)Shell einbinden:

```
$ source script.sh
```

Sodann können Sie ihre Container ganz einfach öffnen und schließen: Rufen Sie die passende Funktion auf, geben Sie das Passwort ein und fertig.

```
$ mount_privat
mounting /pfad/privat.luks
Hint: Passwort Gelb
calling losetup /dev/loop1 /pfad/privat.luks
Geben Sie die Passphrase für »/pfad/privat.luks« ein:
$
```

Sollten Sie sich nun fragen, wie das alles mit dem ersten Teil des Artikels zusammenhängt, so ist die Antwort ganz einfach: Das git-Repository kommt in den Container.

Fazit

Mit dem vorgestellten Ansatz ist es möglich, eine eigene Lösung zur Datenhaltung zusammenzustellen, mit der Sie – und nur Sie - von überall Zugriff auf Ihre Daten haben. Zusätzlich sind sie gleich noch gegen Festplattenschäden und verschiedene andere Schadensszenarien abgesichert.

Der Fokus lag dabei auf Linux, das generelle Vorgehen ist jedoch auf andere Betriebssysteme übertragbar. git läuft unter allen wichtigen Betriebssystemen; DynDNS-Clients sind ebenfalls überall verfügbar. Für Container-basierte Verschlüsselung verwenden Sie unter Windows z. B. VeraCrypt. Oder Sie verschlüsseln gleich die gesamte Platte mit BitLocker.

Wir freuen uns über Anregungen zur Verbesserung, insbesondere Anwendungsfälle, die bisher nicht abgedeckt sind. In einer der nächsten Ausgaben des KaffeeKlatsches werden wir verschiedene Kleincomputer vorstellen, die als Server infrage kommen.

Referenzen

[1] Bertz, Thomas *Wrtzlprmfpft – Mit GnuPG eine sichere Kommunikation aufbauen*
KaffeeKlatsch 04/2015, Jahrgang 8, Nr. 4, Seiten 7 – 10, Bookware, April 2015
www.bookware.de/kaffeeklatsch/archiv/KaffeeKlatsch-2015-04.pdf

[2] git *gittutorial – A tutorial introduction to Git*
http://git-scm.com/docs/gittutorial

[3] Wikipedia *Dynamisches DNS*
https://de.wikipedia.org/wiki/Dynamisches_DNS

[4] Pfligl, Konstantin
Fernzugriff aufs Heimnetz – Kostenlose DynDNS-Dienste im Überblick
http://www.com-magazin.de/praxis/netzwerk/kostenlose-dyndns-dienste-im-ueberblick-215893.html

[5] Wikipedia *TrueCrypt*
https://de.wikipedia.org/wiki/TrueCrypt

[6] Ubuntuusers *Containerdatei*
https://wiki.ubuntuusers.de/LUKS/Containerdatei

Kurzbiografie

Dr. Frank Lauterwald arbeitet als Senior Consultant für MATHEMA Software GmbH. Seit seinem elften Lebensjahr programmiert er (fast) alles, was ihm vor die Tastatur kommt. Aktuell beschäftigt er sich mit JEE-Architekturen. Sein besonderes Interesse gilt dabei der Datenhaltung.

Des Programmieres kleine Vergnügen

Bedingte Bit-Setzerei (exklusiv verbessert)

von Michael Wiedeking

Die letzte Kolumne bot eine ziemlich effiziente Methode, um Bits (ohne ein *if* zu verwenden) bedingt zu setzen oder zu löschen. Dort wurde auch behauptet, dass jene fünf Instruktionen noch zu überbieten wären. Was hier geschehen soll.

Vorgegeben war ein Wort x, in dem ein oder mehrere Bits aus einer Maske m gesetzt oder gelöscht werden sollen. Mit dem Wert einer Bedingung b (mit 1 = *true* und 0 = *false*) sollte das Problem

```
if (b) {
   x = x | m; // Bits setzen
} else {
   x = x & ~m; // Bits löschen
}
```

möglichst effizient gelöst werden, was schließlich mit den beeindruckenden fünf Instruktionen

```
x = (m | (-b)) | (x & ~m);
```

bewerkstelligt werden konnte. Der Clou war hier, dass man das Bit immer löscht (x & ~m) und nur dann setzt, wenn es nötig ist. Nach dieser Optimierung ist aber nicht mehr zu erwarten, dass man dieses Ergebnis noch weiter verbessern kann. So ist es eben manchmal vonnöten, alles erneut zu überdenken und ganz von Vorne zu beginnen.

Das Problem ist also, ein Bit zu setzen oder zu löschen. Will man beispielsweise ein Bit setzen, so ist das aber gleichbedeutend damit, dass man es setzt, wenn es noch nicht gesetzt ist und unangetastet lässt, wenn es schon gesetzt ist. Umgekehrt vertauscht man zum Löschen ein gesetztes Bit und lässt das ungesetzte unangetastet. Und womit tauscht man am besten?

Na klar, mit einem Exklusiv-Oder. Exklusiv-odert man etwas mit 0, so bleibt es unverändert; Exklusiv-odert man es mit 1, so wird es getauscht. Ist die Bedingung 1, so sollen also die Bits aus m gesetzt werden. $x \wedge m$ alleine tauscht leider alle Bits, die in m gesetzt sind, aus. Dies soll aber nur dann passieren, wenn die korrespondierenden Bits in x ungesetzt sind. Mit anderen Worten: Wir benötigen eine Maske m', bei der nur diejenigen Bits ungesetzt sind, die getauscht werden sollen.

x & m liefert eine Maske, bei der alle Bits gesetzt sind, die auch in x gesetzt sind. Das benötigt man im Fall $b = 1$ aber genau umgekehrt. Das ist einfach hinzubekommen, indem man (wie so oft) mit $-b$ eine Maske erzeugt, die nur aus Einsen besteht. Wird diese nun vorher mit x exklusiv-verodert, erhält man das gewünschte Ergebnis:

$$m' = (-b \wedge x) \,\&\, m$$

Jetzt sind in m' nur noch alle die Bits gesetzt, die laut Maske m gesetzt werden müssen und noch nicht in x gesetzt sind. Wird dieses m' nun mit x exklusiv-verodert, werden die gewünschten Bits aus x getauscht.

$$x = x \wedge m' = x \wedge ((-b \wedge x) \,\&\, m)$$

Beispielsweise soll x das Bit-Muster $(0101)_2$ haben. Nun möchte man die niederwertigsten zwei Bits $m = (0011)_2$ setzten. Mit $b = 1$ ergibt sich

$$\begin{array}{rr} b = 0001 & -b \wedge x = 1010 \\ -b = 1111 & m = 0011 \\ x = 0101 & (-b \wedge x) \,\&\, m = 0010 \end{array}$$

$$x \wedge ((-b \wedge x) \,\&\, m) = 0101 \wedge 0010 = 0111$$

Und das Schöne daran ist, dass das auch für das Löschen gilt, wenn $b = 0$ ist. Denn dann ist auch $-b = 0$, und $-b \wedge x$ lässt x unverändert. Die Verundung mit m löscht dann alle in x nicht gesetzten Bits auch in m. Und damit werden abschließend nur noch genau die Bits aus x getauscht, die in m gesetzt sind. Voilà.

$$\begin{array}{rr} b = 0000 & -b \wedge x = 0101 \\ -b = 0000 & m = 0011 \\ x = 0101 & (-b \wedge x) \,\&\, m = 0001 \end{array}$$

$$x \wedge ((-b \wedge x) \,\&\, m) = 0101 \wedge 0001 = 0100$$

Somit werden nunmehr vier Instruktionen gebraucht. Und besser geht's nicht.

User Groups

Fehlt eine User Group? Sind Kontaktdaten falsch? Dann geben Sie uns doch bitte Bescheid.

Bookware, Henkestraße 91, 91052 Erlangen
Telefon: 0 91 31 / 89 03-0, Telefax: 0 91 31 / 89 03-55
E-Mail: redaktion@bookware.de

Java User Groups

Deutschland

JUG Berlin Brandenburg
http://www.jug-bb.de
Kontakt: Herr Ralph Bergmann (orga@jug-bb.de)

Java UserGroup Bremen
http://www.jugbremen.de
Kontakt: Rabea Gransberger (rgransberger@gmx.de)

JUG DA
Java User Group Darmstadt
http://www.jug-da.de
Kontakt: jug-da-orga@googlegroups.com

Java User Group Saxony
Java User Group Dresden
http://www.jugsaxony.de
Kontakt: Herr Falk Hartmann
(falk.hartmann@jugsaxony.org)

rheinjug e.V.
Java User Group Düsseldorf
Heinrich-Heine-Universität Düsseldorf
http://www.rheinjug.de
Kontakt: Herr Heiko Sippel (info@rheinjug.de)

ruhrjug
Java User Group Essen
Glaspavillon Uni-Campus
http://www.ruhrjug.de
Kontakt: Herr Heiko Sippel (heiko.sippel@ruhrjug.de)

JUGF
Java User Group Frankfurt
http://www.jugf.de
Kontakt: Herr Alexander Culum
(alexander.culum@web.de)

JUG Deutschland e.V.
Java User Group Deutschland e.V.
c/o Stefan Koospal
http://www.java.de (office@java.de)

JUG Hamburg
Java User Group Hamburg
http://www.jughh.org

JUG Karlsruhe
Java User Group Karlsruhe
http://jug-karlsruhe.de
(jugkarlsruhe@gmail.com)

JUGC
Java User Group Köln
http://www.jugcologne.org
Kontakt: Herr Michael Hüttermann
(michael@huettermann.net)

jugm
Java User Group München
http://www.jugm.de
Kontakt: Herr Andreas Haug (ah@jugm.de)

JUG Münster
Java User Group für Münster und das Münsterland
http://www.jug-muenster.de
Kontakt: Herr Thomas Kruse (tkjugi@sforce.org)

JUG MeNue
Java User Group der Metropolregion Nürnberg
c/o MATHEMA Software GmbH
Henkestraße 91, 91052 Erlangen
http://www.jug-n.de
Kontakt: Frau Natalia Wilhelm
(info@jug-n.de)

JUG Ostfalen
Java User Group Ostfalen
(Braunschweig, Wolfsburg, Hannover)
http://www.jug-ostfalen.de
Kontakt: Uwe Sauerbrei (info@jug-ostfalen.de)

JUGS e.V.
Java User Group Stuttgart e.V.
c/o Dr. Michael Paus
http://www.jugs.org
Kontakt: Herr Dr. Micheal Paus (mp@jugs.org)
Herr Hagen Stanek (hs@jugs.org)
Rainer Anglett (ra@jugs.org)

Schweiz

JUGS
Java User Group Switzerland
http://www.jugs.ch (info@jugs.ch)

.NET User Groups

Deutschland

.NET User Group Bonn
.NET User Group "Bonn-to-Code.Net"
http://www.bonn-to-code.net (mail@bonn-to-code.net)
Kontakt: Herr Roland Weigelt

.NET User Group Dortmund (Do.NET)
c/o BROCKHAUS AG
http://do-dotnet.de
Kontakt: Paul Mizel (pmizel@do-dotnet.de)

Die Dodnedder
.NET User Group Franken
http://www.dodnedder.de
Kontakt: Herr Udo Neßhöver, Frau Ulrike Stirnweiß
(info@dodnedder.de)

.NET UserGroup Frankfurt
http://www.dotnet-usergroup.de

.NET User Group Friedrichshafen
http://www.dotnet-fn.de
Kontakt: Tobias Allweier
(info@dotnet-fn.de)

.NET User Group Hannover
http://www.dnug-hannover.de
Kontakt: (dnug@indisoftware.de)

INdotNET
Ingolstädter .NET Developers Group
http://www.indot.net
Kontakt: Herr Gregor Biswanger
(gregor.biswanger@web-enliven.de)

DNUG-Köln
DotNetUserGroup Köln
http://www.dnug-koeln.de
Kontakt: Herr Albert Weinert (info@der-albert.com)

.NET User Group Leipzig
http://www.dotnet-leipzig.de
Kontakt: Herr Alexander Groß (agross@dotnet-leipzig.de)
Herr Torsten Weber (tweber@dotnet-leipzig.de)

.NET Developers Group München
http://www.munichdot.net
Kontakt: Hardy Erlinger (hardy_erlinger@hotmail.com)

.NET User Group Oldenburg
c/o Hilmar Bunjes und Yvette Teiken
http://www.dotnet-oldenburg.de
Kontakt: Herr Hilmar Bunjes
(hilmar.bunjes@dotnet-oldenburg.de)
Frau Yvette Teiken (yvette.teiken@dotnet-oldenburg.de)

.NET Developers Group Stuttgart
http://www.devgroup-stuttgart.net
(GroupLeader@devgroup-stuttgart.net)
Kontakt: Herr Michael Niethammer

.NET Developer-Group Ulm
c/o artiso solutions GmbH
http://www.dotnet-ulm.de
Kontakt: Herr Thomas Schissler (tschissler@artiso.com)

Österreich

.NET User Group Austria
c/o Global Knowledge Network GmbH,
http://usergroups.at/blogs/dotnetusergroupaustria/default.aspx
Kontakt: Herr Christian Nagel (ug@christiannagel.com)

Software Craftsmanship Communities

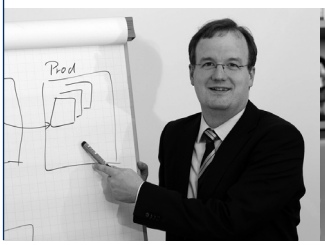
Deutschland, Schweiz, Österreich

Softwerkskammer – Mehrere regionale Gruppen und Themengruppen unter einem Dach
http://www.softwerkskammer.org
Kontakt: Nicole Rauch (nicole.m@gmx.de)

MATHEMA

meet the experts of enterprise infrastructure

- **Scrum Basics**
  2. – 3. Mai 2016, 950,– € (zzgl. 19 % MwSt.)
- **Scrum im Großen**
  Agiles Organisationsdesign nach LeSS
  10. – 11. Mai 2016, 950,– € (zzgl. 19 % MwSt.)
- **Java FX8**
  30. Mai – 1. Juni 2016, 14. – 16. November 2016
  1.250,– € (zzgl. 19 % MwSt.)
- **Generation 8 – Java Update**
  6. – 7. Juni 2016, 21. – 22. November 2016
  950,– € (zzgl. 19 % MwSt.)
- **AngularJS**
  13. – 14. Juni 2016, 28. – 29. November 2016
  950,– € (zzgl. 19 % MwSt.)
- **Entwicklung mobiler Anwendungen mit iOS**
  18. – 20. Juli 2016, 1.250,– € (zzgl. 19 % MwSt.)
- **HTML5, CSS3 und JavaScript**
  27. – 30. Juni 2016, 1.650,– € (zzgl. 19 % MwSt.)




Lesen bildet. **Training macht fit.**

MATHEMA Software GmbH | Telefon: 09131 / 89 03-0 | Internet: www.mathema.de
Henkestraße 91, 91052 Erlangen | Telefax: 09131 / 89 03-55 | E-Mail: info@mathema.de

MATHEMA

„Die Herausforderung, jeden Tag etwas Neues zu lernen, habe ich gesucht und bei MATHEMA gefunden."

Tim Bourguignon, Senior Consultant

Wir sind ein Consulting-Unternehmen mit Schwerpunkt in der Entwicklung unternehmenskritischer, verteilter Systeme und Umsetzung von Service-orientierten Architekturen und Applikationen von Frontend bis Backend. Darüber hinaus ist uns der Wissenstransfer ein großes Anliegen: Wir verfügen über einen eigenen Trainingsbereich und unsere Consultants sind regelmäßig als Autoren in der Fachpresse sowie als Speaker auf zahlreichen Fachkonferenzen präsent.

join the experts of enterprise infrastructure

MATHEMA Software GmbH | Telefon: 09131 / 89 03-0 | Internet: www.mathema.de | Ansprechpartnerin:
Henkestraße 91, 91052 Erlangen | Telefax: 09131 / 89 03-55 | E-Mail: info@mathema.de | Britta Kolb

Herbstcampus

Wissenstransfer par excellence

30. August – 1. September 2016, Nürnberg

Beitragsaufruf

(Call for Papers)

Der Herbstcampus ist eine technologie-orientierte Konferenz, die Software-Entwicklern, Architekten und Projektleitern eine konkrete Hilfe bieten soll. Schwerpunkte sind .NET und Java.

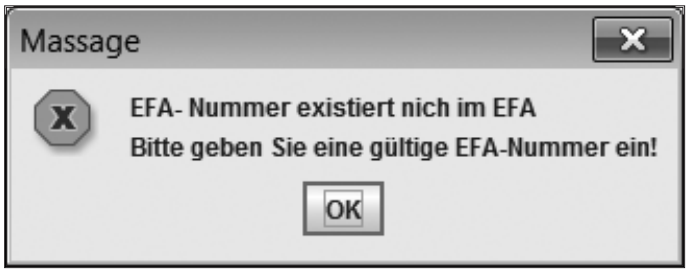
Interessante Themen sollen in verschiedenen Darbietungen behandelt werden: als Einführung, Erfahrungsbericht oder problemlösender Vortrag.

Haben Sie ein passendes Thema oder Interesse daran, einen Vortrag zu halten? Schicken Sie uns Ihre Vorschläge: beitragsaufruf@herbstcampus.de

Haben Sie noch Fragen? Dann schreiben Sie uns eine E-Mail. info@herbstcampus.de

Den vollständigen Beitragsaufruf und weitere Informationen zum Herbstcampus finden Sie unter www.herbstcampus.de

Das Allerletzte



Dies ist kein Scherz!
Diese Meldung wurde tatsächlich in der freien
Wildbahn angetroffen.

Ist Ihnen auch schon einmal ein Exemplar dieser
Gattung über den Weg gelaufen?
Dann scheuen Sie sich bitte nicht, uns das mitzuteilen.

Der nächste KaffeeKlatsch erscheint im Mai.

Herbstcampus

Wissenstransfer par excellence

30. August – 1. September 2016
in Nürnberg