SelfLinux-0.13.0

Der Web-Proxy Squid

Autor: Jörn Bruns (*joern_bruns@gmx.de*)
Formatierung: Florian Frank (*florian.frank@pingos.org*)

Inhaltsverzeichnis

9 Browser-Proxy-Einstellungen automatisieren und optimieren

1 Der Web-Proxy Squid

Sollen sich mehrere Surfer einen Web-Zugang teilen, der sicher, schnell und zudem flexibel ist, bietet sich ein Caching-Web-Proxy wie Squid an.

Squid steht unter der GNU GPL. Er ist sehr ausgereift, schnell und flexibel. Deshalb ist er von den Web-Proxies am weitesten verbreitet und wird gerade in sehr großen Umgebungen, wie Universitäten und großen, verzweigten Unternehmen, verwendet.

Warum sollten wir seine Vorteile nicht auch in kleineren Netzwerken nutzen?

2 Was macht ein Web-Proxy?

Zunächst die Vorteile eines Web-Proxies im allgemeinen:

1. Proxy heißt übersetzt **Stellvertreter**, und als solcher holt er für die Nutzer die Web-Seiten aus dem Netz. Nach außen ist netzwerktechnisch nur der Proxy zu sehen, der Zugreifende ist hinter ihm versteckt und dadurch geschützt.
2. Als weitere Fähigkeit kann ein Proxy meist statische Web-Inhalte zwischenspeichern, was **Caching** genannt wird. Ein erneuter Zugriff auf die gleichen Inhalte wird dadurch erheblich beschleunigt, und das bei gleichzeitig geringerer Netzlast!

3 Warum gerade den Squid als Web-Proxy verwenden?

Der Einsatz des Squid bringt mehrere Vorteile mit sich:

3.1 Freie Lizenz

Squid ist eine Open-Source-Entwicklung unter der GNU GPL. Somit fallen keine Lizenz-Kosten an, der Quelltext ist frei verfügbar und an eigene Bedürfnisse anpassbar.

3.2 Squid ist sehr stabil und ausgereift

Der Squid wird seit vielen Jahren entwickelt und hat sich auch gerade in größeren Umgebungen, wie Universitäten und großen Firmen, bewährt. Im Laufe der Entwicklung ist er schneller und vielseitiger geworden, so dass er sich hinter keinem kommerziellen Proxy verstecken muss.

Ein Blick in die gut dokumentierte /etc/squid.conf zeigt, wie viele Optionen mit dem Squid offen stehen. Und es gibt eine sehr aktive Gemeinde, die diesen Proxy immer weiter entwicket.

3.3 Beschleunigung und Entlastung des Internet-Zugangs

Statische Inhalte, welche einmal abgerufen wurden, können zwischengespeichert werden. Dazu gehören auch Grafiken von dynamisch generierten Seiten. Ein erneutes Abrufen solcher Inhalte, auch von einem anderen Benutzer, kann aus dem Zwischenspeicher bedient werden. Die Anfrage ist dadurch erheblich schneller beantwortet, der Internetzugang wird entlastet. Die Aktualität der Seiten wird durch sehr ausgeklügelte Methoden sichergestellt.

Die am häufigsten genutzten Seiten werden im Arbeitsspeicher gehalten (**hot object**). Die nicht so schnelle Festplatte wird für länger zurückliegende Zugriffe genutzt.

Eine zusätzliche Beschleunigung bewirkt das Zwischenspeichern der Zuordnung **Name zu IP-Adresse** (DNS-Caching).

3.4 Kontrolle der Zugriffe

Soll der Zugriff eingeschränkt werden, ist dies über die Rechtevergabe mit den **Zugriffs-Kontroll-Listen (Access Control Lists, ACL)** von Squid flexibel möglich. Sinnvoll kann dies sein, um die Ablenkung durch die Angebots-Flut des Internets einzuschränken, juristische Probleme zu vermeiden (z. B. durch pornografische Inhalte in Schulen) oder um die Online-Kosten im Griff zu halten.

Bereits getätigte Zugriffe können übersichtlich ausgewertet werden. So ist nachvollziehbar, wer welche Seiten aufgesucht und wer wie viel Daten übertragen hat. Ebenso kann ermittelt werden, welche URLs am häufigsten aufgesucht wurden oder auch wieviele Daten insgesamt übertragen wurden. Zusätzliche Tools, wie webalizer und `cachemanager`, helfen die Logfiles auszuwerten. So kann rechtzeitig ermittelt werden, wann die Proxy-Hardware nicht mehr ausreicht. Oder es können bestimmte Seiten gesperrt werden, die den Internet-Zugang überstrapazieren.

In diesen Kontrollmöglichkeiten liegt natürlich auch die Gefahr, in die Privatsphäre anderer einzugreifen. Deshalb ist die Informationsflut in die Logfiles abgestuft deaktivierbar (z. B. mit dem Parameter `client_netmask`).

3.5 Erhöhte Sicherheit

Eine Firewall kann ein lokales Netz effektiver absichern, wenn sie einen Proxy wie den Squid verwendet, anstatt nur auf Paketfilterung zu vertrauen. Der Grund dafür liegt darin, dass Paketfilter auf TCP/IP-Ebene, nicht aber den Inhalt von HTTP- und FTP-Verbindungen analysieren können. Proxies können aber genau diese Inhalte erkennen.

Zusätzlich können Proxies die Clients des lokalen Netzes erheblich besser verbergen, als es durch Network Adress Translation (NAT) möglich wäre. Mit Hilfe des Squid kann genau definiert werden, was an den Webserver übertragen werden soll und was nicht (z. B. mit dem Parameter `forwarded_for`).

Es können einige Viren geblockt werden (z. B. Nimda).

3.6 Namensauflösung vereinfacht

Praktisch ist auch, dass ein Proxy die Namensauflösung zu den IP-Adressen übernimmt. Es muss im internen Netz kein öffentlicher Name auflösbar sein, was die DNS-Konfiguration erleichtert.

3.7 Große Flexibilität

Insbesondere in komplexeren Netzwerken ist der Squid-Proxy sehr flexibel.

So kann z. B. genau definiert werden, welche Seiten aus dem Internet, welche von einem anderen Proxy und welche direkt aus dem lokalen Netz geholt werden sollen.

Es kann über mehrere Standorte ein so genannter **Cache-Verbund** aufgebaut werden, was die Netzlast deutlich vermindern kann und die Ausfallsicherheit erhöht.

Des weiteren sind zusätzliche Module verfügbar, welche die Funktionalität des Squid erweitern können. Hierzu zählen z. B. SquidGuard oder auch Module für die Nutzung einer Windows-Benutzerverwaltung im Squid.

4 Squid-Konfiguration auf die Schnelle

Nach der Installation ist der Squid nicht ohne Anpassungen lauffähig, er muss zunächst über die Datei /etc/squid.conf an die vorhandene Netzwerk-Situation angepasst werden. Diese Datei ist sehr gut kommentiert, aber aufgrund der vielen Einstelloptionen auch sehr umfangreich. Zum Glück können fast alle Optionen unverändert übernommen werden.

Um den Proxy erst einmal zum Laufen zu bringen, müssen die hier vorgestellten Parameter angepasst werden. Ergänzend kann auch in die QUICKSTART-Datei der Squid-Doku geschaut werden (meist unter /usr/share/doc/squid).

4.1 Zugriffs-Rechte

Die Voreinstellung des Squid ist aus Sicherheitsgründen so, dass keiner surfen darf. Die Rechtevergabe ist sehr flexibel und damit leider auch kompliziert. Es soll der Einfachheit halber zunächst allen Nutzern im eigenen lokalen Netz Vollzugriff auf das Internet gewährt werden.

Für die Rechtevergabe muss zunächst die Zugriffs-Kontroll-Liste (Access Control List, ACL) definiert werden. Anschließend wird über den Namen dieser ACL das Recht mit dem Schlüsselwort `http_access` zugewiesen. Weitere Details sind zu finden unter: Rechtevergabe unter Squid im Detail

Die Syntax der ACL:

```
acl <frei_definierbarer_Name> <acl_Typ> <Werte>
```

Angenommen, die PCs des eigenen Netzes verwenden alle IPs aus dem Bereich **192.168.10.***, könnte die Konfiguration für Vollzugriff des eigenen lokalen Netzes wie folgt aussehen:

```
acl allowed_hosts src 192.168.10.0/255.255.255.0

http_access allow allowed_hosts
http_access deny all
```

Die letzte Zeile ist schon vorhanden, die beiden darüber liegenden müssen neu angelegt werden. Die Reihenfolge der `http_access`-Zeilen ist wichtig! Für die eigene Nutzung muss natürlich die IP-Netz-Adresse **192.168.10.0/255.255.255.0** angepasst werden.

4.2 Größe des Festplatten-Zwischenspeichers

Dieser Parameter ist nicht zwingend zu verändern, damit Squid lauffähig wird, aber so wichtig, dass er hier aufgeführt wird.

Soll der Zwischenspeicher (**Cache**) viele Objekte enthalten können, muss die knappe Voreinstellung von 100 MB vergrößert werden.

Doch Vorsicht: Es kann leider nicht der gesamte verfügbare Platz einer Partition genutzt werden, da zusätzlich noch Verwaltungsdaten Platz benötigen. Steht dem Squid nicht genügend Platz zur Verfügung, kann er nicht

arbeiten. Die Angabe der Zwischenspeicher-Größe sollte deshalb immer mindestens **10%** unter dem Wert des freien Platzes liegen.

Die Syntax ist:

```
cache_dir <Pfad zum Zwischenspeicher> <Größe> <nicht zu verändernde Parameter>
```

Soll die Größe z.B. auf 10000MB eingestellt werden, kann folgendes eingetragen werden:

```
cache_dir /usr/local/squid/var/cache 10000 16 256
```

4.3 Squid hinter einer Firewall

Kann der Squid direkt auf das Internet zugreifen, sollten die beiden genannten Einstellungen genügen, um erst mal loslegen zu können. **Direkt zugreifen** bedeutet hier, dass der Proxy über das **Default Gateway** auf die Ports 80, 443 und 21 im Internet zugreifen kann.

Wenn das nicht möglich ist, wird die vorgelagerte Firewall einen Web-Proxy haben, von dem sich der Squid die Daten holen kann. In diesem Fall ist zunächst folgendes einzutragen:

```
cache_peer <firewall> parent <Proxy-Port> <Optionen>
```

Zum Beispiel:

```
cache_peer 192.168.10.1 parent 8080 no-query
```

Um sicherzustellen, dass Squid für alle unbekannten, nicht im eigenen Netz vorkommenden Web-Server auf das Internet zugreift, die eigenen Web-Server jedoch direkt anspricht, sollte folgendes ergänzt werden:

```
acl localnet srcdom .selflinux.org

always_direct  allow  localnet
always_direct  allow  allowed_hosts
never_direct allow  all
```

Die eigene Domain, hier **selflinux.org**, muss natürlich angepasst werden. Die ACL für die `allowed_hosts` ist schon weiter oben definiert.

5 Squid-Daemon starten

Die Installation von Squid über RPM oder DEB ermöglicht eine relativ einfache Inbetriebnahme des Dienstes. In diesen Fällen ist bereits einiges vorkonfiguriert, was ansonsten von Hand gemacht werden muss (etwa das Anpassen von Dateisystemrechten oder das Erstellen eines Init-Skripts).

Bevor Squid das erste Mal gestartet wird, muss die Cache-Verzeichnisstruktur angelegt werden mit:

```
root@linux / # squid -z
```

Wer will, kann nun den Squid zunächst im Debug-Modus auf der Konsole starten. Fehler werden so auf dem Terminal ausgeben, da Squid nicht in den Hintergrund geschickt wird:

```
root@linux / # squid -NCd1
```

Ist alles in Ordnung, sollte folgende Zeile erscheinen:

```
Ready to serve requests.
```

Läuft alles zur Zufriedenheit, kann Squid in Zukunft über das von der jeweiligen Linux-Distribution vorgesehene Startskript gestartet werden, meist durch:

```
root@linux / # /etc/init.d/squid start
```

Bei Problemen sollten die Logfiles /var/log/messages und cache.log untersucht werden.

Wo die cache.log abgelegt ist, kann ermittelt werden mit:

```
root@linux / # grep cache_log /etc/squid.conf
```

6 Rechtevergabe unter Squid im Detail

6.1 Warum Zugriffsrechte für den Internet-Zugriff?

Werden die Rechte lediglich so gesetzt wie im Abschnitt Squid-Konfiguration auf die Schnelle beschrieben, können alle Nutzer auf sämtliche Inhalte des Internet zugreifen.

Das mag als Konfiguration häufig reichen, es können sich daraus jedoch folgende Nachteile ergeben:

* Der Internetzugang ist schnell überlastet und damit langsam.
* Erhöhte Kosten können entstehen.
* Die Nutzer können sich in der Informationsflut des Internets verlieren (wichtig bei Schulungen/Schülern).
* Es können sich juristische Probleme ergeben, wenn sich z. B. Schüler pornografische Inhalte anschauen

6.2 Umsetzung

Das Setzen der Rechte unter Squid ist sehr flexibel, aber leider nicht selbsterklärend.

Mit einer ACL wird zunächst festlegt, auf was (Ziel) oder von was (Quelle) zugegriffen wird. Über `http_access` wird dieser ACL anschließend ein Verbot oder eine Erlaubnis zugewiesen.

6.2.1 ACL-Elemente - Ziel oder Quelle definieren

Die Syntax einer ACL sieht folgendermaßen aus:

```
acl <frei_definierbarer_acl_Name> <acl-Typ> <Wert ...>
```

Über den frei definierten ACL-Namen kann mit Hilfe des unten erläuterten `http_access` das gewünschte Recht vergeben werden. Diese ACL-Namen dürfen natürlich nicht doppelt vergeben werden.

Es können verschiedene Arten von Quellen und Zielen zur ACL-Definition verwendet werden. Hier die für die Zugriffsverwaltung genutzten ACL-Typen:

* `src`: Absender-IP-Adressen der Client-PCs, die über Squid auf Web-Inhalte zugreifen wollen.
* `dst`: Ziel-IP-Adressen, auf die zugegriffen werden soll.
* `dstdomain`: Name der Ziel-Domain, auf deren Server im Internet zugegriffen werden soll. Hier kann auch direkt ein Server angegeben werden!
* `dstdom_regex`: Wie `dstdomain`, zusätzlich können Reguläre Ausdrücke verwendet werden, um die Liste der Server zu erweitern.
* `time`: Zur Festlegung bestimmter Zeitbereiche, in denen gesurft werden darf.
* `url_regex`: URLs können über Reguläre Ausdrücke definiert werden.
* `urlpath_regex`: Der Pfad, also alles, außer dem Protokoll (wie **http://**) und dem Rechnernamen (wie **www.selflinux.org**), wird mit dem angegebenen Regulären Ausdruck verglichen.
* `ident`: Vergleich der Namen, die von den Unix-Clients mit dem `identd` übertragen werden, mit denen in der Liste. So kann eine einfache Benutzerverwaltung genutzt werden. Den `identd` gibt es auch für Windows als Programm oder als Dienst.
* `external`: Einbinden eines externen Hilfsprogramms, das z. B. ermöglicht, zur Benutzerverwaltung einen NT-Domänen-Kontroller zu nutzen.

Weitere ACL-Typen stehen für spezielle Anwendungen des Squid zur Verfügung, die nicht für die Benutzerverwaltung wichtig sind.

Werden mehrere Werte hinter dem ACL-Typ aufgelistet, braucht nur einer der Werte zu passen, um das zugehörige `http_access` zu aktivieren (OR-Logik, siehe Logik der Rechtevergabe).

Zur besseren Übersicht können die aufzulistenden Werte auch in eine eigene Datei ausgelagert werden. Dort wird für jeden Eintrag eine eigene Zeile angelegt. Die Datei muss Squid wie folgt bekanntgegeben werden:

```
acl <frei_definierbarer_Name> <acl-Typ> "<Pfad_zur_Datei>"
```

6.2.2 Zugriffsrechte für die ACLs definieren

Mit `http_access` wird in Kombination mit `allow` bzw. `deny` ein Recht für die definierten ACL-Elemente festgelegt.

Die Syntax:

```
http_access allow|deny <acl-Name ...>
```

Das sieht zunächst sehr simpel aus. Einer definierten ACL wird über `deny` oder `allow`" ein gewünschtes Recht zugewiesen, so wie im Beispiel des Abschnittes Squid-Konfiguration auf die Schnelle gezeigt:

```
acl allowed_hosts src 192.168.10.0/255.255.255.0
http_access allow allowed_hosts
```

Komplexer wird es, wenn mehrere ACL-Namen in einer Zeile aufgelistet sind. Dies bewirkt, dass alle aufgelisteten ACLs zutreffen müssen, damit das `allow` oder `deny`" in Kraft treten kann.

Beispiel:

```
acl all            src    0/0
acl selflin        dstdom .selflinux.org
acl pcRestr        src    192.168.20.0/255.255.255.0
acl allowed_hosts  src    192.168.10.0/255.255.255.0

http_access allow selflin       pcRestr
http_access allow allowed_hosts
http_access deny  all
```

Die Angabe der zwei ACL-Namen **selflin** und **pcRestr** nach dem ersten `http_access` bewirkt, dass Rechner mit der IP **192.168.20.*** nur auf SelfLinux-Seiten surfen dürfen. Die `allowed_hosts` dürfen alles sehen, da diese nicht mit einer zusätzlichen ACL in der `http_access`-Zeile eingeschränkt werden.

Hier ist gleich eine wichtige Eigenart des ACL-Typs `dstdom` zu erkennen: Der '**.**' vor dem Domainnamen sagt Squid, dass auch Subdomains in diese ACL fallen, wie z. B. www.selflinux.org oder srv.sub.selflinux.org.

6.2.3 Logik der Rechtevergabe

Es ist sehr wichtig für das Berechtigungsmodell von Squid, die OR/AND-Logik zu verstehen:

* Alle Elemente eines ACL-Eintrags werden mit OR verknüpft.
* Alle Elemente eines Access-Eintrags dagegen werden mit einem AND verknüpft.

Noch ein Beispiel, das fatalerweise überhaupt keinen Zugriff ermöglicht:

```
acl wir src 192.168.10.0
acl ihr src 192.168.20.0
http_access allow wir ihr
```

Hier würden Zugriffe dann erlaubt werden, wenn sich der Surfende zugleich mit den beiden Quell-IPs an den Proxy wendet, was nicht möglich ist. Somit werden keine Zugriffe mehr erlaubt.

Sollen dagegen beide IP-Adressen Zugriff erhalten, muss folgendes eingetragen werden:

```
acl wir src 192.168.10.0 192.168.20.0
http_access allow wir
```

Ein weiterer wichtiger Punkt ist die Reihenfolge der `access`-Listen.

Ist erst einmal ein Zugriff erlaubt worden, kann er durch darunter liegende Zeilen nicht wieder zurückgenommen werden.

Beispiel:

```
acl selflin  dstdom .selflinux.org
acl verboten src    10.0.0.23
http_access allow selflin
http_access deny  verboten
```

Obwohl **10.0.0.23** in der letzen Zeile alles verboten wird, kann dieser PC die Seiten von SelfLinux sehen, da die Erlaubnis über dem kompletten Verbot vergeben wurde.

7 Squid aus Quellen kompilieren

Selbst kompilieren hat mehrere Vorteile:

1. Es können zusätzliche Funktionen aktiviert oder zur Stabilitätserhöhung ungenutzte Funktionen deaktiviert werden
2. Die aktuellsten Versionen sind meist nur als Quellen verfügbar
3. Das kompilierte Programm ist optimal an die Umgebung angepasst, wie z.B. Prozessortyp und Bibliotheken.

Der Nachteil ist, dass die Installation erheblich aufwendiger ist.

7.1 Kompilieren und Installation

Zunächst sollten die neuesten Squid-Quellen geholt werden unter www.squid-cache.org (am besten die jeweils aktuellste STABLE-Version wählen).

Anschließend sind diese auszupacken mit:

```
root@linux / # cd /usr/local/src
root@linux /usr/local/src/ # tar xvjf squid-<Version>.tar.bz2
```

Mit dem `configure`-Kommando können Parameter übergeben werden, um Squid den eigenen Anforderungen anzupassen. Die möglichen Parameter sind zu erfahren mit:

```
root@linux / # cd squid-<Version>
root@linux / # ./configure --help
```

Anschließend kann kompiliert und installiert werden.

Soll Squid beispielsweise mit Zugriffsmöglichkeiten auf eine Windows-NT-Domäne installiert werden, kann der Aufruf von `configure` z. B. folgendermaßen aussehen:

```
root@linux / # ./configure --enable-auth="ntlm,basic"
--enable-external-acl-helpers="wbinfo_group"
root@linux / # make all
root@linux / # make install
```

7.2 Konfiguration

Die Konfigurationsdatei squid.conf liegt standardmäßig nach dem Kompilieren unter /usr/local/squid/etc/squid.conf. Der Übersicht und Einheitlichkeit halber sollte folgender Link erzeugt werden:

```
root@linux / # ln -s /usr/local/squid/etc/squid.conf /etc/squid.conf
```

Vor dem ersten Start ist die Cache-Verzeichnisstruktur und deren Rechte anzulegen mit:

```
root@linux / # mkdir -p /usr/local/squid/var/cache
```

```
root@linux / # mkdir -p /usr/local/squid/var/logs
root@linux / # chown -R nobody /usr/local/squid/var/cache
root@linux / # chown -R nobody /usr/local/squid/var/logs
root@linux / # /usr/local/squid/sbin/squid -z
```

Die Konfiguration über die Datei squid.conf ist identisch mit der von RPM- oder DEB-Paket-Installationen.

7.3 Installation kontrollieren

Der Squid kann im Debug-Modus gestartet werden, um mögliche Fehler gleich auf dem Terminal zu sehen (siehe Squid-Daemon starten).

7.4 Squid starten mit Startskript

Läuft der Squid im Testlauf ohne Probleme, sollte ein Startskript erstellt werden. Über dieses kann der Squid bei einem Neustart des Systems automatisch hochgefahren werden.

Zu beachten ist, dass der Squid bei noch aktiven Verbindungen längere Zeit braucht, bis er gestoppt ist. Deshalb ist hier für den Stop-Fall eine Schleife einzubauen, die das Skript erst dann beendet, wenn der Daemon wirklich gestorben ist, da ansonsten ein erneuter Start mit Fehlern abgebrochen wird.

```
#!/bin/bash

squid=/usr/local/squid/sbin/squid
test -x $squid || exit 0

case "$1" in
start)
        echo "Starting squid"
        $squid -D -sYC
        sleep 1
        $0 status
;;
stop)
        echo "Stopping squid"
        $squid -k shutdown
        n=0
        while $squid -k check && [ $n -lt 120 ]; do
                sleep 1
                echo -n .
                n=`/usr/bin/expr $n + 1`
        done
        $0 status
;;
status)
        $squid -k check
        /bin/ps aux | /bin/grep squid | /bin/grep -v -e "status" -e "grep"
;;
reload)
        $squid -k reconfigure
;;
restart)
        $0 stop && $0 start
;;
*)
        echo "Usage: $0 {start|stop|reload|restart|status|}" >&2
;;
esac
```

Dieses Skript ist als /etc/init.d/squid anzulegen und ausführbar zu machen.

Um einen automatischen Start des Squid nach einem Neustart zu erreichen, müssen die entsprechenden Links in den rc-Verzeichnissen auf /etc/init.d/squid gesetzt werden. Da sich die einzelnen Linux-Distributionen hier leider sehr unterscheiden, ist hierfür kein einfaches Beispiel möglich.

8 Squid in Windows-NT-Netz integrieren

8.1 Funktionen für die Windows-NT-Benutzerverwaltung

Der Squid ist flexibel, ausgereift und kostenlos, doch innerhalb eines Windows-Netzes hatte er bis zur Version 2.4 zwei gravierende Nachteile:

1. **Doppelte Benutzerverwaltung**: Sollen User unterschiedliche Web-Zugriffsrechte bekommen, müssen diese Nutzer dem Squid natürlich bekannt sein, d. h. es ist eine Benutzerverwaltung notwendig. Diese musste bislang zusätzlich, neben der Windows-Domänen-Benutzerverwaltung, betreut werden, was einen erheblichen administrativen Mehraufwand bedeutete.
2. **Verminderte Benutzerfreundlichkeit beim Surfen**: Eine transparente Authentifizierung durch den Internet Explorer (IE) war nicht möglich, d.h. es musste für den Zugang Name und Passwort in ein Browser-Fenster eingeben werden, das auch noch unabhängig von dem der Windows-NT-Domäne ist.

Diese Nachteile sind mit der Squid-Version 2.5 bei entsprechender Konfiguration behoben.

Das Squid- und das Samba-Team schufen gemeinsam die Möglichkeit, die Benutzerverwaltung einer Windows-NT-Domäne für Squid nutzen zu können. Der Zugriff auf die Windows-Domäne wird dem Squid-Proxy mit Hilfe neuer Helper-Module in Ergänzung zu Samba-Winbind ermöglicht.

Auch die Benutzerfreundlichkeit für den Surfenden ist mit der MS-Konkurrenz gleichgezogen. Dabei nutzt der Squid die gleichen Mechanismen (NTLM) wie ein MS-Proxy. Es ist mit dem IE keine zusätzliche Authentifizierung mehr notwendig. Mit allen anderen Browsern muss man sich authentifizieren, und zwar mit dem Nutzernamen und Passwort des NT-Benutzers.

Die Arbeitserleichterung für Administratoren eines Windows-Netzes ist enorm. Der neu angelegte User hat sofort die für ihn bestimmten Rechte, wenn er nur der entsprechenden Windows-Gruppe hinzugefügt wird. Da die User im Usermanager für NT normalerweise mit F8 von einer Vorlage kopiert werden, ist dies kein Mehraufwand.

Gibt es Probleme mit fehlenden Rechten, reicht nun meist ein Blick in den Usermanager, um diese zu kontrollieren.

Die Installation und Einrichtung mit der genannten Zielsetzung ist nicht trivial, sollte aber mit geringen Unix-Grundkenntnissen und dieser Anleitung möglich sein. Leider ist es meist unumgänglich, die Programme `samba` und `squid` selbst zu kompilieren. Vorhandene Binaries der Linux-Distributionen sind nicht mit den nötigen Parametern übersetzt und oft nicht aktuell genug.

8.2 Samba mit Winbind

8.2.1 Installation

Es müssen Entwicklertools wie make und `gcc` installiert sein. Die folgenden Ausführungen beziehen sich auf die Samba-Version 3.0.*.

Nach dem Download der Quellen sollten diese unter /usr/local/src/ ausgepackt werden. Samba ist zu übersetzen und zu installieren mit:

```
root@linux / # cd /usr/local/src/samba-<Version>/source
```

```
root@linux / # ./configure --with-winbind
root@linux / # make
root@linux / # make install
```

8.2.2 Konfiguration von Samba/winbind

Als erstes ist eine `smb.conf` aus dem Quelltest in das richtige Verzeichnis zu kopieren:

```
root@linux / # cp
/usr/local/src/samba-<Version>/examples/smb.conf.default
```

Um die Konfigurationsdatei leichter zu finden, sollte ein Link in das Verzeichnis /etc/ erstellt werden mit:

```
root@linux / # ln -s /usr/local/samba/lib/smb.conf /etc/smb.conf
```

Nun sollte diese Datei folgendermaßen angepasst werden:

```
workgroup = <NT-Domänen-Name>
security = domain
# hier die Domain Controller des Standortes eintragen
password server = <dc1> <dc2>
wins support = no
# Hier die WINS-Server der Standorte eintragen
wins server = <IP_des_WINS-Servers>
max log size = 10000
local master = no
winbind enum users = yes
winbind enum groups = yes
winbind use default domain = yes
idmap uid = 10000-20000
idmap gid = 10000-20000
template shell = /bin/false
```

Unter **workgroup** ist der eigene NT-Domänen-Name einzusetzen. Als **password server** muss mindestens ein Domänen-Controller angegeben werden.

Nun kann der Samba-Rechner in die Windows-NT-Domäne aufgenommen werden mit:

```
root@linux / # /usr/local/samba/bin/net rpc join -S <PDC> -U
<Administrator>
```

8.2.3 Installation testen

Nun können die Dienste von Samba gestartet werden:

```
root@linux / # /usr/local/samba/bin/nmbd -D
root@linux / # /usr/local/samba/bin/smbd -D
root@linux / # /usr/local/samba/bin/winbindd -B
```

Um zu sehen, ob diese auch laufen, listet folgendes Kommando die gestarteten Services auf:

```
root@linux / # ps aux | egrep "(mbd|winbind)"
```

Die Anbindung des Samba-Servers an die Windows-Domänen-Benutzerverwaltung kann getestet werden mit:

```
root@linux / # wbinfo -u
```

Wenn Winbind korrekt arbeitet, sollten alle User der NT-Domänen ausgegeben werden.

8.3 Squid

8.3.1 Kompilierung, Installation und Konfiguration

Die Kompilierung und Installation ist im Abschnitt Squid aus Quellen kompilieren abgehandelt und mit einem passenden Beispiel erklärt worden.

Zur Nutzung von NT-Gruppen muss des Weiteren sichergestellt werden, dass ein `perl` ab der Version 5.8 installiert ist.

Die spezifischen Einträge in der /etc/squid.conf sind:

```
# Einbindung der neuen ACL-Helper Schnittstelle, hier NT-Domänen
external_acl_type NT_global_group children=10 ttl=900 %LOGIN
/usr/local/squid/libexec/wbinfo_group.pl

auth_param ntlm program /usr/local/samba/bin/ntlm_auth --helper-protocol=squid-2.5-ntlmssp
auth_param ntlm children 80
auth_param ntlm max_challenge_reuses 1
auth_param ntlm max_challenge_lifetime 5 minutes
auth_param basic program /usr/local/samba/bin/ntlm_auth --helper-protocol=squid-2.5-basic
auth_param basic children 50
auth_param basic realm squid proxy-caching web server
auth_param basic credentialsttl 2 hours
# acl <FreiDefinierbarerName> <aclTyp> <Definition>
acl ProxyUsers external NT_global_group <NT-Gruppen-Name>
acl AuthorizedUsers proxy_auth REQUIRED
# Beispiel für die Rechtevergabe:
acl selflinux dstdomain .selflinux.org
# Vollzugriff für diese Gruppe
http_access allow AuthorizedUsers ProxyUsers
# Dies hätte nur Zugriff auf die Selflinux-Seiten erlaubt:
#http_access allow AuthorizedUsers ProxyUsers selflinux
```

In der Squid-Dokumentation zu NTLM wird eine Anzahl von 5 Kind-Prozessen vorgeschlagen. Dies bewirkte in der vom Autor betreuten Umgebung **aufpoppende** Anmelde-Fenster. Deshalb scheint es empfehlenswert zu sein, reichlich Kind-Prozesse starten zu lassen. Zur Beschleunigung können zusätzlich die beiden challenge-Werte höher gesetzt werden als vorgegeben. Diese legen zusammen die Gültigkeitsdauer einer Anfrage an die Windows-Domänen-Controller fest.

Wer Hilfe zu den Funktionen der weiteren Parameter sucht, findet diese ausführlich in der /etc/squid.conf beschrieben.

Nun muss noch der Squid-Daemon folgende Berechtigung bekommen:

```
root@linux / # chgrp -R nogroup
```

```
/usr/local/samba/var/locks/winbindd_privileged/
```

Um einen gültigen Computeraccount in der Domain zu halten, sollte folgender Cronjob hinzugefügt werden:

```
0 2 * * * /usr/local/samba/bin/net rpc changetrustpw
```

8.3.2 Installation prüfen

Zunächst sollte die Anbindung an die NT-Domäne getestet werden. Nach der Eingabe von

```
root@linux / # /usr/local/samba/bin/ntlm_auth
--helper-protocol=squid-2.5-basic
```

wartet das Programm `wb_auth` auf die Eingabe eines gültigen NT-Accounts in Form von

```
<NT-Domänen-Name>\<Accout> <Password>
```

wie z.B.:

```
Domain\joern geheim
```

Die Ausgabe muss mit einem **OK** abschließen.

Nach dem Anlegen der Cache-Verzeichnisstruktur kann der Squid Debug-Modus gestartet und mögliche Fehler behoben werden, siehe dazu ▶ Squid-Daemon starten.

8.3.3 Squid starten

Nun muss ein Startskript erstellt und mit dem gewünschten Runlevel verlinkt werden. Siehe hierzu ▶ Squid starten mit Startskript

8.3.4 Zugriffsrechte auf Internet-Seiten setzen

Für die User-Verwaltung bleiben folgende Aufgaben:

8.3.4.1 Squid-Rechte

Für jede globale Windows-NT-Gruppe muss eine ACL in der /etc/squid.conf erstellt werden. Dort wird das externe Programm, definiert über den oben angegebenen `external_acl_type`", eingebunden und damit eine Verknüpfung von NT-Gruppen mit Squid-ACL-Namen erreicht:

```
acl <Frei_definierbarer_Name> external NT_global_group <WinNT-GruppenName>
```

Beispiel:

```
acl verwaltung  external NT_global_group verwaltung
```

Eine weitere ACL ist für die Seiten zu erstellen, die von der Gruppe gesehen werden sollen:

```
acl <GruppenName> dstdomain <.Dns-Domäne1> <.Dns-Domäne2>  ...
```

Beispiel:

```
acl verwaltung_dstdom dstdomain .selflinux.org .intranet.de
```

Als letztes sind für die ACLs die Rechte festzusetzen:

```
http_access allow AuthorizedUsers <ACL-GruppenName> <ACL-Zieldomänen-Name>
```

Beispiel:

```
http_access allow AuthorizedUsers verwaltung verwaltung_dstdom
```

Weitere Informationen zur Rechtevergabe siehe unter Rechtevergabe unter Squid im Detail.

8.3.4.2 NT-Domänen-Rechte

Die Surfer müssen hier nur in die globalen Gruppen aufgenommen werden, die auf dem Squid mit den entsprechenden Gruppen verknüpft sind.

9 Browser-Proxy-Einstellungen automatisieren und optimieren

Netscape hat für seinen Browser eine Möglichkeit entwickelt, über Javascript die Proxy-Einstellungen des Browsers zu konfigurieren. Inzwischen haben alle bekannten Browser diese Fähigkeit integriert.

Über ein solches Skript kann festgelegt werden, auf welcher Adresse und über welchen Port der Proxy seinen Dienst zur Verfügung stellt. Zusätzlich kann definiert werden, welche IPs und URLs über den Proxy zu gehen haben und welche dagegen direkt an einen Web- oder FTP-Server weitergeleitet werden.

Worin besteht nun der Vorteil, ein derartiges Skript zu nutzen, gegenüber dem direkten Eintragen des Proxies in der Browser-Konfiguration?

* **Erhöhte Flexibilität:** Wenn der Proxy die IP oder den Port wechselt, sich die Einträge für direkten Zugriff ändern, reicht ein Eintrag in dem zentralen Skript. Ansonsten müssten zur Änderung alle Client-PCs angefasst werden.
* **Erhöhte Ausfall-Sicherheit:** Sind mehrere Proxies vorhanden, können alle eingetragen werden. Ist der erste in der Liste nicht erreichbar, wird der zweite genommen u. s. w. Es kann sogar eingetragen werden, dass, wenn kein Proxy zur Verfügung steht, der Browser direkt ins Internet geht (wenn dies netzwerktechnisch erlaubt sein sollte). Soll der Proxy dagegen direkt in die Browser-Konfiguration eingetragen werden, ist nur **ein** einziger Proxy-Eintrag möglich.
* **Steuerung des Traffic:** Sind die Internet-Nutzer über mehrere Standorte verteilt, kann über die Feststellung, aus welchem Netz er zugreift, der optimale Proxy für ihn zugewiesen werden.
* **Erleichterung bei Wartung:** Schnelle Reaktion bei Ausfall eines Proxys ist durch einfache Konfiguration des Proxy-Skriptes möglich. Bei Wartungsarbeiten kann vorübergehend einfach ein anderer Proxy zugewiesen werden.

Der Pfad zu dem Skript ist in der Browser-Konfiguration einzutragen. Dies kann manuell oder zentral und automatisiert über zu kopierende Konfigurations-Dateien oder unter Windows NT/2000/XP über die Systemrichtlinien/Policy erfolgen.

9.1 Automatische Proxy-Konfiguration

Es ist nicht nötig, Javascript programmieren zu können, um die Möglichkeiten der Auto-Konfiguration zu nutzen. Es sollte reichen, die hier vorgestellten Beispiele den eigenen Netzen anzupassen, d.h. die Domains und IPs mit den eigenen zu ersetzen.

9.1.1 Eigene Server direkt ansprechen

Sollen sich die Clients direkt, ohne Proxy als Bindeglied dazwischen, an die eigenen Web-Server wenden, kann dies über den eigenen Domain-Namen wie im folgenden Skript umgesetzt werden:

```
function FindProxyForURL(url, host)
{
  if (dnsDomainIs(host, ".selflinux.org"))
    return "DIRECT";
  else
    return "PROXY wwwproxy.selflinux.org:3128";
}
```

Die Zahl hinter dem Proxy-Namen legt den Port des Proxies fest.

Eine andere Möglichkeit besteht in der Angabe der Netzadresse, welche direkt, ohne Proxy, angesprochen werden soll:

```
function FindProxyForURL(url, host)
{
  if (isInNet(host, "192.168.10.0", "255.255.255.0"))
    return "DIRECT";
  else
    return "PROXY wwwproxy.selflinux.org:3128";
}
```

9.1.2 Proxy-Auswahl für verschiedene Standorte automatisieren

Sind mehrere Standorte mit relativ langsamen Standleitungen verbunden, ist der Aufbau eines Cache-Verbundes sinnvoll. Dafür werden mehrere Squid-Proxies aufgesetzt und sinnvoll miteinander gekoppelt.

Für einen derartigen Proxy-Verbund ist es wichtig, den Internet-Nutzern immer den Proxy vor Ort zuzuweisen, um die in dem Standort zwischengespeicherten Seiten nutzen zu können. Dadurch wird eine Standleitung oder auch ein VPN entlastet.

Die URL, unter welcher das Skript abgelegt ist, gilt für alle Standorte. Besonders Mitarbeiter, welche oft die Standorte wechseln, werden dies sehr begrüßen.

Das Konfigurations-Skript muss nun feststellen können, an welchem Standort sich der Internet-Nutzer befindet. Da Netze verschiedener Standorte jeweils eigene IP-Bereiche haben sollten, kann dies über die Abfrage der Quell-IP erfolgen. Netscape hatte leider nicht vorgesehen, in der Konfigurationsdatei nach der Quell-IP fragen zu können. Deshalb hier eine Lösung über PHP, die diese Funktion hinzufügt. Der Webserver, der das Auto-Skript beherbergt, muss PHP installiert haben.

```
<?php
header("Content-type: application/x-ns-proxy-autoconfig");
if (preg_match("/192\.168\.85/", $REMOTE_ADDR)) {
  $adrr = "PROXY wwwproxy.selflinux.org:3128; PROXY wwwproxy2.selflinux.org:3128";
}
if (preg_match("/192\.168\.75\./", $REMOTE_ADDR)) {
  $adrr = "PROXY 10.20.20.1:3128; PROXY wwwproxy.selflinux.org:3128";
}
?>
```

```
function FindProxyForURL(url, host)
{
if (isPlainHostName(host) ||
  dnsDomainIs(host, ".selflinux.org") ||
  dnsDomainIs(host, ".partnernetz.de") ||
  isInNet(host, "192.168.85.0", "255.255.255.0") ||
  isInNet(host, "172.20.20.80", "255.255.255.255") ||
  isInNet(host, "192.168.75.0", "255.255.255.0")
    return "DIRECT";
  else
  return "";
}
```

Wird also z.B. von einem **192.168.75.*** Netz zugegriffen, wird der Proxy **10.20.20.1** zugewiesen. Wenn der nicht verfügbar ist, wird **wwwproxy.selflinux.org** genutzt. Und das, ohne dass es der Anwender bemerkt.