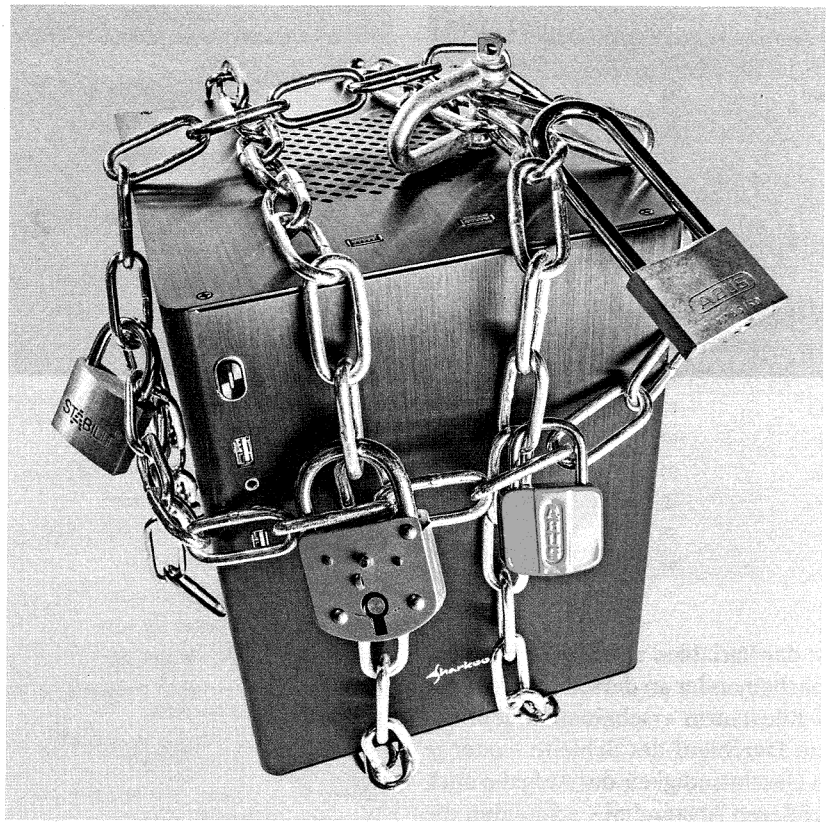


My Home-Server is my castle

Heim-Server verschlüsseln, ohne sich bei Stromausfällen selbst auszusperren

Klaut ein Einbrecher den Heim-Server, soll er wenigstens nicht an die Daten herankommen. Dazu kann man sie verschlüsseln – doch dann bootet der Server nicht mehr ohne manuellen Eingriff. Ein verschlüsselter lxc-Container zerschlägt den Gordischen Knoten: Mit dem folgenden Setup startet man den Heim-Server nach einem Stromausfall auch aus der Ferne.

Von Johannes Merkert

Sie kommen in der Nacht. Dunkle Gestalten verschaffen sich Zugang zu meiner Wohnung, durchsuchen meine Privaträume, nehmen sich, was mir gehört. Darunter mein Heim-Server mit allen Dokumenten und Urlaubsbildern. Auf der kleinen Kiste im Wohnzimmer war sogar das Liebeslied für meine Jugendliebe gespeichert. – Eine Horrorvorstellung! Die Lösung besteht darin, die Daten zu verschlüsseln. In der Installationsanleitung für Linux [1] hatten wir zwar auf Verschlüsselung verwiesen. Allerdings wären dann nur die Daten unter /srv auf der verschlüsselten Partition sicher. Legt ein Dienst seine Daten an anderer Stelle ab, bleiben diese für den Dieb lesbar.

Stattdessen können Sie bei der Installation gleich das ganze System verschlüsseln. Dann fragt Ubuntu beim Start nach dem Passwort und fährt ohne erst gar nicht hoch. Nach einem Neustart wegen Stromausfalls im Urlaub fällt Ihnen diese Lösung auf die Füße: Der Server hängt dann an der Passworteingabe fest und führt weder SSH noch die OwnCloud aus, auf der die Urlaubsbilder eigentlich landen sollten.

Die verstrickte Situation entwirren zwei Linuxe. Ein unverschlüsseltes Linux dient als Host-System, das ohne Hilfe bootet und SSH ausführt. Meldet man sich lokal oder über SSH auf dem System an, steht ein Skript bereit, das die verschlüsselte Datenplatte entsperrt. Als Verschlüsselung kommt das in Linux integrierte Duo aus dm-crypt und LUKS zum Einsatz, das standardmäßig mit AES-128 verschlüsselt. Das dafür nötige Passwort kennt kein Dieb. Auf der verschlüsselten Platte liegt das zweite System in einem Linux-Container. Dieser nutzt den Kernel des Hosts, bringt aber sämtliche Programme und Bibliotheken in seinem eigenen abgeschotteten Bereich mit. Wirft ein Dienst Daten in diesem System ab, landen sie im Container und damit auf der verschlüsselten Platte.

Container einrichten

Die Inbetriebnahme eines solchen Systems geht ganz leicht: Installieren Sie als Host-System Ubuntu 16.04, binden die Datenplatte aber noch nicht während der Installation ins Dateisystem ein. Mit `gnome-disks` („Laufwerke“) formatieren Sie nach der Installation die Platte mit ext4 und aktivieren die Verschlüsselung mit LUKS. Im Auswahlmenü gibt es einen Eintrag für diese Kombination. Legen Sie ein Passwort fest, das kein Einbrecher errät. Unser Entschlüsselungsskript (siehe Listing rechts) wird die Platte später ins System einhängen, sodass Sie keinen Mountpoint angeben müssen.

Das Skript entsperrt die Partition mit `cryptsetup luksOpen`, was unter dem angegebenen Namen ein virtuelles Device in /dev/mapper erzeugt. Anschließend bindet ein normales `mount` dieses Device an der Stelle /srv ins Dateisystem ein. Führen Sie die Befehle aus Zeile 2 und 3 des Skripts jetzt manuell aus, da die Daten des

Containers gleich im Verzeichnis /srv/container unterkommen. Damit das klappt, müssen Sie es anlegen:

```
sudo mkdir /srv/container
```

Um die Verwaltung von Linux-Containern kümmert sich der Container-Manager lxc, den Sie mit apt installieren:

```
sudo apt install lxc debootstrap
sudo apt install bridge-utils
```

lxc speichert Container standardmäßig unter /var/lib/lxc. Der Bind-Mount in Zeile 5 des Skripts sorgt dafür, dass dieses Verzeichnis auf der verschlüsselten Partition liegt. Führen Sie auch diesen Befehl gleich aus.

lxc bringt Profile für die häufigsten Linux-Distributionen mit. Der folgende Befehl lädt ein komplettes Ubuntu-Image herunter, was eine Weile dauert:

```
sudo lxc-create -n encrypted-system \
-t ubuntu
```

Danach liegt ein startbereiter Container namens „encrypted-system“ mit dem zweiten Ubuntu 16.04 auf der verschlüsselten Platte. `lxc-start` fährt ihn hoch (Zeile 7).

Nach dem Start kommen Sie direkt über lxc auf das System im Container:

```
sudo lxc-console -n encrypted-system
```

Welche IP lxc dem Containersystem gibt, erfahren Sie, indem Sie im Container `ip addr` ausführen. Mit diesem Wissen gelangen Sie auch über das virtuelle Netzwerk-Interface und SSH an das System (`ssh ubuntu@<IP-Adresse>`). Das Ubuntu im Container richtet automatisch einen Benutzer „ubuntu“ mit dem Passwort „ubuntu“ ein. Das Passwort können Sie mit `passwd` ändern, sobald Sie dort angemeldet sind.

Je nach Dienst soll entweder das Host-System oder das Linux im Container die Anfragen beantworten. Damit das klappt, richten Sie eine Portweiterleitung für alle Ports des Web-, Medien- und Samba-Servers ein. Das Skript nutzt dafür zwei `for`-Schleifen: Zeile 10 bis 14 für TCP und 15 bis 18 für UDP. Wenn Sie weitere Ports weiterleiten möchten, erweitern Sie einfach die Liste.

```
 1 #!/bin/bash
 2 cryptsetup luksOpen /dev/sdb1 luks-srv
 3 mount /dev/mapper/luks-srv /srv
 4 # Mount container storage to /srv
 5 mount -o bind /srv/container/ /var/lib/lxc/
 6 # Start the encrypted container
 7 lxc-start -d -n encrypted-system
 8 # Set up port forwarding to the container
 9 NETWORK_INTERFACE="enp1s0"
10 for PORT in 80 443 53 88 135 137 138 139 389 445 464 636 \
11 1024 1025 1026 1027 3268 3269 5353 8200
12 do iptables -t nat -A PREROUTING -i $NETWORK_INTERFACE -p tcp \
13 --dport $PORT -j DNAT --to 10.0.3.21:$PORT
14 done
15 for PORT in 1900
16 do iptables -t nat -A PREROUTING -i $NETWORK_INTERFACE -p udp \
17 --dport $PORT -j DNAT --to 10.0.3.21:$PORT
18 done
```

Das Entschlüsselungsskript im Homeverzeichnis fragt nach dem Passwort, entsperrt die verschlüsselte Datenplatte, startet den Container mit dem zweiten Linux und richtet die Portweiterleitungen ein.

```
#!/bin/bash
echo -n "Password:"
read -s password
echo
echo "$password
$password" | adduser "$1" --home /srv/samba-private/$1 --gecos ""
echo "$password
$password" | smbpasswd -a -s $1
```

adduser erstellt Benutzer-Accounts auf der Konsole; smbpasswd setzt das zugehörige Passwort in der Benutzerverwaltung von Samba. Mit diesem Skript erspart man es sich, das Passwort mehrfach einzugeben.

Installation im Container

Im Container installieren Sie Samba, Apache2 und Minidlna, wie in [1] beschrieben. Abgesehen davon, dass Ihnen im Container nur die Textkonsole zur Verfügung steht, ändert sich nicht viel. Lediglich die neue Ubuntu-Version 16.04 macht ein paar Schritte komplizierter.

So verschwand mit 16.04 die `libpam-smbpass` aus den Paketquellen. Die Bibliothek hatte die System- automatisch mit den Samba-Benutzern synchronisiert. Unser kleines Skript `add_samba_user` reduziert die Tipparbeit aber dennoch auf ein Minimum (siehe Listing unten). Es akzeptiert als ersten Parameter den Benutzernamen und fragt dann interaktiv nach dem Passwort.

Außerdem wechselte Ubuntu 16.04 von PHP 5 auf Version 7, das jetzt folgende Pakete für den Webserver braucht:

```
sudo apt install wget apache2 \
libapache2-mod-php php-mysql \
mysql-server phpmyadmin php-gettext
```

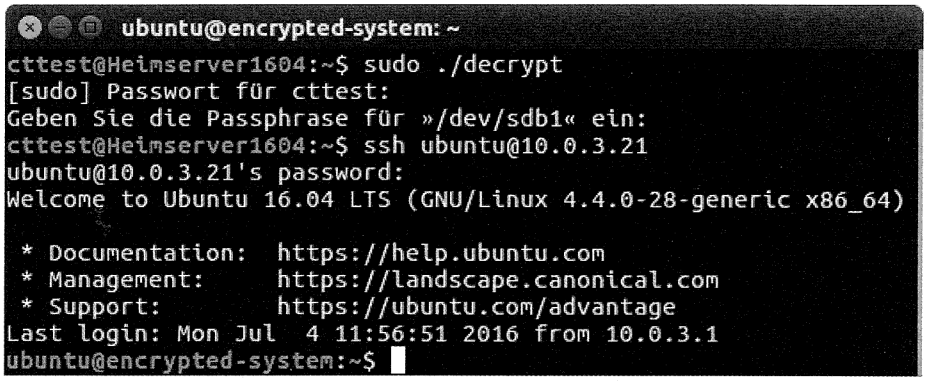
Container stoppen

lxc stoppt den Container nicht automatisch, wenn das Host-System herunter fährt. Da man ihn aus dem Host-System mit

```
lxc-stop -n encrypted-system
```

beenden kann, muss man es aber nicht zur Handarbeit kommen lassen. Eine Systemd-Unit führt den Befehl automatisch beim Herunterfahren aus. Aktivieren Sie die Unit mit

```
sudo systemctl enable \
encrypted-system.service
```



Dank des Entschlüsselungsskripts fährt der Container mit nur einem Befehl hoch. Seine Dienste konfiguriert man bequem über ssh.

```
[Unit]
Description=Mail the admin on boot and stop the container on shutdown
After=multi-user.target

[Service]
Type=oneshot
RemainAfterExit=true
ExecStart=/home/hugo/send-booted-mail.py
ExecStop=lxc-stop -n encrypted-system

[Install]
WantedBy=multi-user.target
```

Dank der Unit /etc/systemd/system/encrypted-system.service startet Systemd beim Start ein Skript, das eine Mail schreibt, und fährt beim Beenden den Container herunter.

```
#!/usr/bin/python3
import smtplib
import time
import os
time.sleep(60)
if not os.path.isfile("/srv/decrypted"):
  from_addr = "jme.ct.de@gmail.com"
  to_addr = "jme@ct.de"
  google_username = "jme.ct.de@gmail.com"
  google_app_password = "xxxxxxxxxxxxxxxx"
  message = "\r\n".join(["From: " + from_addr,
    "To: " + to_addr,
    "Subject: Der Heim-Server wurde gerade neu gestartet", "",
    "Bitte logge dich per SSH ein und entsperre die " +
    "verschluesselte Partition.", "",
    "ssh -p 22000 cttest@heimcloud.dynv6.net", "",
    "Entsperre die Partition mit:", "sudo ./decrypt"
  ])
  server = smtplib.SMTP('smtp.gmail.com:587')
  server.ehlo()
  server.starttls()
  server.login(google_username, google_app_password)
  server.sendmail(from_addr, to_addr, message)
  server.quit()
```

Das Skript send-booted-mail.py verbindet sich mit dem SMTP-Server von Gmail und verschickt eine Mail mit Hinweisen zum Entsperren des Servers.

Mail beim Start

Der Server lässt sich mit dem Entschlüsselungsskript zwar leicht starten, bei einem ungeplanten Neustart, beispielsweise nach einem Stromausfall, würde man aber nichts davon erfahren. Ein paar Zeilen Python verbinden sich mit einem SMTP-Server und schicken eine Mail mit Hinweisen zum Entschlüsseln. Kommt die Mail an der im Smartphone eingerichteten Gmail-Adresse an, erfahren Sie auch im Urlaub vom Neustart. Mit einer SSH-App wie ConnectBot bauen Sie dann unterwegs schnell eine SSH-Verbindung auf und starten das Entschlüsselungsskript auf dem Server.

Das Python-Programm wartet zuerst eine Minute und prüft dann, ob die Partition schon entschlüsselt wurde. So vermeidet es unnötige Mails, wenn man den Server per Hand neu startet und direkt entsperrt. Für diesen Test sucht es im Ordner /srv eine Datei Namens „decrypted“. Damit er funktioniert, müssen Sie diese Datei bei entschlüsselter Partition anlegen:

```
sudo touch /srv/decrypted
```

Das abgedruckte Skript verbindet sich per STARTSSL mit Gmail. Bei aktivierter Zwei-Faktor-Authentifizierung müssen Sie in Ihren Gmail-Einstellungen vorher ein App-Passwort anlegen, das Sie im Skript verwenden können. Bei anderen SMTP-Servern genügt in der Regel das Passwort zum Mail-Account.

Um das Skript bei jedem Boot automatisch aufzurufen, tragen Sie es hinter `ExecStart=` in die Systemd-Unit-Konfiguration ein. Außerdem müssen Sie es noch ausführbar machen:

```
chmod 775 \
/home/hugo/send-booted-mail.py
```

Mit dieser Konfiguration sagt Ihnen Ihr Heim-Server per Mail Bescheid, wenn er startet, und Sie können von überall Ihr Passwort eingeben. Die Daten im Container bleiben dabei sicher in der verschlüsselten Partition verwahrt. Solange Sie keine sensiblen Daten im Host-System speichern, können Sie dann ganz entspannt in Urlaub fahren. *(jme@ct.de)* c't

Literatur

[1] Johannes Merkert, Anke Poimann, Selbstbau-System, Heim-Server mit Linux installieren, c't 8/16, S. 106