gangsprofile (welche Zeiten, welche Dienste, welche Rechner) erstellen, zum anderen kann man die Festlegung der zulässigen Verbindungen anwendungsbezogen vornehmen. Die daraus resultierenden separaten kleinen Regelsätze bleiben besser überschaubar als der komplexe Regelsatz eines Paketfilters. Application Level Gateways sind typische Vertreter der 'Verboten-was-nicht-ausdrücklich-erlaubt'-Strategie und als die sicherste, aber auch aufwendigste Lösung einzuschätzen.
Da beim Proxy alle Zugriffe nach außen über eine Instanz laufen, kann man den Proxy gleichzeitig als **Cache** benutzen. Der Proxy speichert alle erhaltenen WWW-Seiten zwischen, so daß er bei einem erneuten Zugriff darauf - egal, ob vom selben oder einem anderen Anwender - keine Verbindung nach außen aufbauen muß.
Der Einsatz von Firewalls bietet sich auch innerhalb einer Organisation an, um Bereiche unterschiedlicher Sensitivität von einander abzugrenzen. Firewalls bieten jedoch niemals hundertprozentige Sicherheit. Sie schützen nicht vor dem Fehlverhalten eines authorisierten Anwenders und können, etwa durch eine zusätzliche Modem-Verbindung, umgangen werden.
Ein Firewall nach dem heutigen Stand der Technik kann nur gegen bereits bekannte Angriffs-Methoden schützen. Er kann möglichst viele Ereignisse protokollieren, um im Fall des Falles den Vorgang möglichst lückenlos rekonstruieren zu können. Die Datenmengen können pro Tag auf hunderte von Megabytes anwachsen. Wer wertet dies aus, und sucht darin die Nadel im Heuhaufen?
Neueste Ansätze beruhen darauf, daß man ein Firewallsystem durch ein zusätzliches System ergänzt, welches den Verkehr auf dem Netzwerk überwacht. Dieses System ist vom Firewall völlig unabhängig und greift nicht aktiv in den Datenverkehr des Netzes ein. Dieses System hat die Aufgabe eines unparteiischen 'Flugschreibers' (der Name 'Network Flight Recorder' wurde von Marcus Ranum, dem Entwickler des TIS Firewall Toolkit, geprägt). Dieses System kann den Bedürfnissen entsprechend konfiguriert werden. Mit Hilfe dieses Systems kann man zwar einen Einbruch in das Netz nicht aktiv unterbinden, man kann ihn aber leichter erkennen und verfolgen. Das System kann, genauso wie ein Firewall, Alarme bei suspekten Ereignissen auslösen, z.B. wenn plötzlich DNS-Anfragen von einem System beantwortet werden, welches kein DNS-Server ist (DNS-Spoofing vgl. Seite 161). Diese **Intrusion Detection Systeme** werden in Kapitel 10 näher behandelt.

8.2 Proxy Gateways

Proxies sind wie Überwachungsrouter dedizierte Rechner, über die alle Verbindungen zwischen dem internen Netz und dem Internet geleitet werden. Im Unterschied zu den Paketfiltern trennen sie jedoch die Verbindungen am Übergang zwischen den Netzen.
Bei einem Verbindungsaufbau eines internen Clients wird zuerst überprüft, ob er den adressierten Zielhost kontaktieren und den angeforderten Dienst nutzen darf. Danach führt der Proxy den Verbindungsaufbau stellvertretend für den Client aus. Der Zielrechner empfängt die Datenpakete mit der Absenderadresse des Proxies, an die er dann auch die Antworten zurückschickt. Der Proxy reicht die Daten dann nur noch über sein internes Interface an

den Client weiter. Bei Verbindungen vom LAN ins Internet tritt der Proxy als stellvertretender Server für alle Arbeitsplätze im internen Netz auf und gibt sich selbst wieder als Client gegenüber dem kontaktierten Server im externen Netz aus. Da die Zustellung von IP-Paketen ausschließlich von den Proxies durchgeführt werden darf, muß die betriebssysteminterne Routingfunktion des Proxies deaktiviert werden.
Dieses Verfahren hat gegenüber der Paketfilterung einige Vorteile. Alle im Unternehmensnetz verwendeten IP-Adressen sind nach außen hin nicht sichtbar. Die interne Netztopologie bleibt verborgen. Der Proxy versieht alle weitergeleiteten Datagramme mit seiner eigenen, externen Absenderadresse und ordnet die empfangenen Daten anhand interner Verbindungstabellen wieder den richtigen Anwendersystemen zu. Dabei kann eine vorher definierte, statische 1:1-Umsetzung zwischen internen und externen Adressen erfolgen oder eine dynamische Vergabe auf Basis von IP-Adressen in Verbindung mit Portnummern (vgl. IP-Hiding oder Masquerading, Abschnitt 4.3.2).
Obwohl kurzfristige Lösungen wie **Classless Inter-Domain Routing** [42] und die nächste Generation des Internet-Protokolls [43] die Knappheit von international gültigen IP-Adressen in den Griff bekommen sollen, scheint die Adressumsetzung nicht nur aus den genannten Sicherheitsgründen ein vielversprechendes Verfahren für viele Unternehmen zu sein. Auf diese Weise läßt sich mit nur einer Adresse allen Rechnern im internen Netz der Zugriff auf das externen Netz ermöglichen. Im internen Netz sind keine Änderungen an der Adressierung notwendig. Im Idealfall sollten im internen LAN die registrierungsfreien IP-Adressen nach RFC 1918 [RMK$^+$ 96] verwendet werden. Nachteilig auf die Performance wirkt sich der größere Verwaltungs-Overhead solcher Systeme aus.
Unter den Proxy Gateways wird noch einmal zwischen zwei Grundtypen unterschieden: den **Circuit Level Gateways**, die auf der Transportschicht den Datenstrom observieren (wird in Abschnitt 9.1 näher erklärt) und den auf der Anwendungsschicht aktiven **Application Level Gateways** (siehe Abschnitt 8.3, [Raep 98], [Fuhr 98]).

8.3 Application Level Gateways

Dieser Typus arbeitet auf der Anwendungsschicht. Application Level Gateways sind in der Lage, neben den von Paketfiltern und Circuit Level Gateways observierten Schichten drei und vier auch anwendungsspezifische Informationen in den Filterprozeß mit einzubeziehen. Hierzu gehören z.B. Authentisierungsverfahren oder spezifische Befehle der Protokolle der Anwendungsschicht. Ein Application Level Gateway verwendet zur Filterung in der Regel mehrere Proxyprozeße. Aus diesem Grund bieten Application Proxies den größtmöglichen Schutz gegen externe und interne Angriffe. Sie erlauben dem Firewalladministrator z.B., die Ausführung bestimmter Befehle in den Anwendungen zu unterbinden. So kann verhindert werden, daß Anwender Daten per FTP aus dem internen Netz mit dem Befehl PUT auf einem externen Server ablegen oder via HTTP fragwürdige Webseiten im Internet besuchen. Es werden keine Verbindung durchgelassen. Vielmehr endet die hereinkommende

[42] RFC 1519 [FLYV 93]
[43] IPv6

Verbindung auf dem Gateway und das Gateway baut eine weiterführende Verbindung auf. Sodann vermittelt das Gateway auf dem Application Level (daher der Name) zwischen den Verbindungen. Wenn man so will, betätigt es sich als Übersetzer. Dies hat den Vorteil, dass eine Menge Angriffe inhaltlicher Art erkannt und abgewehrt werden können.
Viele Proxies sind so intelligent, daß sie häufig angeforderte Daten aus dem Internet im lokalen Cache zwischenspeichern. Dadurch verringert sich die Zugriffszeit für interne Clients erheblich, da die Daten unmittelbar über die schnellen LAN-Verbindungen verfügbar sind. Geeignete Dienste für das Caching sind z.B. HTTP- oder FTP-Proxies, da die mit diesen Protokollen übertragenen Daten häufig sehr umfangreich sind und eine wiederholte Übertragung viel Zeit und Geld kostet. Es gibt aber mit der Zeit immer mehr Seiten, die nicht gespeichert werden können, da sich ihre Inhalte dynamisch zusammensetzen.
Da Application Level Gateways die Verbindungen auf Anwendungsschicht weitervermitteln, lassen sich auch sehr flexible Mechanismen zur Benutzerauthentisierung implementieren. Von der einfachen Form der Passwortabfrage bis hin zum Challange/Response-Verfahren mit Hardware Token wie z.B. S/Key oder SecurID reichen die Möglichkeiten vieler Produkte. Für jeden über die Policy freigegebenen Dienst muß allerdings ein entsprechender Proxy auf dem Firewall vorhanden sein. Dies hat zur Folge, daß oftmals neue Internetanwendungen nicht sofort unterstützt werden können, weil der passende Anwendungsproxy noch nicht verfügbar ist.
Bei einem Verbindungsaufbau muß sich der Anwender zuerst mit dem Proxy verbinden und diesem die notwendigen Informationen für eine korrekte Authentisierung (falls verlangt) und die Adresse des eigentlichen Ziels mitteilen. Anschließend baut der Proxy eine Verbindung zum Ziel auf und überträgt die Daten des Anwenders. Da für die Verbindung vom Proxy ins Internet immer die Adresse des Application Level Gateways verwendet wird, da die Verbindung am Proxy terminiert und neu aufgebaut wird, findet automatisch eine Adressumsetzung statt, die bei einem Paketfilter erst durch Zusatzfunktionen erreicht werden kann.
Der Ausdruck Proxy beschreibt die Tatsache, daß eine Kommunikation zwischen zwei Rechnern immer mit Hilfe eines Prozeßes auf dem Application Level Gateways stattfindet. Dieser Proxyprozeß übernimmt aus Sicht des Anwenders die Funktion des Servers und entsprechend aus der Sicht des Servers die des Clients. Der Proxy stellt also stellvertretend für den Anwender eine Verbindung mit dem Server im Internet her und kann so eine Reihe von Informationen verbergen. Diese Eigenschaft wird in Abbildung 70 deutlich.

Ein großer Vorteil eines Application Level Gateways ist die Möglichkeit einer **Nutzdatenanalyse**. Unter Nutzdaten versteht man in diesem Zusammenhang alle Daten, die sich nicht in den Headern der Datenpakete befinden, also z.B. Daten in einer Mail oder einer Webseite. Diese Daten können von einem Application Level Gateway analysiert und nach gewissen Schlüsselwörtern durchsucht werden. So bieten einige HTTP-Proxies die Möglichkeit, alle Zeilen innerhalb einer HTML-Seite, die zu einem Java-Applet gehören, zu blockieren.
Man darf bei der Planung eines Firewalls nicht vergessen, daß manche Protokolle für den Einsatz von Proxies besser geeignet sind als andere. So sind z.B. die Store-and-Forward-

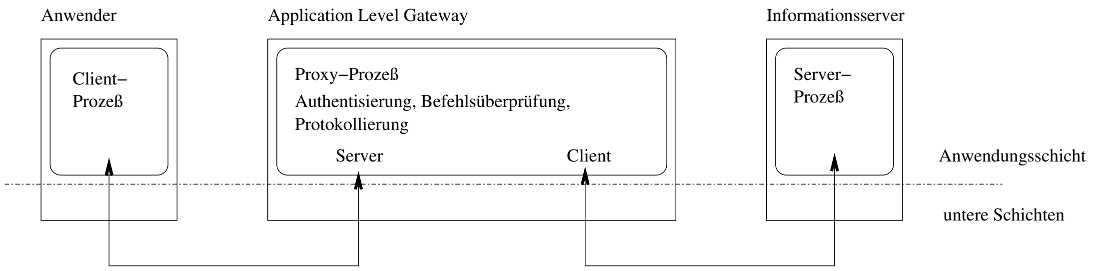

Abbildung 70: Funktionsweise von Proxies

Protokolle SMTP und NNTP sehr gut geeignet, da die Daten von dem eingesetzten Server in der Regel kurzfristig gespeichert werden, bevor sie an den Empfänger ausgeliefert oder von diesem abgeholt werden. Bei anderen Protokollen ist der Einsatz von Proxies und damit die generelle Nutzung über einen Application Level Gateway nicht empfehlenswert.
Besonders wichtig ist es, daß auf einem Application Level Gateway keine überflüssige Software installiert ist. Jede Weiterleitung eines Paketes von einem Interface des Application Level Gateway zum andern darf nur durch einen Proxy durchgeführt werden. Es darf also nicht die Möglichkeit einer Weiterleitung innerhalb des Betriebssystems gegeben sein, d.h. das sogenannte IP Forwarding muß abgeschaltet sein. Die Verwendung eines Minimalsystems verhindert auch, daß ein Angreifer eigene Software auf dem Application Level Gateway kompilieren und installieren kann.
Folgende Eigenschaften charakterisieren einen Application Level Gateway und können in Abhängigkeit von den Sicherheitsanforderungen als Vor- oder Nachteil angesehen werden.

- Da die Analyse der Informationen auf der Anwendungsschicht stattfindet, besteht die Möglichkeit einer sehr umfangreichen Protokollierung.
- Ein Application Level Gateway kann eine Authentisierung der Benutzer vornehmen. Dienste können also benutzerabhängig erlaubt oder verboten werden, wobei natürlich auch die Einrichtung von Benutzergruppen möglich ist.
- Die Verbindungen zwischen dem zu schützenden Netz und dem Internet wird durch ein Application Level Gateway völlig entkoppelt. Alle Daten werden vom Proxyprozeß entgegengenommen und analysiert. Erst wenn keine Regelverletzung vorliegt, werden die Daten mit Hilfe einer neuen Verbindung ins Internet oder Intranet weitergeschickt.
- Für ein Protokoll, für welches kein Application Level Proxy existiert, können Circuit Level Gateways eingesetzt werden.
- Korrekt arbeitende Application Level Gateways stellen aufgrund ihrer Funktionsweise sicher, daß nur Dienste erlaubt sind, die explizit durch den Einsatz von Proxies gesichert werden, alle anderen Dienste sind verboten.
- Application Level Gateways können fragmentierte IP-Datagramme korrekt filtern.

- Application Level Gateways filtern Fehler im Protokoll heraus, da sie das Protokoll sprechen und somit Unregelmäßigkeiten erkennen.

Viele der Möglichkeiten, die ein Eingreifen in die Protokolle und eine vermehrte Überwachung ermöglichen, wirken sich negativ auf die Performance der Systeme aus.
Viele Application Level Gateways lassen sich im sogenannten Transparentmodus betreiben. D.h. sie sind für das Anwenderprogramm nicht sichtbar, es müssen also keine Adressen für den Proxy eingetragen oder andere Veränderungen an den Clients vorgenommen werden. Der transparente Firewall wird in der Regel für ausgehende Verbindungen analog zu einem Paketfilter über entsprechende Routingeinträge als Übergang in das Internet konfiguriert, und alle Pakete, die er auf diese Weise erhält, werden entsprechend den eingestellten Filterregeln verarbeitet. Für ankommende Verbindungen ist der Application Level Gateway natürlich nicht transparent, da sonst alle internen IP-Adressen im Internet bekannt sein und geroutet werden müssen. Der Nachteil dieser Methode besteht darin, daß viele der Eigenschaften eines Application Level Gateways (z.B. Benutzerauthentisierung) nicht mehr zur Verfügung stehen, da ein Firewall im Transparentmodus für die Anwender im zu schützenden Netz gerade unsichtbar sein soll.
Neuere Entwicklungen gehen dazu über, die Proxyfunktionalität aus Geschwindigkeitsgründen nicht mehr durch eigenständige Prozeße ausführen zu lassen, sondern speziell angepaßte Treiber und Betriebssysteme zu verwenden.

8.3.1 Squid

Squid ist ein freies und sehr flexibles Internet-Proxy-Caching Programm. Squid ist standardmäßig bei den meisten Unix Derivaten als prekompiliertes Paket vorhanden. Außerdem kann der Quellcode jederzeit, evtl. mit Parameteränderungen, neu kompiliert werden. Anleitungen und den Quellcode von Squid finden Sie unter [squ 02a].
Squid arbeitet als Agent, der Clientanfragen, die z.b. über einen Browser gestellt werden, an den angefragten Internet Web-Server weiterleitet. Eine Kopie der vom Server übertragenen Daten werden im Squid-Cache gespeichert. Diese Cachingfunktion ermöglicht eine Reduzierung des Internettraffics zwischen Proxy und Internetservern, da bei einer erneuten Anfrage der Inhalte nur noch eine Überprüfung auf Aktualität der Inhalte beim Web-Server durch den Proxy erfolgt und die Daten direkt aus dem Cache zum client geliefert werden, wenn sich an den Daten nichts verändert hat. Im wesentlichen werden die Caching-Objekte so lange im Cache gespeichert bis die Aktualität der Objekte obsolet wird. Eine genauere Beschreibung der Cachingalgorithmen finden Sie unter [Wess 01]. [squ 02b] ist ein sehr gutes Squid Manual.
squid hat nur eine einzige Konfigurationsdatei: `/etc/squid.conf`. Die Logfiles liegen standardmäßig unter `/var/log/squid`. Die Pfade können aber jederzeit im Konfigurationsfile geändert werden.
Zur Grundkonfiguration können die meisten Standardparameter übernommen werden (z.B. Tuningparameter, Einbinden externer Libraries und Programme, wie z.B. zur Userauthentisierung oder URL-Filterung).

Hier soll noch kurz der Unterschied zwischen einer Server- und einer Proxyanfrage erklärt werden. Wie wir bereits in Abschnitt 7.2 gesehen haben, übermittelt der Abfragenstellen bei einer Serveranfrage den Pfad und die Datei, die geladen werden soll:

```
telnet www.sun.de 80
Trying 212.125.100.80...
Connected to www.sun.de.
Escape character is '^]'.
GET / HTTP/1.1
```

Wird nun die gleiche Anfrage über einen Proxy gestellt, so stellt der Browser zum Proxy eine Proxyanfrage und der Proxy zum Server die oben beschriebene Serveranfrage. Hier ein Beispiel für eine Proxyanfrage:

```
telnet proxy.my-domain.com 3128
Connected to proxy.my-domain.com.
Escape character is '^]'.
GET http://www.sun.de HTTP/1.1
```

Hier nun die wichtigsten Parameter:

Basiskonfiguration

Mit diesen Parametern wird die Grundfunktionalität des Squid konfiguriert:

- Tag `http_port` legt fest, auf welcher Socketadresse Squid auf Clientanfragen hört. Das kann auf drei verschiedene Arten geschehen: Port, IP-Adresse und Port oder Name und Port. Somit wird Squid den Socket auf alle Interfaces oder auf eine bestimmte Adresse mit dem Port binden. Der Defaultport ist 3128: z.B. `http_port 10.10.10.10:3128`.
- Tag `icp_port` legt den Port fest, über den Squid ICP[44]-Anfragen rausschickt und erwartet. ICP ist ein UDP-Protokoll, das zum Austausch von Cache-Inhalten zwischen Proxies dient [45].
- Tag `tcp_outgoing_address` legt fest, mit welcher Absenderadresse HTTP Anfragen nach außen gehen.
- Tag `cache_effective_user` legt den User fest, unter dem der Squid auf dem System läuft. Um Squid mit einem Port < 1023 laufen lassen zu können, muß Squid als `root`

[44] Internet Cache Protocol

[45] RFC 2186 [WeCl 97a] und RFC 2187 [WeCl 97b]

Prozeß gestartet werden. Dannach gibt es aber keine guten Grund mehr, Squid auch weiterhin unter `root` laufen zu lassen. Deshalb wird hier ein User mit weniger Rechten definiert, unter dem Squid dann läuft.

- Tag `cache_effective_group` legt die Gruppe fest, unter dem der Squid auf dem System läuft.

- Tag `cache_dir`: In dieser Option können mehrere Verzeichnisse festgelegt werden, unter denen der Cachebereich aufgeteilt werden soll (evtl. über verschiedene Partitionen hinweg). Es können die Tiefe der Verzweigung und die Größe des Caches definiert werden. `ufs`, `aufs` oder `diskd` sind die verwendbaren Speichertypen. Standard ist `cache_dir ufs /var/cache/squid 100 16 256`, wobei `/var/cache/squid` das Cacheverzeichnis festlegt, `100` ist die Größenangabe in Megabytes, bis zu welcher Datenmenge im Cachebereich gespeichert wird. `16` ist die Anzahl der Verzeichnisse in der ersten Verzweigung und `256` die in der Zweiten. Durch diese Verzweigung soll der Zugriff auf die Platte beschleunigt werden, da es schneller geht, ein File über einen Verzeichnisbaum hinweg zu suchen, als ein file in einem Verzeichnis mit mehreren Millionen Files zu finden.

- Tag `cache_access_log` legt Ort und Namen des Accesslogfiles für Squid fest. Hier werden die Aktivitäten der Clients mitprotokolliert. Für jeden HTTP oder ICP Request ist hier ein Eintrag enthalten.

- Tag `cache_mgr` legt die E-Mailadresse fest, an die Fehlermeldungen verschickt werden. Die hier definierte Adresse wird auch am Ende von Fehlermeldungsseiten, die an den Client zurückgeschickt werden, angezeigt.

- Tag `cache_log` legt Ort und Namen des Cachelogs fest. Hier werden generelle Informationen zum Verhalten des Caches abgelegt.

- Tag `cache_store_log` legt Ort und Namen des Storelogs fest. Dieses Logfile zeigt das Verhalten des Speichers an, d.h. welche Daten gespeichert wurde, für wie lange sie gespeichert sind, etc.

- Tag `pid_filename`: Unter diesem Pfad und Namen ist das PID File abgelegt.

- Tag `ftp_user`: Dieses Passwort wird für anonymous-Login bei FTP benutzt und sollte auf etwas Sinnvolles in der eigenen Domain gesetzt sein. In diesem Fall wird FTP vom Browser zum Proxy über den HTTP-Port gesprochen. Erst der letzte Proxy in der Kette baut die FTP-Verbindung (vgl. 6.3.1 zum FTP-Server auf.

- Tag `request_header_max_size` legt die maximale Länge des HTTP Headers innerhalb eines Requests fest. Dadurch können eventuelle Buffer-Overflow-Attacken verhindert werden.

- Tag `request_body_max_size`: hier wird die maximale Größe des HTTP Request Body's definiert. In anderen Worten wird die Größe von PUT und POST Requests festgelegt.
- Tag `reply_body_max_size` beschränkt die Größe von Downloads.
- Tag `authenticate_program` legt den Pfad des exteren Authentisierungsprogramms fest.

Access Control Lists und ihre Anwendung

Mit ACLs[46] werden Listen definiert, die dann in verschiedenen Tags verwendet werden können. Somit können Zugriffsberechtigungen definiert werden. Die Zeilen mit `acl` zu Beginn werden `acl classes` genannt, die Tags, in denen ACLs angewandt werden, wie z.B. `http_access` oder `icp_access`, heißen `acl operators`. `acl operators` werden der Reihe nach, wie sie in der Konfiguration stehen, abgearbeitet. Sobald ein `acl operators` zutrifft, wird die in diesem Operator beschriebene Aktion ausgeführt und die Liste der `acl operators` verlassen. Trifft keiner der `acl operators` zu, so wird anhand des letzten `acl operators` in der Reihe festgelegt, welche Aktion für diesen Request ausgeführt wird. Es wird immer das Gegenteil der letzten Regel angewendet. Somit sollte immer eine letzte Regel definiert werden, die alle vorher noch nicht definierten Requestkombinationen abdeckt, um hier Konfusion zu vermeiden.

- Tag `acl` sind Access Control Lists: `acl aclname acltype string1 string2 string3`
  - `acl aclname src ip-address/netmask`: definiert ACLs, die die Quell-IP-Adressen betreffen.
  - `acl aclname dst ip-address/netmask`: definiert ACLs, die die Ziel-IP-Adressen betreffen.
  - `acl aclname srcdomain .domain.de`: betrifft alle Quellrechner, die sich laut DNS Auflösung in dieser Domain befinden.
  - `acl aclname dstdomain .domain.de`: betrifft alle Zielrechner, die sich laut DNS Auflösung in dieser Domain befinden.
  - `acl aclname port 80 70 21`: definiert eine ACL für Ports.
  - `acl aclname proto HTTP FTP`: definiert eine ACL für Dienste.
  - `acl aclname method GET POST`: legt ACLs für Methoden fest.
  - `acl aclname proxy_auth username`: legt eine ACL fest, bei der erlaubte Usernamen aufgeführt werden. Mit `REQUIRED` statt `username` wird jeder gültige, authentisierte User akzeptiert. Soll eine externe Authentisierung erfolgen, so muß auch `authenticate_program` konfiguriert werden.

[46] Access Control Lists

- Tag `http_access` legt anhand von ACLs fest, unter welchen Voraussetzungen der Proxydienst benutzt werden darf.

```
acl all src 0.0.0.0/0.0.0.0
acl SSL_ports port 443 563
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443 563     # https, snews
acl Safe_ports port 70          # gopher
acl Safe_ports port 210         # wais
acl Safe_ports port 1025-65535  # unregistered ports
acl Safe_ports port 280         # http-mgmt
acl Safe_ports port 488         # gss-http
acl Safe_ports port 591         # filemaker
acl Safe_ports port 777         # multiling http
acl CONNECT method CONNECT
acl allowed-network 10.10.10.0/255.255.255.0

http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow allowed-network
http_access deny all
```

  Diese Definition sorgt dafür, daß nur die unter der ACL `Safe_ports` angegebenen Ports als Zielport erlaubt sind. Die HTTP-Methode CONNECT darf nur auf Zielports angewandt werden, die unter `SSL_ports` definiert sind. Außerdem ist die Benutzung des Proxies nur für IP-Adressen aus dem Bereich `10.10.10.0/24` erlaubt. Die Verknüpfung von mehreren ACLs in einem `acl operator` erfolgt in einem logischen AND.

- Tag `icp_access`: legt die IP-Adressen fest, die ICP Anfragen stellen dürfen.

- Tag `miss_access`: Hiermit können Nachbarcaches dazu gezwungen werden, den Cache als Sibling und nicht als Parent zu verwenden (Erklärungen hierzu finden Sie auf Seite 221). D.h. es wird nur den lokalen Clients erlaubt, `MISSES`[47] zu holen, alle anderen dürfen nur `HITS` [48] abfragen. Es ist zu beachten, dass zuerst alle `http_access` Tags abgearbeitet werden, bis die Tags für `miss_access` überprüft werden.

- Tag `always_direct`: Hier können mittels ACLs Elemente definiert werden, die IMMER direkt zum Webserver geschickt werden sollen. Das würde für die lokale Domain so aussehen:

[47] ICP_MISS besagt, daß das angefragte Objekt nicht im Cache ist

[48] ICP_HIT wird zurückgesandt, wenn das angefragte Objekt im Cache gespeichert ist

```
acl local-servers dstdomain .my.domain.net
always_direct allow local-servers
```

Diese Option ist aber **nicht** die positive Entsprechung zu `never_direct`. Wenn ich also

```
acl external dstdomain .foo.net
always_direct deny external
```

verwende, dann bedeutet das nur, daß es nicht erlaubt ist, die Anfragen direkt an den Server für die Domain `.foo.net` zuzustellen, sagt aber noch nichts darüber aus, was sonst damit passieren soll. Die `always_direct` Direktiven werden vom Squid **immer** als erstes verwendet. Wird also hier an Match mit der eingegebenen URL gefunden, so werden die `never_direct` Einträge nicht mehr angefaßt. D.h. die Reihenfolge der Anordnung der `always_direct` und der `never_direct` Zeilen im Konfigurationsfile spielen keine Rolle.

- Tag `never_direct`: Damit wird festgelegt, welche Domains NIEMALS direkt zugestellt werden sollen. Wenn nun das obige Beispiel des externen Servers aufgegriffen wird, so muß die Definition so aussehen:

```
acl external dstdomain .foo.net
never_direct allow external
```

  Mit diesen beiden Optionen wird festgelegt, wie mit einzelnen Domains zu verfahren ist. Wohin die Weiterleitung stattfinden soll, ist ja bereits bei `cache_peer_access` und `cache_peer` festgelegt worden.

- Tag `no_cache`

Alle weiteren Konfigurationsmöglichkeiten werden im Konfigurationsfile ausführlich beschrieben. Eine gute FAQ findet man unter `http://www.squid-cache.org/` [squ 02a].

Kommunikation mit anderen Proxies, Firewalls oder Caches

Squid unterstützt das Hirarchiekonzept von Proxychaining. Hat ein Squid die angefragten Infos nicht auf seiner Platte, so fragt er standardmäßig beim Webserver die Daten an. Befindet sich der Proxy in einer Proxykette, so kann der Proxy mit einem Nachbarproxy kommunizieren, um so Daten zwischen den Caches auszutauschen. Können mehrere Caches für eine Anfrage gefragt werden, so schickt der Proxy alle IPC-Request gleichzeitig los und setzt den Clientrequest so lange auf die Warteliste, bis einer der Caches eine positive Antwort geliefert hat.

- Tag `cache_peer` definiert andere Caches in einer Hierarchie, mit denen er kommunizieren kann. Dabei sind Informationen über die IP adresse, den HTTP-Port und natürlich auch den ICP-Port nötig. Jedem Kommunikationspartner können verschiedene Optionen zugeordnet werden, so daß sich das Kommunikationsverhalten festlegen läßt. Folgende Syntax wird dabei verwendet:

  ```
  cache_peer hostname type http_port icp_port options
  ```

  `type`:
    - `parent`: 'Eltern', d.h. die Caches mit dieser Option stehen in der Caching Hierarchie weiter oben
    - `sibling`: 'Geschwister', d.h. diese Caches stehen in der Caching Hierarchie auf gleicher Ebene. Diese Option ist oft nicht empfehlenswert, da oft die Weiterleitung mit dieser Option nicht korrekt funktioniert.
    - `multicast`

  `http_port`: Portangabe, auf dem der entfernete Proxy auf Clientanfragen hört.

  `icp_port`: Über diesen Port wird der benachbarte Cache nach Objekten befragt.

  `options`:
    - `proxy-only`: Alle Objekte, die über diesen Proxy geholt werden, sollen nicht lokal gespeichert werden. Normalerweise nimmt Squid die Cacheinformationen von den Cache, der als erster antwortet.
    - `weight=n`: Das entspricht einer Gewichtung eines Caches. Standardmäßig ist der Wert 1, höchere Werte haben eine höhere Gewichtung. Dieses Verhalten kann durch die Gewichtung der Caches beeinflußt werden. Maschinen mit höherer Gewichtung werden bevorzugt.
    - `ttl=n`: dient zur Spezifikation einer IP-Multicast-TTL, die verwendet wird, wenn ICP Pakete zu diesem Cache gesendet werden. Dies ist nur sinnvoll, wenn ICP Pakete an eine Multicast Gruppe geschickt werden. Dann müssen aber auch die Gruppenmitglieder mit `multicast-responder` definiert werden.
    - `no-query`: sorgt dafür, daß keine ICP Anfragen an den Cache gestellt werden, d.h. der bei diesem Proxy angegebene ICP-Port wird ignoriert. Das kann nötig sein, wenn ein Proxy nicht auf ICP-anfragen reagiert. Ohne diese Option würde so ein Cache als nicht funktionsfähig eingestuft und auch nicht für HTTP-Anfragen verwendet und der lokale Cache w2urde versuchen, das Ziel der HTTP-Anfrage direkt zu erreichen.
    - `default`: kennzeichnet einen Cache, der als 'Letzte Instanz' zu deuten ist. D.h. an diesen Cache werden alle Anfragen geschickt, die nicht anders zuordenbar sind. Oft wird diese Option zusammen mit `no-query` verwendet, um auch ganz sicher zu gehen, dass alle nicht zuordenbaren Requests darüber abgehandelt werden.

  - `round-robin`: Die so gekennzeichneten Caches werden im Round Robin Verfahren abgefragt.
  - `multicast-responder`: definiert Mitglieder einer Multicast Gruppe. ICP Anfragen werden nicht direkt zu diesem Peer gesendet, aber Antworten davon werden akzeptiert.
  - `no-digest`: sorgt dafür, das von diesem Nachbarn keine Cacheauszüge abgefragt werden.
  - `no-netdb-exchange`: läßt keinen NetDB Datenbankaustausch mit diesem Cache zu.
  - `login=user:password`: Ist der zu konfigurierende Proxy nur für eine Arbeitsgruppe zuständig, die zur Authentisierung nur eine User-ID und ein Passwort bei dem Cache benutzt, an den die Anfragen weitergeleitet werden, so kan der Proxy die Anmeldung übernehmen. Debei ist zu beachten, daß im HTTP-Protokoll festgelegt ist, daß für eine HTTP-Verbindung nur **eine** Proxyauthentisierung gestattet ist. Es kann bei der Verbindung noch zu einer Serverauthentisierung kommen, mehrere Proxyauthentisierungen sind **nicht** möglich!
  - `connect-timeout=nn`: legt einen spezifischen Timeout an.
  - `digest-url=url`: gibt die URL an, unter der der Cacheauszug bei diesem Proxy zu finden ist.

- Tag `cache_peer_access` definiert zusammen mit den definierten ACLs die Domains, die an einen Cache weitergeleitet werden sollen. Es ist auch sinnvoll anzugeben, welche Domains nicht hierhin geschickt werden dürfen.

Beispielkonfiguration Squid

Folgende Anforderungen soll der zu konfigurierende Squid erfüllen:

- Folgende Proxies sind mit den angegebenen Optionen festzulegen:
  - `10.11.12.13`:
    - HTTP Port 80
    - ICP Port 7
    - steht in der Hierarchie auf gleicher Ebene
    - alle über diesen Proxy zu holenden Objekte sollen nicht lokal gespeichert werden
    - es sollen keine ICP Anfragen an den Proxy gestellt werden
  - `10.20.30.40`:
    - HTTP Port 8080
    - ICP Port 3130

    * steht in der Hierarchie auf einer höheren Ebene
    * es soll kein Datenbankaustausch erfolgen
    * er soll als letzte Instanz fungieren

- Als Zieldomains sind nur `.de` und `.com` erlaubt.
- Das Netz `10.10.10.0/24` darf den Dienst uneingeschränkt nutzen.
- Das Netz `10.10.11.0/24` darf nur auf die Domain `.intern.de` zugreifen.
- Das Netz `10.10.12.0/24` darf auf die Domain `.intern.de` und `.extern.de` ohne Authentisierung zugreifen. Bei allen anderen Domains soll Userauthentisierung erfolgen.
- Die interne Domain `.intern.de` soll direkt aufgelöst werden.
- Die Domain `.extern.de` soll an den Proxy `10.11.12.13` weitergeleitet werden.
- Alle anderen Domain werden über die `10.20.30.40` geschickt.

Die Umsetzung dieser Anforderungen könnte in der `/etc/squid.conf` folgendermaßen aussehen:

```
#
# Praktikum
#
# Definition der anzusprechenden Caches mit Ports und Optionen
#
cache_peer 10.11.12.13 sibling 80 7 proxy-only no-query
cache_peer 10.20.30.40 parent 8080 3130 no-netdb-exchange default
#
# Festlegung des Authentisierungsprogramms und des dazugehoeringen Files
#
authenticate_program /usr/bin/ncsa_auth /usr/local/etc/htpasswd-file
#
# Definition der ACLs
#
acl all src 0.0.0.0/0.0.0.0
acl direkt dstdomain .intern.de
acl intranet dstdomain .extern.de
acl erlaubte-domains dstdomain .de .com
acl admin-netz src 10.10.10.0/255.255.255.0
acl auth-netz src 10.10.12.0.0/255.255.255.0
acl user-netz src 10.10.11.0/255.255.255.0
acl authentisierung proxy_auth REQUIRED
```

```
#
# Zugriffsdefinitionen
#
http_access allow admin-netz erlaubte-domains
http_access allow user-netz direkt
http_access allow auth-netz direkt
http_access allow auth-netz intranet
http_access allow auth-netz erlaubte-domains authentisierung
http_access deny all
#
# Zuordung der Weiterleitungen
#
cache_peer_access 10.11.12.13        allow   intranet !direkt
cache_peer_access 10.11.12.13        deny    all
#
# Definition von Weiterleitung bzw. lokaler Aufloesung
#
always_direct allow direkt
always_direct deny all
never_direct allow all
#
```

Für die Userauthentisierung wurde das Programm NCSA, das beim Sourcecode von Squid mitgeliefert wird, kompiliert und installiert. Das Authentisierungsfile kann mit Hilfe von `htpasswd` wie beim Apache in Abschnitt 7.2.4 erzeugt und mit Usern gefüllt werden.
Die Tests erfolgen mittels Browser von den einzelnen IP-Adressen aus, zusammen mit den Einträgen in der `/var/log/squid/access.log`.
Die Einträge in der `/var/log/squid/access.log` sehen bei direkter Auflösung so aus:

```
1023701105.450 46 10.10.10.10 TCP_MISS/200 8509 GET http://intra.intern.de/ -
                                                    DIRECT/10.0.0.1 text/html
```

wobei die erste Spalte die Zeit, die zweite Spalte die vom Cache benötigte Zeitspanne und die dritte Spalte die Source-IP-Adresse beinhalten. In der vierten steht der Ergebniscode des Caches zusammen mit dem HTTP Code. Die fünfte Spalte gibt die übertragenen Bytes an, die sechste die angefragte HTTP Methode, dannach kommt die verlangte URL. In der achten Spalte würden die durch den ident-Lookup vom Client erfragten Informationen stehen. Dies ist standardmäßig deaktiviert. Danach beschreibt man, wie die URL behandelt wurde, d.h. ob sie lokal aufgelöst wurde oder an einen anderen Cache weitergereicht worden ist. Im letzten Feld wird der Type des Objekts aus dem HTTP Header angezeigt. Für eine genauere Erklärung der einzelnen Codes und Einträge schlagen Sie bitte in den FAQ's auf `http://www.squid-cache.org/` [squ 02a] nach.
Wird eine URL weitergeleitet, so sieht der Logeintrag z.B. so aus:

```
1023709965.121 37 10.10.10.10 TCP_MISS/200 856 GET http://intra.extern.de -
                                      FIRST_UP_PARENT/10.11.12.13 text/html
```

Und wird Userauthentisierung verlangt, so ist im Logfile die User-ID sichtbar:

```
1023711015.728  12481 10.10.12.10 TCP_MISS/200 21765 GET http://www.gmx.de/
                      testuser TIMEOUT_DEFAULT_PARENT/10.20.30.40 text/html
```

Wurde versucht, etwas anzufragen, was nicht erlaubt war, so kann man das auch recht deutlich im Logfile erkennen:

```
1023711016.567      2 10.10.12.10 TCP_DENIED/403 1038 GET http://213.165.64.39
                                                           testuser NONE/- -
```

und als Fehlermeldung für den Benutzer erscheint im Browser:

```
The requested URL could not be retrieved


 While trying to retrieve the URL: http://www.squid-cache.org/

The following error was encountered:

Access Denied.

Access control configuration prevents your request from being allowed at this time.
Please contact your service provider if you feel this is incorrect.

Your cache administrator is root@sectest.muc.meinedomain.de.

Generated Mon, 10 Jun 2002 13:49:06 GMT by sectest.muc.meinedomain.de (Squid/2.4.STABLE3
```

wobei die Definition der hier zu erscheinenden Mailadresse in der `/etc/squid.conf` bei dem Tag `cache_mgr` erfolgt.

8.3.2 Firewall Tool Kit

Das FirewallToolKit ist ein typisches Beispiel für einen Application Level Gateway. Die ursprüngliche Entwicklung wurde durch Trusted Information Systems, Incorporated (TIS) vorgenommen. Daraus entwickete sich das kommerzielle Firewallprodukt Gauntlet von TIS. Mittlerweile wurde TIS von Network Associates Inc. (NAI) aufgekauft und das Produkt

Gauntlet an Secure Computing abgegeben.
FWTK ist keine freie Software, sondern man darf es nur dann privat nutzen, wenn man es selber installiert, und ist zur kommerziellen Nutzung nicht erlaubt. Ausführliche Tutorial und Installationsanleitungen findet man unter `http://www.fwtk.org/`. Dort kann man auch einen Antrag auf Download der Software stellen. Diese liegt im Sourcecode vor und muß von Hand kompiliert werden.
Das FWTK ist ein Verteter der Philosophie: 'Alles, was nicht explizit erlaubt ist, ist verboten'.
Hier nun eine kleine Installationsanleitung des FWTK für SuSE 8.0:
Das Paket `gdbm-devel` muß auf der Maschine installiert sein. Nach dem Entpacken sollten Files, die evtl. noch geändert werden, zuerst einmal als Sicherung abgespeichert werden:

```
cp -p Makefile Makefile.ori
mv Makefile.config Makefile.config.ori
cp -p Makefile.config.linux Makefile.config
cp -p firewall.h firewall.h.ori
```

Somit kann ohne großen Aufwand der Originalzustand wieder hergestellt werden. Das mitgelieferte README beinhaltet alle nötigen Informationen, um die Software erfolgreich installieren zu können. Da wir als `Makefile.config` das `Makefile.config.linux` verwenden, haben wir `Makefile.config.linux` zu `Makefile.config` umbenannt. Im `Makefile.config` wird nun folgendes geändert:

```
#CC=    cc
CC=     gcc
AUXLIB= -lcrypt
```

Beim anschließenden `make` sollten nun die zur Verfügung stehenden Proxies kompiliert werden. Eine eventuell auftretende Fehlermedung bei `x-gw` kann ignoriert werden. Mit `make install` werden die Binaries nach `/usr/local/etc/` installiert. Folgende Dateien sollen nun zur Verfügung stehen:

- `authsrv`: ist der Authentisierungsserver des Firewalls. Hiermit können User administriert werden.

  ```
  /usr/local/etc/authsrv
  authsrv# help
  Command List:
  (Commands may be invoked with unique leading abbreviation)
  authorize username [comment]
  authenticate username
  response <text>
  quit
  ```

```
exit
display username
adduser username [fullname]
deluser username
enable username [onetime]
disable username
password [username] passwordtext
passwd [username] passwordtext
proto username protoname
group username groupname
rename username newname [fullname]
wiz username
unwiz username
superwiz username
operation group/user username command dest [tokens]
list [group]
ls [group]
?
help
```

- Userinformationen anzeigen lassen: `display username`
- User anlegen: `adduser username [fullname]`
- User löschen: `deluser username`
- User anzeigen lassen: `display username`
- Festlegen des Authentisierungsverfahrens pro User: `proto username protoname`
- Passwort für einen User festlegen: `passwd [username] passwordtext`
- User aktivieren: `enable username [onetime]`, wobei sich der User bei der Option `onetime` nur einmal anmelden darf und dann wieder gesperrt ist.
- User sperren: `disable username`
- User einer Gruppe zuordnen: `group username groupname`
- etc.

Zur Administration ist natürlich der Super User (`root`) berechtigt. Das Anlegen eines Users über Kommandozeile sieht folgendermaßen aus:

```
authsrv# adduser oberhait Oberhaitzinger:Barbara
ok - user added initially disabled
authsrv# passwd oberhait q1w2e3r4
Password for oberhait changed.
```

```
authsrv# group oberhait t-systems
set group
authsrv# enable oberhait
enabled
authsrv# wiz oberhait
set group-wizard
authsrv# display oberhait
Report for user oberhait
Group t-systems (Oberhaitzinger:Barbara)
Authentication protocol: password
Flags: GROUP-WIZARD
```

Die Administration von Usern kann aber auch über den vom Authentisierungsserver geöffneten Port über TCP erfolgen. Dazu brauchen die berechtigten Administratoren keinen Account auf dem Betriebssystem. Es genügt, dem Useraccount in der Authentisierungsdatenbank des Firewalls die Rechte zur Gruppenverwaltung (`wiz`) bzw. die Rechte zur Verwaltung aller User in der Firewallauthentisierungsdatenbank (`superwiz`) zu geben. Der Authentisierungsserver läuft normalerweise auf Port 7777, was in der `/etc/inetd.conf` noch einzutragen ist:

```
authsrv  stream  tcp nowait root  /usr/local/etc/authsrv  authsrv
```

mit dem Eintrag in der `/etc/services`

```
authsrv         7777/tcp                        # authserver fwtk
```

Somit kann auch über den Port 7777 die Useradministration erfolgen:

```
telnet localhost 7777
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Authsrv ready. (V2.1)
auth oberhait
password
response q1w2e3r4
ok
display oberhait
Report for user oberhait
Group t-systems (Oberhaitzinger:Barbara)
Last authenticated: Wed Jun  5 16:28:42 2002
Authentication protocol: password
Flags: GROUP-WIZARD
```

```
.
quit
Connection closed by foreign host.
```

Die Firewalluserdatenbank heißt `/usr/local/etc/fw-authdb`.

- `authdump`: Um die Einträge der Firewalluserdatenbank im ASCII Format zu erhalten, kann ein Dump in eine Datei umgeleitet werden: `authdump > userdaten-acsii`
- `authload`: Hat man eine ASCII Datei von FWTK Userdaten vorliegen und möchte man diese in die Firewalluserdatenbank überführen, so wird `authload < userdaten-acsii` verwendet.
- `authmgr`: Diese Programm dient ebenfalls zur Useradministration:

```
./authmgr
Connected to server
authmgr-> login
Username: oberhait
Password:
Logged in
authmgr-> display oberhait
Group t-systems (Oberhaitzinger:Barbara)
Last authenticated: Fri Jun  7 11:08:35 2002
Authentication protocol: password
Flags: GROUP-WIZARD
authmgr-> exit
```

- `smap`: Mailserver, der Mails entgegennimmt und in einem abgeschlossenen Bereich ablegt.
- `smapd`: Nimmt die Mails aus dem abgeschlossenen Bereich und stellt sie in das Mailqueue Verzeichnis des Mailers.
- `plug-gw`: Ist als Circuit Level Gateway frei konfigurierbar.
- `ftp-gw`: FTP Proxy
- `tn-gw`: Telnet Proxy
- `rlogin-gw`: rlogin Proxy
- `http-gw`: HTTP Proxy
- `netperm-table`: Konfigurationsdatei der Dienste.

Das Starten der gewünschten Dienste kann auf verschiedene Arten geschehen: entweder werden eigene Startscripte geschrieben und in die Runlevelverzeichnisse verlinkt oder man macht die entsprechenden Einträge in der `/etc/inetd.conf`

```
http-gw stream  tcp nowait root  /usr/local/etc/http-gw  http-gw
ftp     stream  tcp nowait root  /usr/local/etc/ftp-gw   ftp-gw
telnet  stream  tcp nowait root  /usr/local/etc/tn-gw    tn-gw
```

und `/etc/services`

```
http-gw         8080/tcp                        # http proxy fwtk
```

Nun sind die Dienste zwar gestartet, aber noch nicht konfiguriert. Dies erfolgt in der `/usr/local/etc/netperm-table`.
Folgende Konfiguration soll nun umgesetzt werden:

- Alle Rechner aus dem Netz `10.10.10.0/24` dürfen die Dienste HTTP, Telnet und FTP benutzen.
- Telnet soll Userauthentisierung verlangen.
- Telnet soll nur zur `10.10.2.2` möglich sein.
- FTP soll nur für das Netz `10.10.10.128/25` Userauthentisierung verlangen, für `10.10.10.0/25` soll keine Authentisierung erforderlich sein.
- Bei FTP sollen STOR und RETRIEVE Kommandos mitprotokolliert werden.
- Der Authentisierungsserver soll auf `127.0.0.1` Port 7777 laufen.
- Der Authentisierungsserver soll nur von der `127.0.0.1` angesprochen werden dürfen.
- Alle Gateways sollen den Authentisierungsserver auf `127.0.0.1` Port 7777 ansprechen.

```
# Example ftp gateway rules:
# --------------------------
ftp-gw: timeout 3600
ftp-gw: permit-hosts 10.10.10.0:255.255.255.128
ftp-gw: permit-hosts 10.10.10.128:255.255.255.128 -auth -log { retr stor }

# Example telnet gateway rules:
# -----------------------------
tn-gw: timeout 3600
tn-gw: permit-hosts 10.10.10.0:255.255.255.0 -auth -dest 10.10.2.2
```

```
# Example http gateway rules:
# -----------------------------
http-gw: timeout 3600
http-gw: permit-hosts 10.10.10.0:255.255.255.0

# Example auth server and client rules
# ------------------------------------
authsrv: hosts 127.0.0.1
authsrv: database /usr/local/etc/fw-authdb
authsrv: badsleep 1200
authsrv: nobogus true

# clients using the auth server
*: authserver 127.0.0.1 7777
```

Wichtige Hinweise und Anleitungen zur Konfiguration erhalten Sie auch durch die mitgelieferten PostScript Dateien unter `/usr/local/src-local/fwtk/fwtk/doc/`.
Die Zugriffe sehen nun fogendermaßen aus:
FTP von einem Rechner aus dem Netz `10.10.10.0/25` aus zu `majestix.muc.meinedomain.de` laut Vorgabe ohne Userauthentisierung:

```
ftp sectest.muc.meinedomain.de
Connected to 10.10.1.1.
220 sectest FTP proxy (Version V2.1) ready.
Name (10.10.1.1:oberhait): testuser@majestix.muc.meinedomain.de
331-(----GATEWAY CONNECTED TO 10.50.181.8----)
331-(220 majestix.muc.meinedomain.de FTP server (Version 6.4/OpenBSD/Linux-ftpd-0.17) re
331 Password required for testuser.
Password:
230- Linux majestix 2.2.18pre21 #1 Sat Nov 18 18:47:15 EST 2000 i686 unknown
230-
230- Most of the programs included with the Debian GNU/Linux system are
230- freely redistributable; the exact distribution terms for each program
230- are described in the individual files in /usr/share/doc/*/copyright
230-
230- Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
230- permitted by applicable law.
230 User testuser logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> quit
221 Goodbye.
```

Zuerst wird die Verbindung zum Firewall aufgebaut. Da von diesem Netzbereich aus keine Userauthentisierung verlangt wird, kann sofort `userid-entferntes-system@entferntes-system` angegeben werden. Die Passwortabfrage kommt bereits von `majestix.muc.meinedomain.de`. Kommt man aus dem Netzwerkbereich `10.10.10.128/25`, für den Authentisierung vorgeschrieben ist, so sieht der Dialog so aus:

```
ftp sectest.muc.meinedomain.de
Connected to 10.10.1.1.
220 sectest FTP proxy (Version V2.1) ready.
Name (10.10.1.1:oberhait): oberhait
331 Enter authentication password for oberhait
Password:
230 User authenticated to proxy
ftp> user testuser@majestix.muc.meinedomain.de
331-(----GATEWAY CONNECTED TO 10.50.181.8----)
331-(220 majestix.muc.meinedomain.de FTP server (Version 6.4/OpenBSD/Linux-ftpd-0.17) re
331 Password required for testuser.
Password:
230- Linux majestix 2.2.18pre21 #1 Sat Nov 18 18:47:15 EST 2000 i686 unknown
230-
230- Most of the programs included with the Debian GNU/Linux system are
230- freely redistributable; the exact distribution terms for each program
230- are described in the individual files in /usr/share/doc/*/copyright
230-
230- Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
230- permitted by applicable law.
230 User testuser logged in.
ftp> quit
```

Hier erfolgt erst die Userauthentisierung. Danach ist man mit dem FTP-Proxy auf dem Firewall verbunden. Mit `user testuser@majestix.muc.meinedomain.de` wird nun die Verbindung zu `majestix.muc.meinedomain.de` als User `testuser` aufgebaut. Diese Verbindungen sind auch wunderbar im Logfile `/var/log/messages` mitzuverfolgen.
Wird von einer IP-Adresse aus versucht, den FTP-Proxy anzusprechen, die nicht erlaubt ist, so kommt folgende Meldung:

```
ftp sectest.muc.meinedomain.de
Connected to 10.10.1.1.
500 test.muc.meinedomain.de/10.1.1.1 not authorized to use FTP proxy
ftp> quit
```

Der Eintrag im Logfile:

```
Jun  6 08:46:06 sectest ftp-gw[16848]: deny
                     host=test.muc.meinedomain.de/10.1.1.1 use of gateway
```

zeigt an, daß diese Verbindung verboten ist.
Telnet ist folgendermaßen zu handhaben:

```
telnet sectest.muc.meinedomain.de
Trying 10.10.1.1...
Connected to 10.10.1.1.
Escape character is '^]'.
Username: oberhait
Password: ########
Login Accepted
sectest telnet proxy (Version V2.1) ready:
tn-gw-> c 10.10.2.2
Trying 10.10.2.2 port 23...
Connected to testrechner.muc.meinedomain.de.

SunOS 5.8

login: oberhait
Password:
Last login: Mon Mar 25 11:20:15 from sectest.muc.meinedomain.de
Sun Microsystems Inc.   SunOS 5.8       Generic February 2000
testrechner % exit
testrechner % logout
Remote server has closed connection
Connection closed by foreign host.
```

Wird nun versucht, auf eine nicht erlaubte Ziel-IP-Adresse zuzugreifen, so kommt dieser Dialog zustande:

```
tn-gw-> c 10.10.2.3
Not permitted to connect to 10.10.2.3
tn-gw-> quit
Disconnecting...Connection closed by foreign host.
```

Im Logfile sieht das dann so aus:

```
Jun  5 17:12:55 sectest tn-gw[15400]: deny
                  host=rechner.muc.meinedomain.de/10.10.10.2 destination=10.10.2.3
```

Tests für HTTP können durch Eintrag des HTTP-Proxies in den Browser und Aufruf beliebiger Seiten gemacht werden. Natürlich kann das auch über die Kommandozeile simuliert werden:

```
telnet sectest.muc.meinedomain.de 8080
Trying 10.10.1.1...
Connected to 10.10.1.1.
Escape character is '^]'.
GET http://www.muc.meinedomain.de HTTP/1.0

HTTP/1.1 200 OK
Date: Thu, 06 Jun 2002 07:42:19 GMT
Server: Apache/1.3.9 (Unix) Debian/GNU
Last-Modified: Mon, 08 Oct 2001 12:18:52 GMT
ETag: "6fb6-2007-3bc199ac"
Accept-Ranges: bytes
Content-Type: text/html
X-Pad: avoid browser bug

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html> <head>
<title>meinedomain SH CCS</title>
</head>

<script language="JavaScript">

<!--
function Sicherheitshinweis()
.
.
.
.
.
<!-- hhmts end -->
</body> </html>

Connection closed by foreign host.
```

Versucht nun eine IP-Adresse den HTTP-Proxy zu benutzen, die nicht zugelassen ist, so kommt diese Meldung:

```
telnet sectest.muc.meinedomain.de 8080
Trying 10.10.1.1...
Connected to 10.10.1.1.
Escape character is '^]'.
GET http://www.muc.meinedomain.de HTTP/1.0
HTTP/1.0 403 Error
```

```
Content-type: text/html

<h1>Error- 403</h1>
<HR>
<PRE>
Sorry, Access denied for rechner.muc.meinedomain.de (10.10.10.2)<br>
</PRE>
<br><hr>
<A HREF="http://-internal-/-http-gw-internal-/version.html">http-gw version </A>
 (sectest.muc.meinedomain.de)
Connection closed by foreign host.
```

8.4 Praktische Aufgaben

Der derzeitig immer noch gültige Versuchsaufbau ist in Abbildung 71 dargestellt.

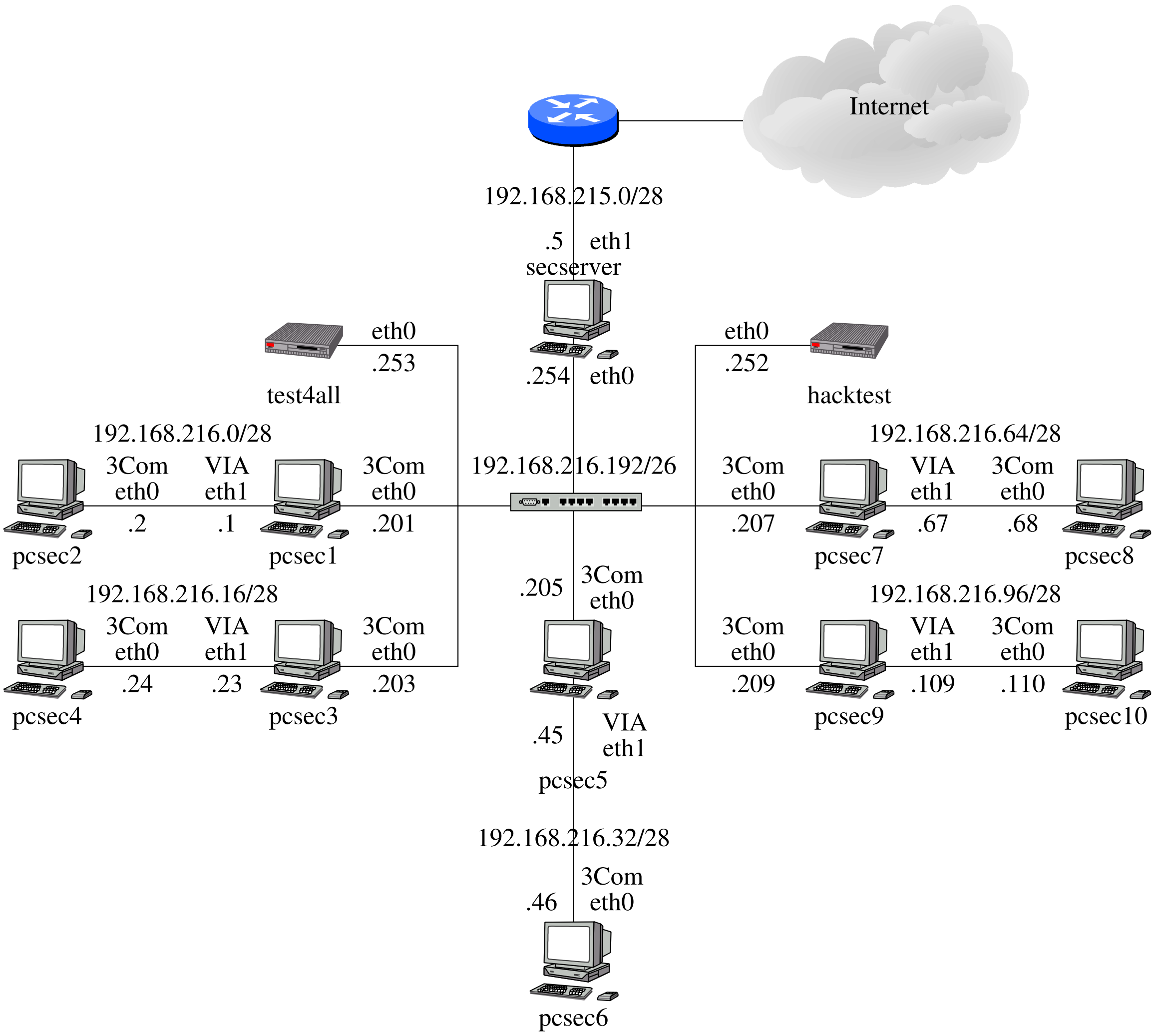

Abbildung 71: Der Versuchsaufbau für die weiteren Versuche des Praktikums

Für alle folgenden Aufgaben ist eine schriftliche Ausarbeitung zu erstellen, die folgendes beinhaltet:

- Angabe aller gemachten Konfigurationsänderungen mit Begründung.
- Angabe aller gemachten Tests mit Aufführung der bekommenen Meldungen inklusive Interpretation.

- Auflistung der erzeugten Logeinträge mit den Erklärungen, welche Schlüsse daraus abzuleiten sind.

Diese Ausarbeitung ist zum nächsten Termin per Mail an die E-Mail Adresse `secp@nm.informatik.uni-muenchen.de` zu schicken. Als SMTP-Gateway kann der `secserver` verwendet werden. Ein Tip ist das Führen eines Logfiles, so daß alle gemachten Änderungen mitprotokolliert werden.

8.4.1 Squid

1. Deaktivieren Sie alle Paketfilterregeln und sorgen Sie dafür, daß alle Dienste auf allen Interfaces hören und von allen Adressen aus dem Netz `192.168.216.0/24` erreichbar und nutzbar sind.
2. Installieren Sie Squid von der SuSE 8.0 Distribution.
3. Tragen Sie in der `/etc/resolv.conf` die `192.168.216.254` als einzigen Nameserver ein.
4. Nun folgt die Squidkonfiguration. **Überprüfen Sie alle logischen Teilschritte mittels in der Vorbereitung beschriebener Tests und der Logfileeinträge. Gehen Sie immer schrittweise vor!**
   - Setzen Sie den Squidport auf 8888. ICP Port ist standardmäßig 3130.
   - Setzen Sie die bei einer Fehlermeldung angezeigte Mailadresse auf den `root`-Account Ihres Rechners.
   - Es sind nur Domains aus `.de` und `.org` erlaubt.
   - Bei allen Definitionen der übergeordneten Proxies soll dafür gesorgt werden, daß
     – kein von diesem Cache geholtes Object gespeichert wird.
     – keine ICP-Anfrage erfolgt.

     Alle übergeordneten Proxies sind als `parent` zu betrachten.
   - Implementieren Sie ein Proxychaining, daß hier beispielhaft für `pcsec4` und `pcsec3` dargestellt ist. Bitte transferieren Sie die hier gemachten Angaben auf Ihre spezielle Situation:
     – Proxychain für `pcsec4`:
       * `pcsec4.secp.nm.informatik.uni-muenchen.de` und `pcsec3.secp.nm.informatik.uni-muenchen.de` sind lokal aufzulösen. Hierzu ist keine Userauthentisierung nötig.
       * Schicken Sie den Rest an den in der Hirarchie über Ihnen stehenden Cache `pcsec3 192.168.216.23`.

    * Erlauben Sie nur Ihren Rechner und den `pcsec3`, Ihren Cache zu verwenden.
    * Laden Sie sich aus dem Internet den aktuellen Squid-Quellcode herunter und entpacken Sie ihn. Installieren Sie das Authentisierungsprogramm `ncsa_auth`, erzeugen Sie ein Authentisierungsfile mit `htpasswd` und aktivieren Sie die Authentisierung in der `/etc/squid.conf`.
    * Erlauben Sie alle URLs in der Domain `secp.nm.informatik.uni-muenchen.de` ohne Authentisierung.
    * Verlangen Sie für alle restlichen Verbindungen in den erlaubten Domains `.de` und `.org` Authentisierung.
  - Proxychain für `pcsec3`:
    * `pcsec4.secp.nm.informatik.uni-muenchen.de` und `pcsec3.secp.nm.informatik.uni-muenchen.de` bzw. `pcsec3-switch.secp.nm.informatik.uni-muenchen.de` sind lokal aufzulösen. Hierzu ist keine Userauthentisierung nötig.
    * Schicken Sie den Rest an den in der Hirarchie über Ihnen stehenden Cache `secserver 192.168.216.254` HTTP Port 3128, ICP Port 3130. Dabei müssen Sie bedenken, dass zumindest der `secserver` für den Zugriff auf die von Ihnen direkt aufzulösenden Domains erlaubt sein muß.
    * Sorgen Sie dafür, daß `pcsec1`, `pcsec5`, `pcsec7` und `pcsec9` nur die Webseiten `pcsec4.secp.nm.informatik.uni-muenchen.de` und `pcsec3.secp.nm.informatik.uni-muenchen.de` bzw. `pcsec3-switch.secp.nm.informatik.uni-muenchen.de` über Ihren Cache erreichen können.
    * Sorgen Sie dafür, daß `pcsec4` und Ihr Rechner alle erlaubten Domains ohne Userauthentisierung erreichen können.

8.4.2 FWTK

1. Deaktivieren Sie FTP und Telnet in der `/etc/inetd.conf`.
2. Kompilieren und Installieren Sie das FWTK nach Anleitung.
3. Aktivieren Sie den Authentisierungsserver `authsrv` auf Port 7777 über die `/etc/inetd.conf` und die `/etc/services`.
4. Legen Sie mit dem Programm `authsrv` Testuser (`testsec1`, `testsec2`, Passwort analog zur User-ID) an.
5. Aktivieren Sie den `http-gw` auf Port 8282.

6. Erlauben Sie alle Interfaces Ihres Rechners, den `http-gw` zu benutzen. Userauthentisierung ist nicht nötig.

7. Aktivieren Sie `ftp-gw` und `tn-gw`.

8. Erlauben Sie das Netz `192.168.216.0/24`, den `ftp-gw` und `tn-gw` mit Userauthentisierung zu benutzen, wobei Telnet nur zu `test4all` erlaubt sein soll.

9 Circuit Level Gateways und Firewallarchitekturen

9.1 Circuit Level Gateways

Circuit Level Gateways arbeiten nahezu transparent für den Benutzer, sofern kein explizites Login auf dem Gateway verlangt wird. Ein Circuit Level Proxy findet dann Verwendung, wenn man nicht unbedingt einen Paketfilter einsetzen möchte, ein Proxy für diesen Dienst aber nicht vorhanden ist. Cicuit Level Proxies gibt es für UDP-basierte und TCP-basierte Protokolle. Sie werden dann häufig auch als **UDP-Relay** oder **TCP-Relay**, manchmal auch als **Circuit-Gateway** bezeichnet. Als anwendungsunabhängige Multiprotokoll-Proxies auf Transportebene haben sie die Aufgabe, die Verbindungsdaten als Mittler zwischen dem internen und dem externen Netz hin und her zu kopieren. Auf Basis der Sender-, Empfängeradressen und Portnummern entscheidet der Gateway, ob die Verbindung zum Zielsystem aufgebaut werden darf oder nicht. Alle Aktionen werden dabei zentral protokolliert. Circuit Level Gateways unterscheiden sich von Paketfiltern eigentlich nur in der Tatsache, daß sie die Verbindung unterbrechen und somit immer intern verwendete IP-Adressen verbergen.
Anders als Application Level Gateways können Cicuit Level Gateways nur eine Verbindung zu einem Rechner im Internet aufbauen, der bei der Konfiguration vom Administrator angegeben wurde. Alle Rechner im zu schützenden Netz können dann nur mit dem einen Rechner im Internet kommunizieren (**n-zu-1-Verbindung**).
Da Circuit Level Gateways historisch vor den Application Level Gateways entwickelt worden sind, werden sie auch als zweite Firewall Generation bezeichnet, während die Application Level Gateways die dritte Generation darstellen.
Müssen Verbindungen zu verschiedenen Rechnern im Internet mit Hilfe eines Circuit Level Gateways aufgebaut werden, gibt es für diese sogenannten **n-zu-m-Verbindungen** mehrere Möglichkeiten. Zum einen kann der Rechner, auf dem die Circuit Level Gateways laufen, so konfiguriert sein, daß er mehrere IP-Adressen hat, unter denen er für Rechner aus dem zu schützenden Netz erreichbar ist. Die Circuits werden dann so konfiguriert, daß sie in Abhängigkeit von der IP-Adresse, unter der das Paket den Proxyserver erreicht hat, eine andere Ziel-IP-Adresse verwenden. Bei der anderen Alternative wird derselbe Effekt erreicht, indem unterschiedliche Portnummern auf dem Proxy definiert sind und der Circuit Level Proxy in Abhängigkeit von der Portnummer die Ziel-IP-Adresse auswählt. Eine weitere Möglichkeit ist, den Circuit Level Gateway als Defaultrouter zum Internet hin zu verwenden. Dabei ist zwar das IP-Forwarding weiterhin deaktiviert, aber alle Pakete für Zieladressen im Internet landen beim Gateway und werden über die Circuit Level Proxies zugestellt.
Der Proxy wird in der Regel auf dem Client programmübergreifend definiert und muß nicht explizit in jeder Anwendung angegeben werden. Damit eine Internetanwendung trotz Eingabe der externen Zieladresse zuerst den Gateway kontaktiert, sind oft programminterne Anpassungen der Software nötig. Alle Socketcalls in der Altanwendung müssen durch neue Aufrufe ersetzt und anschließend der Code neu compiliert werden. Dies setzt voraus, daß