DynDNS selbst-gemacht!

HERMANN GOTTSCHALK UND THOMAS BIRNTHALER

Dynamische IP-Adressen zu verwalten, bedarf einer zentralen Instanz, die über eine feste IP-Adresse erreichbar ist. Eine Möglichkeit, diese Informationen zu administrieren, ohne sie über einen DNS-Server öffentlich zu machen, wird in diesem Beitrag gezeigt.

Das Bedürfnis, in unserer vernetzten Welt Remotezugänge zu schaffen, ist offensichtlich. Sei es, um Mails auf dem heimischen PC mit einem Kommandozeilentool wie *mutt* oder *pine* abzurufen, Datenbestände über »dünne« Leitungen abzugleichen, Backups zu fahren (mit *rsync* oder *cvs*) oder den PC eines Heimmitarbeiters zu administrieren. Allen diesen Anforderungen ist eine Hürde gemeinsam: Solche Systeme besitzen meist keine statische IP-Adresse. Woher also die Information nehmen, unter welcher IP-Adresse die Gegenstelle zu erreichen ist?

Ein geläufiges Hilfsmittel ist *DynDNS* (*http://www.dyndns.org*), also ein Dienst, der diese Information zeitnah zur Verfügung stellt und aktuell hält. Jedoch ist dieser Dienst zumindest mit einer Registrierung und bei kommerziellem Einsatz zusätzlich mit Kosten verbunden.

Besitzt man einen eigenen öffentlichen DNS-Server, kann man das natürlich selbst realisieren. Aus sicherheitstechnischer Sicht ist es aber nicht immer erwünscht, Informationen über die eigene Netzwerkstruktur über DNS öffentlich zu machen. Im folgenden wird eine einfache Möglichkeit gezeigt, diese Funktionalität ohne Einsatz eines DNS-Servers zu realisieren, die Informationen sind nur einer geschlossenen Benutzergruppe zugänglich.

Die Praxis zeigt, daß in den Außenstellen häufig vom Provider mitgelieferte DSL-Router zum Einsatz kommen, die die Internetverbindung herstellen. In diese proprietären Systeme jeweils ein spezielles Skript einzubauen, das die aktuelle IP-Adresse einer zentralen Stelle übermittelt, erscheint doch sehr aufwendig. Auch könnte ein anderes System wie beispielsweise Windows als Zugangspunkt zum Einsatz kommen, an dem man vielleicht noch nicht einmal Einstellungen vornehmen darf.

Netzwerkstruktur

Hinter einer DSL-Router-Anbindung wie oben beschrieben ist sowohl der Zeitpunkt des Verbindungsneuaufbaus als auch die neue IP-Adresse schwierig zu ermitteln; vor allem ist der Weg, um an diese Information zu gelangen, bei jedem System anders.

Die von uns gewählte Lösung bedarf keiner Anpassung und ist somit *portabel* einsetzbar, egal, welches System den Internetzugang realisiert. Ein beliebiger Client im Netz der Außenstelle muß einfach periodisch eine Webseite abfragen. Das bekommen die meisten Linux- (und Windows-) Benutzer sicherlich hin.

Zwingend notwendig ist ein mit statischer IP-Adresse im Internet erreichbarer Server. Er wird eingesetzt, um die dynamischen IP-Adressen zu sammeln und zur Verfügung zu stellen. Alle anderen Einheiten sind mit dynamischen IP-Adressen an das Internet angebunden. Es wird angenommen, daß auf dem Server ein Webserver, beispielsweise Apache, installiert ist, Zugriff auf dessen Logdateien besteht und Skripte abgelegt und ausgeführt werden können.

Server I

Der auf dem Server installierte Webserver muß die Zugriffe im *Common Log Format (CLF)* mitprotokollieren. Es wird angenommen, daß die Logdatei *access.log* heißt sich unter */var/www/logs* befindet.

Auf dem Webserver wird eine Datei *dslcheck.html* angelegt, die eine korrekte, aber inhaltlich leere HTML-Seite enthält. Diese Seite wird von den auf diverse Standorte verteilten und mit dynamischen IP-Adressen ausgestatteten Clients minütlich abgerufen (siehe »Client I« weiter unten). Damit diese Aufrufe auswertbare Einträge in der Logdatei hinterlassen, die den Client identifizieren, muß diese Datei einer Zugangskontrolle unterworfen werden. Beim Apache-Webserver geschieht das mit einem Eintrag in der *httpd.conf*:

```
AuthType Basic
AuthName "DSLCheck"
AuthUserFile  /var/www/etc/passwd
Require valid-user
```

Mit dem Tool *htpasswd* ist die Datei *passwd* auf dem Webserver anzulegen und darin für jede zu verwal-

tende Außenstelle ein Benutzer einzutragen.

Clients I
In jeder Außenstelle muß auf einem Client das Programm *wget* installiert sein und ein Eintrag in einer *crontab* erfolgen:

```
* * * * * wget --http-user='office10' \
--http-passwd='secret' -O \
- http://foo.bar.com/dslcheck.html \
> /dev/null 2>&1
```

Diese Zeile generiert durch eine HTTP-Abfrage der Datei *dslcheck.html* jede Minute einen Eintrag in der Datei *access.log* des Servers in der Form *212.114.231.253 - office10 [09/Oct/2004 :00:55:22 +0200] "GET /dslcheck.html HTTP/1.0" 200 54*

Server II
Um diese Einträge in der Datei *access.log* auszuwerten, wird auf dem Server minütlich eine ASCII-Text-Datei erstellt, die die Außenstellen beziehungsweise User mit ihren IP-Adressen enthält. Ein Skript, das über die *crontab* ausgeführt werden kann, ist im Listing *fetch-dsl-ip.sh* auf der CD-ROM zu dieser freeX enthalten.
Die so erzeugte Datei *dsl-ip.txt* wird in einem zugangsgeschützten Bereich */staff* auf dem Webserver veröffentlicht, optimalerweise via https.

Clients II
Um nun auf die IP-Adresse einer Außenstelle zuzugreifen (hier auf die Außenstelle *home10*), muß clientseitig nur die Datei *dsl-ip.txt* vom Webserver angefordert und ausgewertet werden:

```
IP=$(wget --http-user="office10" \
     --http-passwd="secret" -O - \
  https://foo.bar.com/staff/dsl-ip.txt \
  2> /dev/null |
  grep "home10" | cut -d " " -f3,3);
```

Diese Information kann nun für SSH-Verbindungen, in Skripten und so weiter ausgenützt werden. Natürlich muß an den Außenstellen ein Dienst in irgend einer Form oder durch Portforwarding aktiv sein, der eine Verbindungsanfrage entgegennimmt.

Besitzt man bei einem Provider nur einen Webserver, darf aber darauf keine Shellskripten installieren, kann die beschriebene Vorgehensweise auch serverseitig durch den Einsatz eines PHP-Skripts realisiert werden (Listing *fetch-dsl-ip.php* auf der CD). Dieses Skript wird nur bei Bedarf vom Client aufgerufen, beispielsweise per *Wget* oder in einem Browser.

Webzugriffe auswerten

Die Dateien *dslcheck.html* und *dsl-ip.txt* sind natürlich zu ignorieren, wenn die Zugriffe auf den eigenen Webserver beispielsweise mittels *webalizer* ausgewertet werden. Die dann resultierenden Zugriffsstatistiken wären doch ein wenig zu optimistisch ;-)
Sicher wäre es geschickt, nur dann eine Meldung mit der IP-Information an den Server zu senden, wenn sich die IP-Adresse ändert. Hinter einer DSL-Router-Anbindung wie oben beschrieben ist aber sowohl der Zeitpunkt des Verbindungsneuaufbaus als auch die neue IP-Adresse schwierig zu ermitteln, und vor allem ist der Weg, um an diese Information zu gelangen, bei jedem System anders.
Auch DynDNS verwendet eine Server-Client-Lösung, die in periodischen Intervallen die IP-Information abgleicht. Es gibt keine andere praktikable und universell einsetzbare Lösung, die auf periodischen Verbindungsaufbau verzichtet. Die Intervalle kann man vergrößern (beispielsweise alle zehn Minuten statt jede Minute), um weniger Einträge in der Webserver-Logdatei zu erzeugen.
Angenommen, zehn Außenstellen verfahren gemäß dem obigen Schema. Dann kommen pro Tag 10 x 1440 x 80 Byte und damit circa 1 MByte an Logdaten zusammen. Soviel ist das auch nicht, zumal ja von beiden Skripten nur ein möglichst kleiner »hinterer Teil« dieser Daten gelesen wird. Werden die Logdaten einmal pro Woche beispielsweise mit *logrotate* komprimiert, ist das nicht der Rede wert. Dabei könnte man gleich noch die nun überflüssigen Abfragen wegwerfen und/oder die Uptime-Zeit der Verbindung zu den Außenstellen auswerten.
Die Vorgehensweise mit Shell- oder PHP-Skript haben spezifische Vor- und Nachteile. Die PHP-Variante belastet den Webserver nur bei Bedarf, erfordert jedoch eine PHP-Installation. Die Shell-Variante arbeitet mit Unix-Bordmitteln, belastet aber das System mehr. Bekommt eine Außenstelle eine neue IP-Adresse zugeteilt, dauert es maximal zwei Minuten – in der PHP-Variante sogar nur maximal eine Minute – bis diese Information wieder für alle Clients zur Verfügung steht.

Bewertung der Lösung

Gibt es aus irgendeinem Grund keine Möglichkeit, CGI- und/oder PHP-Skripten auszuführen, ist das Shellskript das Mittel der Wahl (häufig soll auch die Installation von PHP vermieden werden). Aus Performancegründen wertet es analog dem PHP-Skript nicht die gesamte Logdatei aus, sondern holt sich die benötigte Information möglichst weit hinten (*tail* liest definitiv nicht die ganze Datei, PHP springt zum Ende der Logdatei). Weiterhin kann man davon ausgehen, daß das Ende einer Logdatei meist noch im Speicher steht, wenn ständig in diese Datei geschrieben wird. Es finden also kaum echte Plattenzugriffe statt.

Fazit

Das beschriebenen Verfahren wird von uns für Remote-Datensicherung und Remote-Zugriff bei Kundensystemen eingesetzt. Sowohl die Kunden als auch unsere Firma sind per DSL-Anschluß mit dynamischen IP-Adressen an das Internet angebunden. Unser Webserver spielt den zentralen Part für die Ermittlung der IP-Adressen. Für einen Kunden sichern wir beispielsweise jede Nacht seine Daten im Umfang von etwa 16 GByte (hier kommt natürlich *rsync* zum Einsatz). ◆