

# Squid IP Auth

---

Sichere Benutzerauthentifikation an einem Firewall Proxy

Steffen Dettmer © 1999  
Darf unter der GPL verwendet werden

# Inhalt

Inhalt	2
Projektziel	3
Aufgabenstellung	3
Hintergrund	3
Planungsgedanken	3
Abgrenzung	4
Einsatz	4
Lösungsbeschreibung	4
Einführung	4
Idee	5
Umsetzung	5
Arbeitstiere	6
Implementierung	6
SSL-Server	7
Distribution	7
Installation	7
Durchgeführte Tests	8
Authentifikation am Squid Proxy aus Anwendersicht	9
Authentifikation aus Sicht des Benutzers	9
Authentifikation aus Sicht des Administrators	9
Bewertung unter verschiedenen Gesichtspunkten	9
Installations-, Konfigurationsaufwand	9
Sicherheit	9
Portierbarkeit	10
Wartbarkeit	10
Erweiterbarkeit	10
Skalierbarkeit	11
Weiterführende Literatur:	11

# Projektziel

## **Aufgabenstellung**

Es ist eine Möglichkeit zu suchen, um Benutzer an einem Firewall Proxy, genauer gesagt an einem HTTP Internetcache anzumelden. Dabei sollen die Benutzer ihr gewohntes Windows Domain Kennwort verwenden können, um sich zu authentifizieren. Da dieses Passwort auch für andere Dienste gültig ist, darf dieses nicht im Klartext über das Netzwerk übertragen werden. Nach Möglichkeit soll dabei der bewährte Hochleistungscache „Squid“ verwendet werden und das System unter Linux laufen, so daß ein gute Stabilität erreicht wird und keine Softwarelizenzen erworben werden müssen.

## **Hintergrund**

Einem lokalen Netz wird über eine Firewall das Abrufen von Internetseiten über das HTTP Protokoll ermöglicht. Diese Firewall arbeitet dabei als Application Gateway. Vor dieser Firewall steht ein zweiter Proxy, der den Datenstrom nach Viren untersucht und gegebenenfalls entsprechend reagiert. Davon wiederum steht sinnvollerweise ein Cacheing Proxy („Clean Cache“), um die durch die Virensuche erzeugten Performanceverluste zu reduzieren und den Datenverkehr ins Internet und die damit verbundenen Kosten zu senken.

Eine Firewall unterstützt in der Regel Benutzerauthentifizierung, die allerdings aus dreierlei Gründen nicht verwendet werden kann:

1. Die Passwörter werden im Klartext übertragen
2. Die Kennwörter werden nicht gegenüber der NT Domain überprüft
3. Ein sich authentifizierender Benutzer würde auch allen anderen den Zugriff erlauben, da auf Grund der Natur des HTTP Protokolls prinzipiell nur auf Grund von IP-Adressen autorisiert werden kann.

Im skizzierten Fall würden aber alle Verbindungen vom Virenschanner aus kommen, und dessen IP-Adresse würde freigeschaltet werden. Die Authentication muß also am Cacheing Proxy erfolgen, da nur dieser direkte Verbindungen mit den Client Workstations unterhält.

Leider erfüllt der Internetproxy „Squid“ diese Kriterien nicht, so daß er entsprechend angepaßt bzw. erweitert werden muß, ohne die Stabilität oder Sicherheit zu gefährden. Hinzukommend wird verschlüsselte Proxyauthentifizierung von den gängigen Browsern leider nicht unterstützt.

## **Planungsgedanken**

Es folgen einige Punkte, die im Vorfeld bedacht wurden und das Aussehen der Lösung natürlich maßgebend beeinflussten. Hierbei wurde das Problem genauer definiert und eine Lösung entworfen

## **Abgrenzung**

Die Lösung soll möglichst klein und einfach sein. Es ist ausreichend, ein Netz mit weniger als 1000 Clientenmaschinen bedienen zu können. Es muß nicht auf mehreren parallel arbeitenden Proxies lauffähig sein. Andere Authentifizierungsverfahren als Passwort-Authentifizierung werden ebensowenig unterstützt, wie „Failsafe“ oder „automatic Failover“ Features. Es sind nur einfache Logfiles erforderlich, Angriffsversuche sollen nicht besonders erkannt und behandelt werden.

## **Einsatz**

Dieses System wird in genau der folgenden Konfiguration verwendet: Authentifikation auf einem Proxy vor einem anderen Proxy („Proxy-Chain“). Es muß ein Windows NT Server oder eine Windows NT Workstation für die Passwortvalidierung bereitstehen (aber nicht dediziert). Das System läuft auf PC Hardware (IBM-PC kompatible Architektur) gängiger Leistungsklasse und Ausstattung (Standard-PC-Server). Die typische Zielgruppe ist ein mittleres Firmen LAN mit geringem bis mittlerem Sicherheitsbedarf, in dem nicht alle Benutzer berechtigt sind, aus dem Internet Daten zu laden (z.B. zu „surfen“). Als unterliegendes Betriebssystem wurde aus hinlänglich diskutierten Gründen Linux in einer aktuellen Version verwendet (z.B. SuSE Linux 6.x). Zum Einsatz kommt eine aktuelle Version von Squid, z.B. eine 2.3er Version. Als Webserver dient Apache in einer 1.3.x Version mit der dazugehörigen Mod\_SSL Unterstützung. Als Hardwareminimum wird ein Pentium II mit 64 MB RAM und 4 GB Festplatte erwartet. Es wird für Netze mit mehr als 50 Clients mindestens ein 500Mhz Pentium III mit mindestens 256 MB RAM und beispielsweise 10GB Festplatte empfohlen.

## **Lösungsbeschreibung**

### **Einführung**

Die Standardauthentifizierung von Squid verwendet Authentication laut RFC2617. Squid und die meisten Browser unterstützen allerdings davon nur „Authtype Basic“, das bedeutet, die Passwörter werden im Klartext (Base64 codiert) übertragen. Besonders, wenn diese Passwörter die selben sind, wie sie von Benutzeraccounts verwendet werden (z.B. wenn diese gegen einen NT-Server validiert werden), ist dieser Zustand sehr ungünstig, da mit einem Packetsniffer dann auf einfachste Weise die Benutzerpasswörter gestohlen werden können. Die in RFC2617 vorgeschlagene Möglichkeit der „Digest“ Authentifizierung (Authtype Digest) ist in der Praxis nicht einsetzbar, da die gängigen Browser (Netscape Navigator 4 und Internet Explorer 4 haben zusammen einen Marktanteil von 90%) diese nicht unterstützen.

## **Idee**

Die sichere Passwortauthentifizierung kann man über ein CGI Programm machen, das auf einem SSL/TLS-fähigen WWW-Server läuft. Die Kennwörter werden dabei über die verschlüsselte SSL Verbindung übertragen. Bei der Verwendung des verbreiteten Webservers „Apache“ mit der exzellenten SSL/TLS Implementierung „mod\_ssl“ von Ralf Engelschall kann auch starke Kryptografie verwendet werden, so fern der Client das unterstützt. So kann sich dann die IP-Adresse des Clienten authentifizieren, daß der Client direkt auf den WWW-Server zugreifen kann (alle gängigen Browser unterstützen eine Liste mit Servern, zu denen direkt und nicht über den eingestellten Proxy zugegriffen wird). Dieses CGI Programm kontrolliert dann über einen sicheren Weg das vom Browser übermittelte Passwort, z.B. gegenüber der NT Domain, im Normalfall RC4-export verschlüsselt. Dann wird im Erfolgsfall dieser Client (genauer: dessen Netzwerkadresse) entsprechend vermerkt. Nun muß also vom Proxy nur noch überprüft werden, ob die IP-Adresse authentifiziert ist und vom Proxy Daten erhalten darf. Ist das nicht der Fall, wird statt der angeforderten Seite ein Anmeldebildschirm präsentiert.

## **Umsetzung**

Die Anmeldung über HTTP (Proxy Authentication laut RFC2617) funktioniert folgendermaßen:

Ein Browser fordert eine Seite an, bekommt (da kein Username/Passwort angegeben wurde) „407 - Proxy Authentication Required“ als Antwort (laut RFC 2068/2616). Der Browser öffnet ein Fenster, in das der Benutzer seine Zugangsdaten eingibt und dessen Inhalt dann leider im Klartext übertragen wird. Deshalb wurde eine Modifikation vorgenommen: Der kleine Patch ändert das Verhalten an zwei Punkten: einmal wird die IP-Adresse als „Name“ (genauer: „userid“ der „basic-credentials“ laut RFC2617), und das „Token“ IP\_AUTH als Passwordplatzhalter verwendet, falls kein Proxy-Auth-Header vorliegt, d.h. wenn der Client keine HTTP Proxy Authentication verwendet. Zum anderen wird anstatt „407 - Proxy Authentication Required“ ein „403 Forbidden“ gesendet, damit der Browser nicht mehr nach einem Usernamen fragt. Das mitgesendete Dokument enthält dann auch gleich ein HTML-Formular für die Anmeldedaten, über das ein CGI-Script auf einem SSL-Server gestartet wird. Dieses CGI-Script bekommt in einem versteckten Feld (INPUT TYPE=„HIDDEN“) die ursprünglich angeforderte URL mitübertragen und veranlaßt den Browser (über einen „Refresh-Header“), diese URL nach dem Laden des CGI-Scriptes erneut zu holen.

## **Arbeitstiere**

Squid verwendet nun das `authentication_program`, um die IP-Adressen überprüfen zu lassen. Beim ersten Zugriff schlägt dieses fehl, da diese IP-Adresse ja unbekannt ist, der Client erhält „403 Forbidden“. Dabei wird das Anmeldeformular angezeigt. Nach dem Ausfüllen klickt der Benutzer auf „LOGIN“, und so wird das CGI-Script gestartet. Dieses kann nun die übertragenen Passwörter überprüfen, z.B. in dem es diese verwendet, um sich eine Datei bei einem WindowsNT Server zu holen, um zu erkennen, ob dieser NT Account leseberechtigt auf die Datei und damit „surfberechtigt“ ist. Im vorliegenden Fall wurde dazu der `smclient` aus der Samba-Suite verwendet. Ist das erfolgreich, wird die IP-Adresse des Clients freigeschaltet, also den `authenticate_program` bekannt gegeben. Im einfachsten Fall macht man das über eine named pipe/FIFO. Das CGI-Script erzeugt einen Refresh-Header mit der ursprünglich angeforderten URL, damit der Browser nach der Anmeldung die richtige Seite lädt. Falls eine SSL/TLS-Verbindung aufgebaut werden sollte, funktioniert das jedoch so nicht. Der Name der geforderten Seite ist dem Proxy überhaupt nicht bekannt, dieser würde erst bei Aufgebauter Verschlüsselung übertragen. Ein möglicher Ausweg ist z.B. die Javascript-Funktion `history.go(-1)` zu verwenden. Dazu ist es allerdings erforderlich, daß der Clientbrowser Javascript unterstützt und aktiviert hat. Dann fordert der Browser also die ursprüngliche URL an, wird wiederum vom `authenticate_program` verifiziert, diesmal ist die IP bekannt (falls die Authentifizierung erfolgreich war), und der Client bekommt die gewünschten Daten.

## **Implementierung**

Verwendet man einfach Perl-Scripts für das CGI-Programm und das `authenticate_program`, kommt man zügig zum Ziel. Perl (in einer 5er Version) bietet sich an, da auf einen großen Sprachumfang zurückgegriffen werden kann. Zudem können Standardaufgaben, wie das Auswerten von Formulardaten, die über CGI übermittelt wurden, durch stabile Perl-Module vereinfacht werden. Perl ist stabil und schnell, und es ermöglicht zügiges Entwickeln, so daß geringere Kosten entstehen. Das `authenticate_program` kennt einen Hash der IP-Adressen, in dem ein Zeitstempel mitverwaltet wird. Das Perlscript verwendet `select(2)`, um die pipe und `stdin` gleichzeitig überwachen zu können (asynchrones I/O). Liegen Daten an der pipe an, so werden sie gelesen, die IP wird in den Hash gelegt bzw. dessen Timestamp aktualisiert oder aus diesem entfernt (falls es ein logout war). Daten, die über `stdin` vom Squid kommen, werden gelesen, die IP-Adresse wird im Hash nachgesehen, und akzeptiert, sofern sie nicht zu alt ist. Dabei wird wiederum der Timestamp aktualisiert. Login/Logout-CGI sind davon weitgehend unabhängig, nur die IP-Adressen müssen übertragen werden. Verwendet man eine named-pipe, können die IP-Adressen höchstens lokal gefälscht werden, was mit den Berechtigungen der FIFO entsprechend verhindert werden muß. Zu keinem Zeitpunkt werden sensitive Daten über das Netzwerk unverschlüsselt übertragen.

## **SSL-Server**

Zwei Scripte sollten über einen SSL/TLS-Server gestartet werden können: login.cgi und logout.cgi. Dieser Server muß erreichbar sein, ohne daß der Proxy verwendet wird, da ansonsten keine Authentification möglich wäre. Beim Netscape Navigator und dem Internet Explorer kann man einstellen, daß für bestimmte Hosts kein Proxy verwendet werden soll. Dieser SSL/TLS-Server soll auf der gleichen Maschine wie der Proxyserver laufen. Die Verwendung von Apache+mod\_ssl bietet sich hierbei an. Dabei ist es nicht zwingend erforderlich, ein richtiges SSL-Zertifikat zu besitzen. Ein Zertifikat einer Dummy-CA oder ein Testzertifikat sind hierzu ausreichend.

## **Distribution**

Für eine praxistaugliche Verwendung muß nun ein Distributionspaket verfügbar sein, da es unzumutbar wäre, für jede Installation den Quelltext des Squids zu modifizieren (patchen), neu zu compilieren und die benötigten CGI-Scripte downzuladen. Um diese Vorgänge zu automatisieren, empfiehlt sich die Verwendung von RPM (Redhat Package Manager), da als Zielsystem aus in der Presse ausreichend diskutierten Gründen Linux gewählt wurde. RPM ist unter den beiden führenden Distributionen RedHat und SuSE verfügbar und auf vermutlich alle Linuxdistributionen portierbar. Sollen andere Architekturen verwendet werden, so müssen diese eine Zielplattform von Squid sein und Perl verfügbar haben. Damit sind alle gängigen U\*NX Clones verfügbar. Unter Solaris kann z.B. mit sehr wenig Aufwand ein SUN-Binary-Package erzeugt bzw. angepaßt werden.

Außerdem ist es leicht möglich, ein Source-RPM Packet einer anderen Distribution so zu modifizieren, daß der Patch Verwendung finden kann.

## **Installation**

Die Installation des RPM Packetes ist sehr einfach, z.B.:

```
# rpm -i squid-IP_AUTH-2.3STABLE1.i386.rpm
```

Wird ein Source-RPM verwendet, so muß es kompiliert werden. Das ist auch sehr einfach, da alle Aktionen im „SPEC“ File des RPM's beschrieben sind, es ist nur ein:

```
# rpm --rebuild squid-IP_AUTH-2.3STABLE1.src.rpm
```

nötig, um es zu installieren und ein (Binär-) RPM für die aktuelle Architektur zu erzeugen. Eine SUN Solaris Binary Package könnte so installiert werden:

```
# pkgadd -d squid-IP_AUTH-2.3STABLE1.sparc.pkg squid-2.3STABLE1.sparc
```

Wird das RPM Packet installiert, so müssen die benötigten Beispielscripte aus contrib/IP\_AUTH an die gewünschten Orte kopiert werden. Diese funktionieren nur, wenn ein WWW-SSL/TLS-Server auf dem selben Host wie der Proxy läuft. Die Pfade bzw. URL's sind dann in der squid.conf und im Anmeldeformular einzustellen. Dabei ist zu beachten, daß das authenticate\_program nicht parallelelisierbar ist! Es muß also zwingend „authenticate\_children 1“ gesetzt sein!

Wird von einem Tarball installiert, sind folgende Schritte mindestens notwendig:

```
# tar -xzvf squid-2.3STABLE1.tgz
```

```
# cd squid-2.3STABLE1
```

```
# patch -p0 <./squid-IP_AUTH.patch
# ./configure --prefix=/usr/local/squid --enable-ip_auth
# make
# make install
```

Nun müssen die CGI-Scripte konfiguriert und installiert werden. Entweder liegen sie in einem Verzeichnis, das einen Script-Alias besitzt, oder für dieses Verzeichnis sind folgende Optionen in ".htaccess" gesetzt: "Options +ExecCGI" und "AddHandler cgi-script .cgi". Damit können dann CGI-Scripte dort gestartet werden, wo sie hingehören. Funktioniert das nicht, so ist zunächst zu prüfen, ob die Scripte von der Kommandozeile aus starten und ob bzw. welche sie Fehlermeldungen bringen. Dann ist im Serverlog zu prüfen, welcher Fehler sich ereignet hat, und dieser entsprechend zu korrigieren. In der Regel müssen die Scripte etwas angepaßt werden. Vorallem der Pfadname der FIFO-Datei zur Kommunikation mit dem authenticate\_program muß richtig gesetzt sein (die FIFO muß für die Scripte, also in der Regel für den Apache-User „www????“ schreibbar, für den Squid-User lesbar sein). Andere Benutzer dürfen keinen Zugriff auf diese FIFO erhalten, da ansonsten Passwörter gestohlen werden könnten, und das System gestört werden könnte! Zusammenfassend gesprochen sind die üblichen Probleme zu erwarten, die bei der Installation von CGI-Scripten auftreten, die wie gewohnt gelöst werden. Im CGI-Script wird auch die zu verwendende Sprache (momentan werden nur englisch und deutsch unterstützt) eingestellt werden. Natürlich muß auch das authenticate\_program dieselbe FIFO verwenden. Bei diesem muß zusätzlich ein Logfile konfiguriert werden. Dabei sollte dieses File überwacht werden, damit es nicht zu groß wird. Es darf dann nicht einfach gelöscht werden, sondern muß überschrieben werden (z.B. "cp /dev/null \$LOGFILE"). Andernfalls müßte der Proxy neugestartet werden, damit auch das authenticate\_program neu gestartet wird. In der Datei "ERR\_CACHE\_ACCESS\_DENIED" (in /usr/local/squid/etc/errors) muß dann ggf. die URL zum login.cgi CGI-Script angepaßt werden (die IP\_AUTH Contribution setzt voraus, daß es auf dem selben Host wie der Proxy läuft, und http://<proxy-server>/login.cgi heißt).

### ***Durchgeführte Tests***

Zunächst wurde das System in ein kleines Netz integriert (6 Clienten). Anstatt eines NT-Servers wurde ein Samba-Anmeldeserver verwendet. Das lief dann ohne Veränderungen mehrere Wochen, um die Wahrscheinlichkeit zu senken, daß später Probleme auftreten. Anschließend wurde mit speziell angefertigten Testprogrammen massive Last erzeugt, um die Stabilität und Robustheit zu testen. Dabei wurden nacheinander und parallel drei Belastungstestklassen gefahren:

1. Der Datendurchsatz durch den modifizierten Cache (mit gecachten und nicht gecachten Dokumenten) wurde getestet. Dafür wurde eine speziell angepaßte Version des Programms „Apache Bench“ erstellt und verwendet.
2. Das authenticate\_program wurde mit unverhältnismäßig vielen Daten versorgt, von denen zusätzlich ein Teil ungültig war. Hierzu wurde ein Simulationsprogramm erstellt, daß unter anderem Zufallsdaten und konstruierte Daten, bei denen Teile zufällig waren erzeugte. Diese wurden dann direkt in die Schnittstelle geschrieben.
3. Das Anmeldeprogramm wurde mit sinnvollen und sinnlosen Daten versorgt, richtige und fehlerhafte Authenticationsversuche unternommen. Dazu wurde ebenfalls ein spezielles Simulationsprogramm verwendet.

# Authentifikation am Squid Proxy aus Anwendersicht

## **Authentifikation aus Sicht des Benutzers**

1. Anmeldung am Windows-System (9x, NT, Novell)
2. Starten des Webbrowsers
3. Bei dem erstem Aufruf einer Webseite wird der Benutzer aufgefordert sich für den Internet-Service anzumelden. Er authentifiziert sich mit dem gleichen Benutzernamen und Passwort wie bei der Windows-Anmeldung. Die Passwortpflege unterliegt somit dem Windows Netzwerksystem. Die Berechtigung gilt so lange, bis der Benutzer für eine bestimmte Zeit keine Aktionen im Internet durchgeführt hat. Nach dieser Periode muß sich der Benutzer erneut authentifizieren.
  - Erfolgreiche Anmeldung: der Benutzer kann sich frei im Internet bewegen (HTTP, FTP).
  - Fehlerhafte Anmeldung: der Benutzer erhält eine Fehlermeldung und kann versuchen, sich erneut anzumelden.

## **Authentifikation aus Sicht des Administrators**

- Der Administrator legt auf einem beliebigen Computer im System eine vordefinierte Freigabe an (z.B. netlogon). In dieser Freigabe existiert ein File (z.B. proxyauth) mit einem Textinhalt (z.B. allow).
- Um verschiedene Benutzer zur Nutzung des Internets zu berechtigen, muß der Administrator den Benutzern Leserechte auf das o.g. File erteilen (proxyauth). Dazu verwendet er einfach Windows-NT ACL's (Access Control Lists).

# Bewertung unter verschiedenen Gesichtspunkten

## **Installations-, Konfigurationsaufwand**

Wird ein System verwendet, das zu dem erstellten RPM paßt, ist der Installationsaufwand gering. Es muß lediglich das RPM installiert werden und die CGI Scripte müssen kopiert werden. Andere RPM fähige Linuxmaschinen müßten das RPM neubauen, was nur etwas Rechenzeit benötigt. Nicht RPM fähige Maschinen müssen die Quellen neu kompilieren. Dort ist der Aufwand so groß wie eine „normale“ Squid-Installation plus Kopieren und evtl. Anpassen der CGI Scripte.

Der Aufwand ist nur mit umfangreicher Fleißarbeit senkbar, was keinem günstigen Kosten-Nutzenverhältnis entspricht, und unpraktikabel erscheint. Zudem müßte das System dann untern anderem Pfade und URLs „raten“. Das ist nicht erwünscht.

## **Sicherheit**

Zunächts ist der Vergleich mit einer normalen Squid-Konfiguration interessant. Das System verwendet exportbeschränkte oder starke Kryptographie, je nach Fähigkeiten des Clienten, damit ist die Übertragung der Passwörter zum Login-Prozeß ungleich sicherer als die Windows-NT Authentifizierung. Ein potentielles Problem wäre ein Angriff von der

Squidmaschine aus. Es soll aber auf solch dedizierten Servern grundsätzlich keine Benutzer und nicht benötigte Dienste geben. Auf Grund des Overheads, der durch die SSL/TLS Handshakes und durch die remote-Authentifizierung entsteht, dauert eine Passwortverifikation relativ lange (in der Praxis zeigten sich Werte um die zwei Sekunden). Eine „Brute-Force-Attacke“ über den Loginprozeß ist selbst bei der Verwendung von exportbeschränkten Verschlüsselungen ungleich aufwendiger und langwieriger als gegen geschnittene Windows-Passwörter. Ein Denial-Of-Service Angriff kann die Anmeldegeschwindigkeit stark verschlechtern oder gar zum Erliegen bringen. Damit fallen aber keine weiteren Dienste aus. Zusätzlich ist auch diese Möglichkeit nur intern durchführbar, kann demzufolge besser verfolgt und abgewehrt werden; die erzeugten Logfiles liefern dazu mehr als ausreichend Informationen. Erfolgreiche IP-Spoof-Attacken sind nicht zu erwarten, da ein Handshake zwingend notwendig ist, und auch nur eine gefälschte IP Adresse erlaubt werden könnte, dabei bleibt das darunterliegende Betriebssystem westentlich gefährdeter. Setzt man eine aktuelle Linuxdistribution voraus, die entsprechend konfiguriert ist, sind insgesamt keine erfolgreichen Attacken bis auf die skizzierte Denial-Of-Service Attacke zu erwarten.

### ***Portierbarkeit***

Squid ist auf vielen U\*NX Plattformen lauffähig, darunter Solaris und IRIX. Perl ist ebenfalls auf sehr vielen Plattformen verfügbar. Damit sind die beiden Hauptabhängigkeiten erfüllt, ohne daß Portierungsaufwand getrieben werden müßte. Der Apache Webserver mit mod\_ssl ist ebenfalls auf verschiedensten Architekturen installierbar. Bei der Verwendung eines anderen ähnlich leistungsfähigen Webserver sind möglicherweise geringe Modifikationen an den CGI-Programmen erforderlich, die sich größtenteils auf Enviroment-Variablenamen beziehen, was in der entsprechenden Serverdokumentation erwähnt sein wird, so daß auch diese Aufgabe überschaubar klein bleibt.

### ***Wartbarkeit***

Ein automatisches Rotieren der Logfiles sollte über einen Cronjob erledigt werden. Zusätzlicher Pflegeaufwand für die Accounts entsteht nicht, da diese aus der NT Domain stammen. Wird der SSL/TSL-Server nur für diesen Zweck eingesetzt, ist zu beachten, daß die Zertifikate u.U. alle ein bis zwei Jahre gewechselt werden müssen. Ansonsten entsteht durch diese Proxy-Erweiterung kein zusätzlicher Pflegeaufwand.

### ***Erweiterbarkeit***

Da das System in Login (mit der eigentlichen Authentifizierung), Logout und AuthenticationServer unterteilt ist, ist eine Erweiterbarkeit überschaubar möglich. Sollen andere Passwortquellen wie z.B. LDAP verwendet werden, muß lediglich das Loginprogramm modifiziert werden. Eine Isolation des hier in das authenticate\_program integrierten AuthenticationServers (z.B. auf eine andere Maschine) ist mit einem Aufwand möglich. Dabei ist der Gesamtaufwand in etwa gleich der Entwicklung des neuen, unabhängigen AuthenticationServers und einem neuen authenticate\_program, da die Schnittstelle (über die name PIPE) eine einfache und harte Schnittstelle ist. Diese kann verwendet werden und auch über einen Socket gelegt werden, der netzwerkweit gültig ist.

Diese Entwicklungen wären in diesem Falle ja sowieso erforderlich. Das entspricht aber einer ganz anderen Aufgabenstellung.

### **Skalierbarkeit**

Das System skaliert leider wesentlich schlechter als andere nicht-authentifizierende Proxies, da DNS-Roundrobbing nicht einsetzbar ist. Der User müßte sich dann jeweils an *jedem* Proxy anmelden, die Abmeldung würde nicht funktionieren. Das authenticate\_program läßt sich in der vorliegenden Form nicht parallelisieren. Für Netzwerke, deren Größe einer derartigen Skalierung bedürftig ist, wird von dieser Form der Authentifikation aber unabhängig davon abgeraten, da hierfür Windows-NT als Authentifizierungsursprung nicht geeignet ist. Hier müßte nach besser skalierenden Lösungen wie LDAP, NDS oder ActiveDirectory (soweit verfügbar) gesucht werden. Mit gängiger PC Hardware sollten je nach „Surfverhalten“ Firmennetze bis etwa eintausend Benutzern beherrschbar sein, da Arbeitsplatzrechner in einem Unternehmen nur einen kleinen Teil der Zeit mit Authentifizierung und Internetdownloads verbringen. Gerade in Deutschland verfügen viele Firmen nur über E1 oder E3 Anbindungen (mit zwei bzw. 34Mbit/sec Bandbreite), was vom Squid spielend verwaltet werden sollte, da er mit Datendurchsätzen bis etwa 100Mbit/sec arbeiten können sollte, richtige Konfiguration vorausgesetzt.

### **Weiterführende Literatur:**

- RFC1945 - Hypertext Transfer Protocol -- HTTP/1.0
- RFC2068 - Hypertext Transfer Protocol -- HTTP/1.1 (obsoleted by 2616)
- RFC2616 -Hypertext Transfer Protocol -- HTTP/1.1 (obsoletes: 2068)
- RFC2617 - HTTP Authentication: Basic and Digest Access Authentication
- RFC2246 - The TLS Protocol Version 1.0
- Rivest, R., „A Description of the RC2(r) Encryption Algorithm“, RFC 2268, 1998.
- R. Rivest, A. Shamir, and L. M. Adleman, „A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,“ Communications of the ACM, 1978
- The issued U.S. Patent No. 5,657,390 („the SSL Patent“)
- Eilebrecht, Lars: „Apache Web- Server“ MITP, 501 S., ISBN: 3-8266-0438-5
- Gundavaram, Shishir: „CGI-Programming in Perl.“ O'REILLY, 450 S., ISBN: 1-56592-419-3
- Patchett, Craig Wright, Matthew: „CGI/ Perl Cookbook.“, WILEY-VCH, 624 S., ISBN: 0-471-16896-3
- Pearson, Oskar: „Squid Users Guide“, <http://www.squid-cache.org/Doc/Users-Guide/>
- Blair, John D: „Samba, Integrating UNIX and Windows“, ISBN 1-57831-006-7
- Dr. Borkner-Delcarlo, Olaf: „Das Samba-Buch“, SuSE Press, ISBN 3-930419-93-9