öffentliche IP-Adresse Datenlogs -> Webserver

Wiki

Sicher gibt es einen saubereren Weg. Tips dazu dürfen gerne hinterlassen werden. ToDo: IP auf dem Webserver nur aktualisieren, wenn sich diese wirklich geändert hat, dafür Rate erhöhen.

Problem: Ein Rechner (bei mir ein RaspberryPI), den ich von außerhalb erreichen will, befindet sich hinter einem Router. Meine IP wechselt täglich. Dadurch muss ich diese täglich neu ermitteln lassen, wenn ich diesen Rechner von außen erreichen möchte. Hinter einem Router kann ich die öffentliche IP-Adresse per Shell nicht ohne weiteres Zutun von außen herausfinden. Einen Dienst wie DynDNS möchte ich nicht verwenden. Die auf diesem Rechner gesammelten Daten sollen später regelmäßig auf einem schneller angebundenen Webserver bereit gestellt werden. Das ist auch sinnvoll, wenn der Rechner „in der Wildnis“ per UMTS angebunden ist und dadurch selbst gar nicht erreichbar ist.

Vorgehensweise: Per PHP-Script auf einem Webserver soll die öffentlichte IP-Adresse des Zielrechners dargestellt werden. Diese wird auf dem Ziel-Rechner abgelegt und auf einem Webserver mit fester Adresse hochgeladen. Dabei sollen auch andere Dateien mit übertragen werden. Dies soll beim Start und in regelmäßigen Abständen passieren.

Zielrechner-IP → ruft seine öffentl. IP-Adresse vom Webserver ab → legt diese auf dem Webserver ab → Mit anderem Terminalrechner IP-Adresse (und Datenlogs) des Zielrechners vom Webserver abrufen → Terminalrechner kann sich nun direkt auf die nun ihm bekannte öffentliche IP des Zielrechners verbinden

„Zielrechner“ ist bei mir ein Raspberry, auf dessen Shell ich von überall zugreifen möchte. „Terminalrechner“ sind verschiedene Rechner, mit denen ich auf den Raspberry zugreifen möchte.

Voraussetzungen:

- Einen per SSH erreichbaren Webserver mit PHP (gibt einige günstige, sonst mal bei uberspace.de vorbeischauen)
- Portweiterleitungen innerhalb des Netzwerks auf den Rechner, der erreichbar sein soll (im Router feste IP festlegen, Port 22 für SSH weiterleiten).

oder eine öffentliche IP-Adresse/Direktanbindung am Anschluss.

- Eine Shell, die mit dem Rechner im lokalen Netz (z.B. Raspberry) verbunden ist.

1. Vorbereitung auf dem Webserver

Eine Datei mit dem Namen ip.php anlegen und editieren:

```
nano ip.php
```

Diese Zeilen einfügen:

```
   <?php
      echo $_SERVER["REMOTE_ADDR"]
   ?>
```

Die Datei dann speichern und schließen (STRG+O → [Return] → STRG-X)

Diese Datei dann auf dem gewünschten Ort (z.B. raspip) auf dem Webserver ablegen:

```
sftp USER@SERVERNAME
(Kennwort)
cd html
mkdir raspip
cd raspip
put ip.php
exit
```

oder wenn das Verzeichnis ~/html/raspip auf dem Webserver schon vorhanden ist:

```
scp ip.php USER@SERVERNAME:~/html/raspip/ip.php
```

Mit dem Aufruf von http://SERVERNAME/raspip/ip.php erhält man nun seine öffentliche IP zurück. Zum Beispiel:

```
curl http://aetherfoton.de/raspip/ip.php
85.183.148.169
```

Die Ausgabe lässt sich natürlich in eine Datei umleiten (hier index.html - der Name ist uns später nützlich ;) :

```
curl http://aetherfoton.de/raspip/ip.php > index.html
```

(Die Adresse in beiden Beispielen funktioniert. Die könnt Ihr auch nutzen, aber ich gebe keine Garantie auf dauerhafte Funktion ;)

Aktuell bietet auch ein Hackerspace-Webserver diese Funktion - siehe API virtueller Webserver.

2. Einrichten der SSH-Verbindung (auch für SFTP oder SCP) vom Zielrechner zum Webserver ohne manuelle Passworteingabe

Hat man einen ssh-key erzeugt und den auf dem Webserver hinterlegt, wird vom Webserver das Passwort nicht manuell abgefragt. Den ssh-key erzeugt man (auf dem Zielrechner) mit

```
ssh-keygen -t dsa
```

Ich habe die Standardvorgabe nicht geändert und keine Passphrase verwendet.

In ~/.ssh liegen nun die Dateien „id_dsa“(privater Schlüssel) und „id_dsa.pub“(öffentlicher Schlüssel). Der öffentliche Schlüssel in „id_dsa.pub“ muss nun beim Webserver eingetragen werden.

Bei manchen Anbietern kann man das per Webinterface machen, bei anderen (z.B. Uberspace.de) muss man diesen zu „~/.ssh/authorized_keys“ hinzufügen. Ich hab mir das wieder ganz einfach gemacht (auf dem Zielrechner):

```
cat ~/.ssh/id_dsa.pub
```

Als Ausgabe wird ssh-dss 1234….ganz…viele…zeichen……. angezeigt Diese habe ich aus der Shell herauskopiert und auf dem Webserver der Datei ~/.ssh/authorized_keys in der letzten Zeile hinzugefügt.

Meldet man sich nun per ssh oder sftp auf dem Webserver an, wird der ssh-key in ~/.ssh verwendet - eine manuelle Eingabe ist nun auch nicht mehr nötig.

3. Vorbereiten (auf dem Zielrechner) des automatischen Uploads

Die Datei laden wir mit SCP zum Webserver *(Danke an flaushy für den Tip, SCP statt SFTP zu verwenden)*

Das Senden von index.html (und evtl. weitere Dateien) an den Webserver macht man nun mit:

```
scp index.html USER@SERVERNAME:~/html/raspip/index.html
```

ggf. müsst Ihr den Pfad noch anpassen. '~' kürzt dabei '/home/USER' ab. Auf die gleiche Weise könnt Ihr noch andere Dateien (z.B. Sensorlogs) auf den Webserver laden.

Das Abrufen der öffentlichen IP-Adresse und das Speichern auf dem Webserver habe ich im Script ip-update.sh hinterlegt.

```
nano ip-update.sh
```

Der Inhalt lautet (wie oben beschrieben):

```
#!/bin/sh
curl http://SERVERNAME/raspip/ip.php > index.html
scp index.html USER@SERVERNAME:~/html/raspip/index.html
cat index.html
rm index.html
```

Die letzten beiden Befehle zeigen nochmal den Inhalt (die akt. IP-Adr.) und löschen danach die angelegte Datei.

Nach dem Speichern muss diese Datei noch ausführbar gemacht werden:

```
chmod a+x ip-update.sh
```

Ein Aufruf von ./ip-update sollte nun ungefähr so aussehen:

```
./ip-update.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100    15    0    15    0     0     47      0 --:--:-- --:--:-- --:--:--
72
index.html                                     100%   15     0.0KB/s   00:00
85.183.148.169
```

Die Eingabe von „curl http://SERVERNAME/raspip/index.html“ gibt uns nun auf jedem anderen Rechner die aktuelle öffentliche IP-Adresse des Zielrechners zurück. Durch die Endung .html können wir die Datei auch im Browser anzeigen lassen. Durch den Namen „index.html“ brauchen wir (je nach Konfiguration des Webservers) auch nur http://SERVERNAME/raspip/ anzugeben. „curl http://SERVERNAME/raspip/“ funktioniert also in den meisten Fällen genauso ;)

4.Anlegen eines Cronjobs (auf dem Zielrechner)

Nun müssen wir einen cronjob anlegen, damit das Script regelmäßig ausgeführt wird. - Einmal stündlich sollte reichen, nachts benötige ich die Updates nicht. Dazu wird die benutzereigene crontab editiert:

```
crontab -e
```

Dadurch wird ein Texteditor geöffnet, mit dem man nun die Daten eintragen kann. Eine kleine Einführung ist dort kommentiert, näheres findet man dazu z.B. auch auf http://de.wikipedia.org/wiki/Cron Will man nun einen Cronjob anlegen, der täglich zu jeder vollen Stunde die ip-update.sh (in /home/user) ausführt, fügt man in die letzte Zeile folgendes ein:

```
0 * * * * ~/ip-update.sh
```

Das könnt Ihr entsprechend anpassen, wobei das Schema „Minute(0-59) Stunde(0-23) Kalendertag(1-31) Monat(1-12) Wochentag(So=0-Sa=6) Kommando“ ist. Vielleicht wollt Ihr die Updates täglich zu jeder vollen Stunde zwischen 9 und 23 Uhr haben, dann sieht das folgendermaßen aus:

```
0 9-23 * * * ~/ip-update.sh
```

Aufzählungen sind mit ',' möglich, Bereiche können mit '-' definiert werden. Es gibt noch weitere Möglichkeiten, aber im Netz ist alles leicht dazu zu finden. Man kann das natürlich auch sinnvoll kombinieren. Will ich z.B. den Rechner ohnehin Mo-Fr nur nach Feierabend und am WE auch tagsüber erreichen können, wäre ein sinnvoller Eintrag z.B.

```
0 17-23 * * 1-5 ~/ip-update.sh
0 10-23 * * 0,6 ~/ip-update.sh
```

Urlaub und Feiertage werden natürlich nicht berücksichtigt ;) Probiert es aus und tragt ein, was Euch sinnvoll erscheint.

Die Datei dann speichern und schließen (STRG+O → [Return] → STRG-X), Crontab übernimmt den Rest.

Bitte bedenkt, dass es bei cron die Tücke gibt, dass nicht überprüft wird, ob ein voran gegangener Job bereits erledigt ist. Führt Ihr also alle 5min einen Upload aus, der z.B. 10 min dauert (wg. großer Dateien oder langsamer Verbindung), werden mehrere Jobs parallel ausgeführt.

5. Ausführung bei Systemstart (als root, auf dem Zielrechner)

Dazu müssen wir erstmal dafür sorgen, dass 'root' die ssh-Keys zu unserem Webserver bekommt. ACHTUNG! Damit werden evtl. vorhandene ssh-keys von root gelöscht! root erhält damit alle keys von Deinem User-Account! (Wer das vermeiden will, muss das händisch machen - Auf einer neuen Installation stellt das aber i.d.R. kein Problem dar ;)

```
sudo cp ~/.ssh/* /root/.ssh/
```

Mit einem Eintrag in die /etc/init.d/rc.local sorgen wir dafür, dass unser Script beim Systemstart als letztes ausgeführt wird.

```
sudo nano /etc/init.d/rc.local
```

Dort keine Änderungen vornehmen, außer den Eintrag in die letzte Zeile:

```
/home/USER/ip-update.sh
```

Wichtig ist, den vollständigen Pfad anzugeben, wobei USER mit Deinem Usernamen zu ersetzen ist (beim Raspbian also home/pi/ip-update.sh). Die Datei dann speichern und schließen (STRG+O → [Return] → STRG-X)

Das war's auch schon auf dem Rechner und Webserver, ab jetzt wird vom Zielrechner zu den in der crontab angegebenen Zeiten und beim Systemstart (oder Reboot) die öffentliche IP vom externen Webserver abgeholt, zwischengespeichert und wieder hochgeladen, damit man von außen weiß, wo der Zielrechner zu finden ist.

Von einem entferntem Terminal auf den Zielrechner zugreifen

Man könnte nun einfach im Browser mit http://SERVERNAME/raspip/index.html oder http://SERVERNAME/raspip/ die IP des Zielrechners anzeigen lassen und diese manuell eingeben. Das

ist jedoch sehr umständlich, weshalb wir das ein Script machen lassen. Wir legen auf unserem Rechner, mit dem wir auf unser unbekanntes Ziel zugreifen wollen eine Datei mit dem Namen 'raspssh' an.

```
nano raspssh
```

In diese fügen wir nun folgenden Inhalt ein:

```
#!/bin/sh
curl http://SERVERNAME/raspip/index.html > ~/raspip
xargs -a ~/raspip ssh -l USER
rm ~/raspip
```

Erklärung: „curl http://SERVERNAME/raspip/index.html > ~/raspip“ lädt die vom Zielrechner hinterlegte IP in die lokale Datei ~/raspip „xargs -a ~/raspip ssh -l USER“ verkettet den Inhalt von ~/raspip (die Ziel-IP) mit dem Befehl 'ssh -l USER', wobei USER durch den Benutzernamen auf dem Zielrechner ersetzt werden sollte (bei raspbian … ssh -l pi) „rm ~/raspip“ löscht nach Beenden von ssh die angelegte Kopie der IP wieder.

Das Script muss noch ausführbar gemacht werden:

```
chmod +x raspssh
```

Damit das Script auch auf anderen User-Accounts nutzbar ist, schieben wir es nun nach /usr/bin :

```
sudo cp raspssh /usr/bin/
```

Mit 'raspssh' verbinden wir uns nun von überall mit unserem Zielrechner. Vorraussetzung ist natürlich, dass dieser entweder direkt am Netz hängt oder eine Portweiterleitung eingerichtet ist.