Squid 中文权威指南

（第 12 章）

译者序：

本人在工作中维护着数台 Squid 服务器，多次参阅 Duane Wessels（他也是 Squid 的创始人）的这本书，原书名是"Squid: The Definitive Guide"，由 O'Reilly 出版。我在业余时间把它翻译成中文，希望对中文 Squid 用户有所帮助。对普通的单位上网用户，Squid 可充当代理服务器；而对 Sina,NetEase 这样的大型站点，Squid 又充当 WEB 加速器。这两个角色它都扮演得异常优秀。窗外繁星点点，开源的世界亦如这星空般美丽，而 Squid 是其中耀眼的一颗星。

对本译版有任何问题，请跟我联系，我的Email是：yonghua_peng@yahoo.com.cn

彭勇华

目 录

第 12 章　验证辅助器

先前我在 6.1.2.12 章里谈起过代理验证。然而，我仅仅解释了如何编写用于代理验证的访问控制规则。这里，我将告诉你如何选择和配置部分验证辅助器。

回想一下，Squid 支持三种方式用于从用户端采集验证信用项：基本，摘要(Digest)，和 NTLM。这些方式指定 squid 如何从客户端接受用户名和密码。从安全观点看，基本验证非常脆弱。摘要和 NTML 验证显然更强壮。对每种方式，squid 提供一些验证模块，或辅助进程，用于实际处理认证的过程。

我提到的所有验证辅助器都包含在 squid 的源代码发布里。你可以在编译时使用./configure 选项来指定它们的目录名。例如：

```
% ls helpers/basic_auth

LDAP                    NCSA                    getpwnam
MSNT                    PAM                     multi-domain-NTLM
Makefile                SASL                    winbind
Makefile.am             SMB
Makefile.in             YP

% ./configure --enable-basic-auth-helpers=LDAP,NCSA ...
```

辅助器程序正常安装在$prefix/libexec 目录。

如同重定向器一样，squid 使用一个验证辅助器进程池。某个验证请求会被送往第一个空闲辅助器。当所有验证器进程都忙时，squid 将未处理请求放进队列。假如队列变得太大，squid 会以致命错误消息退出。大部分情况下，squid 缓存验证结果。这样就减少了辅助器进程的负载，并改进了响应时间。

12.1 配置 Squid

auth_param 指令控制了配置 squid 的验证辅助器的每个方面。不同的方式（基本，摘要，NTLM）有一些共性，也有一些唯一的参数。紧跟在 auth_param 后的第一个参数必须是 basic, digest, 或 ntlm 之一。我将在随后章节里，详细讲解每种验证机制的配置细节。

除了 auth_param 外，squid 还有 2 个指令影响到代理验证。可以使用 max_user_ip ACL 来阻止用户与其他人共享用户名和密码。假如 squid 检测到相同的用户名来自太多不同 IP 地址，该 ACL 被匹配，就可以拒绝这样的请求。例如：

```
acl FOO max_user_ip 2
acl BAR proxy_auth REQUIRED
```

http_access deny FOO
http_access allow BAR

在该情形中，假如用户从 3 个或更多的不同 IP 地址提交请求，squid 就拒绝该请求。authenticate_ip_ttl 指令控制 squid 记住每个用户的源 IP 地址多长时间。对于经常改变 IP 地址的用户，更小的 TTL 可能好点。在一个用户的 IP 地址很长时间不变的环境里，可以使用较大的 TTL。

12.2 HTTP 基本验证

基本验证最简单，然而最不安全。它本质上以明文来传送用户密码，尽管密码被编码成可打印字符。例如，假如用户敲入其用户名 Fannie 和密码 FuRpAnTsClUb，用户代理首先将这 2 者结合到一个单一串里，以冒号来分割用户名和密码：

Fannie:FuRpAnTsClUb

然后它用 base64 方法（定义在 RFC 2045）来编码这个串。它在 HTTP 头部里看起来如此：

Authorization: Basic RmFubmllOkZ1UnBBblRzQ2xVYgo=

若有人碰巧捕获到用户的 HTTP 请求，他能轻易获取到用户名和密码：

% echo RmFubmllOkZ1UnBBblRzQ2xVYgo= | /usr/local/lib/python1.5/base64.py -d

Fannie:FuRpAnTsClUb

遵循 HTTP/1.1 RFC 的要求，squid 不会转发验证信用项到其他服务器。换句话说，假如信用项是用于访问 squid 的，Authorization 头部会从外出请求里移除。

你会注意到，某些基本验证器可被配置来检查系统密码文件。因为基本信用项不被加密，所以在 cache 访问密码里包含登陆密码是个坏想法。假如选择使用 getpwnam 验证器，你应该完全理解让用户密码以明文在网络中传送的意义。

HTTP 基本验证支持下列 auth_param 参数：

auth_param basic program command
auth_param basic children number
auth_param basic realm string
auth_param basic credentialsttl time-specification

program 参数指定验证辅助程序的命令及其参数。大多数情况下，这里是到某个验证辅助程序的路径名。它们默认安装在/usr/local/squid/libexec 下。

children 参数告诉 squid 使用多少辅助器进程。默认值是 5，假如你不了解需要多少进程来处理请求，这个值就是个好起点。假如指定得太少，squid 会在 cache.log 里告警。

realm 参数是在提示输入用户名和密码时，用户代理显示给用户看的验证域字符串。可以使用一些简单的句子，例如“访问 squid 的缓存代理”。

credentialsttl 参数指定 squid 内在的缓存验证结果的时间数量。较大的值减少了外部验证器进程的负载，但加长了刷新期，直到 squid 检测到验证数据库的改变。注意，这仅影响到积极结果（例如成功的验证），消极的结果不会被 squid 缓存。默认的 TTL 值是 2 小时。

如下是个完整的示例：

```
auth_param basic program /usr/local/squid/libexec/pam_auth
auth_param basic children 10
auth_param basic realm My Awesome Squid Cache
auth_param basic credentialsttl 1 hour

acl KnownUsers proxy_auth REQUIRED
http_access allow KnownUsers
```

下面我将讨论 squid 自带的基本验证辅助器程序。

12.2.1 NCSA

```
./configure —enable-basic-auth-helpers=NCSA
```

NCSA 验证辅助器相对流行，这归咎于它的简单性和历史原因。它将用户名和密码存储在一个单独的文本文件里，类似于 Unix 的/etc/passwd 文件。这个密码文件格式最初是作为 NCSA HTTP 服务器项目的一部分发展而来的。

在 squid.conf 里，只须指定密码文件的路径作为程序的单一命令行参数。

```
auth_param basic program /usr/local/squid/libexec/ncsa_auth
    /usr/local/squid/etc/passwd
```

可以使用 Apache 自带的 htpasswd 程序来创建和更新密码文件。也可以在 http://www.squid-cache.org/htpasswd/ 这里下载。在该页面里，你也可以下载chpasswd CGI 脚本，它允许用户改变自己的密码（假如必要）。

12.2.2 LDAP

./configure —enable-basic-auth-helpers=LDAP

LDAP辅助器是到轻量级目录访问协议(LDAP)服务器的接口。在编译squid_ldap_auth辅助器之前，OpenLDAP库和头文件必须安装到系统中。可以在这里找到OpenLDAP：http://www.openldap.org/

squid_ldap_auth 程序至少需要 2 个参数：基本开放名(DN)和 LDAP 服务器主机名。例如：

```
auth_param basic program /usr/local/squid/libexec/squid_ldap_auth
    -b "ou=people,dc=example,dc=com"    ldap.example.com
```

LDAP 辅助器有 Unix 的 man 页，描述了其所有选项和参数。然而，在运行 make install 时，通常并未安装 squid 的这个 man 页。进入源代码树，手工运行 nroff，你可以读到这个 man 页。例如：

```
% cd helpers/basic_auth/LDAP
% nroff -man squid_ldap_auth.8 | less
```

12.2.3 MSNT

./configure —enable-basic-auth-helpers=MSNT

MSNT 验证器是通过服务消息块(SMB)协议到 Microsoft NT 域数据库的接口。它使用一个小配置文件，叫做 msntauth.conf，它必须放在$prefix/etc 或--sysconfidr 目录。在该配置文件里，最多可以指定 5 个 NT 域控制器。例如：

```
server pdc1_host bdc1_host my_nt_domain
server pdc2_host bdc2_host another_nt_domain
```

默认情况下，MSNT 验证器允许服务器验证任何用户。然而，它也能允许或拒绝指定用户名。假如创建一个 allowusers 文件，仅仅在该文件里列出的用户可允许访问 squid。假如你的 NT 服务器上有很多用户，但仅仅允许少数用户使用 cache，那就可以用到这个功能。另外，可以创建一个 denyusers 文件。任何列举在该文件里的用户会被拒绝访问，这点甚至发生在检查 allowusers 文件之前。

可选择的，还可以把用户名放在 proxy_auth ACL 里，从而允许或拒绝指定用户名，请见 6.1.2.12 章的描述。

附加的文档，请见 helpers/basic_auth/MSNT 目录下的 README.html 文件。

12.2.4 Multi-domain-NTLM

./configure —enable-basic-auth-helpers=multi-domain-NTLM

multi-domain-NTLM 验证器类似于 MSNT，两者都会查询 Windows NT 域数据库。不同于 MSNT 最多查询 5 个域控制器，multi-domain-NTLM 验证器要求用户在其用户名前插入 NT 域名，象这样：

ntdomain\username

multi-domain-NTLM 辅助器程序是个相对较短的 perl 脚本。它依赖于 CPAN 的 Authen::SMB 包。假如你没有在 perl 脚本里硬编码域控制器的主机名，它会利用 Samba 包里的 nmblookup 程序来自动查找它们。

perl 脚本命名为 smb_auth.pl，它在 squid.conf 里看起来如下：

auth_param basic program /usr/local/squid/libexec/smb_auth.pl

multi-domain-NTLM 的文档很少，但假如你熟悉 perl，通过阅读源代码就可以了解更多。

12.2.5 PAM

./configure —enable-basic-auth-helpers=PAM

感觉上，插件式验证模块(PAM)是在验证方式（例如一次性密码， kerberos, smart cards）和要求验证服务的应用（例如 ssh,ftp,imap）之间的胶合剂。系统的/etc/pam.conf 文件描述了对每种应用使用何种验证方式。

为了使用 squid 的 PAM 验证辅助器，你必须将"squid"作为一个服务增加到/etc/pam.conf 文件，并且指定使用哪个 PAM 模块。例如，为了使用 FreeBSD 的 Unix 密码文件，必须将这个放在 pam.conf 里：

squid auth required pam_unix.so try_first_pass

为了检查 Unix 密码数据库，pam_auth 进程必须以 root 运行。这是个安全风险，你必须手工设置可执行 setuid root。假如 pam_auth 不以 root 运行，并且它被配置为检查 Unix 密码数据库，那么每个验证请求都会失败。

PAM 验证器的文档以 man 页形式提供，可在 helpers/basic_auth/PAM 目录下找到。

12.2.6 SASL

./configure —enable-basic-auth-helpers=SASL

简单验证和安全层(SASL)是个 IETF 提议标准，文档在 RFC 2222 里。它是个为面向连接的协议（例如 FTP,SMTP,HTTP）提供安全参数协商的协议。然而，SASL 验证器类似于 PAM 验证器。它使用第三方库接口，查询许多不同的验证数据库。

特别的，squid的SASL验证器要求Cyrus SASL库，它由Carnegie Mellon大学开发。可在这里找到： http://asg.web.cmu.edu/sasl/

可以配置 SASL 验证器来检查传统密码文件，PAM 系统，或任何其他被 CMU 的库支持的数据库。更多信息，请见 helpers/basic_auth/SASL 目录的 README 文件。

12.2.7 SMB

./configure —enable-basic-auth-helpers=SMB

SMB 是另一个对 Microsoft Windows 数据库的验证器。该验证器自身是一个 C 程序。该程序在每次与 Windows 域控制器会话时，执行一个 shell 脚本。这个 shell 脚本包含来自 Samba 包的命令。这样，在使用 SMB 验证器之前，你必须安装 Samba。

SMB 验证器程序，smb_auth 取 Windows 域名作为参数。例如：

auth_param basic program /usr/local/squid/libexec/smb_auth -W MYNTDOMAIN

通过重复-W 选项，可以列举多个域。完全的文档，请见：
http://www.hacom.nl/~richard/software/smb_auth.html

12.2.8 YP

./configure —enable-basic-auth-helpers=YP

YP 验证器检查系统的"Yellow Pages"（例如 NIS）目录。为了在 squid 里使用它，必须在验证器命令行里提供 NIS 域名和密码数据库的名字，通常是 passwd.byname：

auth_param basic program /usr/local/squid/libexec/yp_auth my.nis.domain passwd.byname

yp_auth 程序相对简单，但没有任何文档。

12.2.9 getpwnam

./configure —enable-basic-auth-helpers=getpwnam

该验证器是个简单的到 getpwnam()函数的接口，该函数在 Unix 系统的 C 库里。对给定的用户名，getpwnam()函数查询系统的密码文件。假如使用 YP/NIS，getpwnam()也检查那些数据库。在某些操作系统上，它也利用 PAM 系统。假如你的 cache 用户在 squid 运行的系统上也有登陆帐号，就可使用该验证器。另外，可在密码文件里对 cache 用户建立 nologin 帐号。

12.2.10 winbind

./configure —enable-basic-auth-helpers=winbind

Winbind 是 Samba 套件的功能之一。它允许 Unix 系统利用 Windows NT 的用户帐号信息。winbind 验证器是 Samba winbindd 服务进程的客户端。在使用该验证器之前，必须安装 Samba 和运行 winbindd 服务。

winbind 基本验证器的名字是 wb_basic_auth。它在 squid.conf 里看起来如下：

auth_param basic program /usr/local/squid/libexec/wb_basic_auth

12.2.11 基本验证 API

在 squid 和基本验证器之间的接口非常简单。squid 发送用户名和密码到验证器进程，它们以空格分开并以新行结束。验证器在其 stdin 里读取用户名和密码。在检查信用项后，验证器将 OK 或 ERR 写入 stdout。

任何“不安全的 URL”字符会参照 RFC 1738 规则进行编码。这样，名字 "jack+jill"变成了"jack%2bjill"。squid 接受包含空格的用户名和密码。例如"a password"变成了"a%20password"。在解码用户名和密码后，验证器程序能处理空格和其他的特殊字符。

可在命令行上轻易测试基本验证器。简单的在终端窗口里运行验证器程序，并输入用户名和密码。或者，可以这样做：

% echo "bueller pencil" | ./ncsa_auth /tmp/passwd

OK

如下是个用 perl 写成的简单的验证器模板：

#!/usr/bin/perl -wl

```
use URI::Escape;

$|=1;                           # don't buffer stdout

while (<>) {
        ($u,$p) = split;
        $u = uri_unescape($u);
        $p = uri_unescape($p);
        if (&valid($u,$p)) {
                print "OK";
        } else {
                print "ERR";
        }
}

sub valid {
        my $user = shift;
        my $pass = shift;
        ...
}
```

12.3 HTTP 摘要验证

摘要验证被设计为比基本验证更安全。它广泛利用了加密 hash 函数和其他技巧。本质上，与发送明文密码不同，用户代理发送密码、用户名和其他信息的"消息摘要"（见 RFC 2617 和 O'Reilly's HTTP: The Definitive Guide 的更多信息）。

HTTP 摘要验证支持下列 auth_param 参数：

```
auth_param digest program command
auth_param digest children number
auth_param digest realm string
auth_param digest nonce_garbage_interval time-specification
auth_param digest nonce_max_duration time-specification
auth_param digest nonce_max_count number
auth_param digest nonce_strictness on|off
```

program, children, 和 realm 参数与基本验证的一样。与摘要验证相关的唯一不同参数是 nonce。

nonce 是个特殊的数据串，它偶尔改变。在验证过程中，服务器（这里就是 squid）提供一个 nonce 值到客户端。客户端在产生摘要时要用到这个 nonce 值。没有 nonce 数据，攻击者能简单的拦截和重现(replay)摘要值来获取对 squid 的访问。

nonce_garbage_interval 参数告诉 squid 每隔多久清空 nonce 缓存。默认值是每 5 分钟。对于有许多摘要验证客户端的非常忙的 cache，更频繁的回收 nonce 碎片可能有益。

nonce_max_duration 参数指定每个 nonce 值保持多长时间的有效期。当客户端试图使用某个已过期的 nonce 值时，squid 产生 401（未验证）响应，并随之发送一个新的 nonce 值，以便客户端能重新认证。默认值是 30 分钟。注意任何被截获的 Authorization 头部能被用于 replay 攻击，直到 nonce 值过期。然而将 nonce_max_duration 值设得过低，导致 squid 过频繁的产生 401 响应。对每个 401 响应，客户端和服务器要重新协商它们的验证信用项，这本质上浪费了用户的时间。

nonce_max_count 参数对 nonce 值可使用多少次设置一个上限。在指定数量的请求后，squid 返回 401（未验证）响应和一个新的 nonce 值。默认是 50 个请求。

nonce 计数是另一个设计成阻止 replay 攻击的功能。squid 在 401 响应里发送 qop=auth。这导致用户代理在它们的响应里包含 nonce 计数，并在产生摘要自身时使用这个 nonce 计数。nonce 计数值必须逐次增加。下降的 nonce 计数意味着 replay 攻击。然而，计数可能跳跃的增加，跨过某些数值，例如：5,6,8,9。nonce_strictness 参数决定在这种情形下 squid 如何做。若设置为 on，假如某个 nonce 计数不等于前次 nonce 计数加 1，squid 会返回 401 响应。若设置为 off，squid 允许不连续的 nonce 计数值。

如下是个完整示例：

```
auth_param digest program /usr/local/squid/libexec/digest_pw
auth_param digest children 8
auth_param digest realm Access to Squid
auth_param digest nonce_garbage_interval 10 minutes
auth_param digest nonce_max_duration 45 minutes
auth_param digest nonce_max_count 100
auth_param digest nonce_strictness on

acl KnownUsers proxy_auth REQUIRED
http_access allow KnownUsers
```

下面我将讨论 squid 自带的摘要验证辅助器程序。

12.3.1 password

./configure —enable-auth=digest —enable-digest-auth-helpers=password

这是 squid 摘要验证的简单可参考执行的方法。它展示了如何编写基于摘要验证的辅助器。这个代码简单的从明文文件里读取用户名和密码。该文件的格式类似如下：

```
username:password
```

密码文件的路径是 digest_pw_auth 程序的单一参数，例如：

```
auth_param digest program /usr/local/squid/libexec/digest_pw_auth
        /usr/local/squid/etc/digest_passwd

auth_param digest realm Some Nifty Realm
```

squid 不提供任何工具来维护这种格式的密码文件。假如你选择使用摘要验证，就必须管理自己的密码文件，可使用文本编辑器或 perl 脚本来做到。

12.3.2 摘要验证 API

假如要编写自己的摘要验证辅助器，你必须理解在 squid 和辅助器进程间的通信。数据交换类似于基本验证，但稍微复杂点。

第一个不同是 squid 将用户名和域值，而不是用户名和密码，写往辅助器进程。这些串被引用起来，并以冒号分隔。例如：

```
"bobby":"Tom Landry Middle School"
```

第二个不同是假如用户名有效，辅助器进程返回一个 MD5 摘要串，而不是返回 OK。与基本验证一样，假如用户不存在，或者来自 squid 的输入不可解析，辅助器进程返回 ERR。

辅助器随着用户名，域值和密码返回一个 MD5 摘要。这 3 个串连在一起，并以冒号分割：

```
username:realm:password
```

记住密码不会在 HTTP 请求里发送。辅助器从数据库里（类似于 password 辅助器使用的明文文件）获取用户的密码。例如，假设 Bobby 的密码是 CapeRs。辅助器从 squid 接受用户名和域值，从它的数据库里获取密码，并计算这个串的 MD5 校验和：

```
bobby:Tom Landry Middle School:CapeRs
```

Squid 的源代码包含一个库函数叫做 DigestCalcHA1( ), 它用来执行这个计算。我们可以在终端窗口里来测试这些，并观察辅助器返回什么：

```
% echo 'bobby:CapeRs' > /tmp/pw

% echo bogus_input | digest_pw_auth /tmp/pw
ERR

% echo "nouser":"some realm" | digest_pw_auth /tmp/pw
ERR
```

```
% echo '"bobby":"Tom Landry Middle School"' | digest_pw_auth /tmp/pw
c7ca3efda238c65b2d48684a51baa90e
```

Squid 存储这个 MD5 校验和，并将其用于摘要验证算法的其他部分。注意这个校验和仅在用户改变其密码时才会改变。在 squid 的当前摘要执行里，只要用户保持活跃状态，这些校验和就保存在内存里。假如用户不活跃的时间达到 authenticate_ttl 秒，MD5 校验和可能从 squid 的内存里移除。若该用户下次请求，squid 会要求外部辅助器进程重新计算校验和。

12.4 Microsoft NTLM 验证

NTLM是Microsoft的私有连接验证协议。许多组织，包括squid的开发者，通过少许可用信息以及检查网络传输，已经反向工程了该协议。可以在这里找到一些技术细节：http://www.innovation.ch/java/ntlm.html

NTLM 使用三次握手来验证一个连接。首先，客户端发送请求，它带一对标识符。接着，服务器发送回一个挑战消息（译者注：即用于加密的随机种子）。第三步，客户端再次发送其请求，包含了对这个种子的响应。这时，连接验证成功，在同一连接里的任何进一步的请求，不再需要挑战/响应信息。假如关闭了连接，客户端和服务器必须重复整个三次握手过程。持续连接有助于减少 NTLM 验证的负载。

NTLM 使用加密的 hash 函数和 nonce 值，类似于摘要验证，尽管专家认为 NTLM 要脆弱一些。

NTLM 验证支持下列 auth_param 参数：

```
auth_param ntlm program command
auth_param ntlm children number
auth_param ntlm max_challenge_reuses number
auth_param ntlm max_challenge_lifetime time-specification
```

program 和 children 参数与基本验证和摘要验证的相同。剩余的参数决定 squid 每隔多久重用某个挑战令牌。

max_challenge_reuses 参数指定某个挑战令牌可被重用多少次。默认值是 0，所以这个令牌永不会被重用。增加该值可以减少 squid 和 NTLM 辅助器进程的计算负载，但带来了弱化协议安全的风险。

类似的，max_challenge_lifetime 参数对令牌重用设置了一个时间限制，即使 max_challenge_reuses 次数还没有用完。默认值是 60 秒。

如下是个完整示例：

```
auth_param ntlm program /usr/local/squid/libexec/ntlm_auth foo\bar
```

```
auth_param ntlm children 12
auth_param ntlm max_challenge_reuses 5
auth_param ntlm max_challenge_lifetime 2 minutes

acl KnownUsers proxy_auth REQUIRED
http_access allow KnownUsers
```

Squid 自带了下列 NTLM 验证辅助器程序：

12.4.1 SMB

```
./configure —enable-auth=ntlm —enable-ntlm-auth-helpers=SMB
```

NTLM 的服务消息块(SMB)验证器与基本验证的相似。用户简单的提供他们的 windows NT 域名，用户名和密码即可。该验证器能在多个域控制器间负载均衡。域和控制器名字出现在命令行中：

```
auth_param ntlm program /usr/local/squid/libexec/ntlm_auth
    domain\controller [domain\controller ...]
```

12.4.2 winbind

```
./configure —enable-auth=ntlm —enable-ntlm-auth-helpers=winbind
```

该验证器类似于基本验证的 winbind。两者都要求安装和运行了 Samba winbindd 服务。NTLM 的 winbind 验证器名字是 wb_nltm_auth。它在 squid.conf 里的配置看起来如下：

```
auth_param basic program /usr/local/squid/libexec/wb_ntlm_auth
```

12.4.3 NTLM 验证 API

在 squid 和 NTLM 验证器之间的通信相对于基本和摘要验证而言，要复杂得多。理由之一是每个辅助器进程实际创建了它自己的挑战令牌。这样，辅助器变得与状态相关，squid 必须记住哪个连接属于哪个辅助器。

Squid 和辅助器进程使用一些 2 字符的代码来指示它们正在发送什么。这些代码如下：

YR

在 squid 需要新的挑战令牌时，它发送这个给辅助器。这总是在两个进程间的第一个通信。它也可能在 squid 需要新挑战令牌的任何时候发生，归咎于 auth_param max_challenge_lifetime 和 max_challenge_uses 参数的设置。辅助器响应一个 TT 消息。

TT 挑战令牌

辅助器发回这个消息给 squid，包含了一个挑战令牌。它是对 YR 请求的响应。挑战令牌用 base64 编码，在 RFC 2045 里有定义。

KK 信用项

当 squid 想要验证某个用户的信用项时，它发送这个到辅助器。辅助器响应如下几个代码：AF, NA, BH, 或 LD。

AF 用户名

当用户的验证信用项有效时，辅助器发回这个消息给 squid。辅助器在本消息里发送用户名，是因为 squid 不去尝试解码 NTLM 验证头部。

NA 理由

当用户的信用项无效时，辅助器发回这个消息给 squid。它也包含了一个“理由”字符串，squid 能将其显示到错误页面。

BH 理由

当验证过程失败时，辅助器发回这个消息给 squid。这点可能发生在，例如，辅助器进程无法与 windows NT 域控制器通信的时候。squid 拒绝用户请求。

LD 用户名

这个辅助器到 squid 的响应类似于 BH，除了 squid 允许用户请求之外。类似于 AF，它返回用户名。为了使用该功能，必须在编译 squid 时使用 —enable-ntlm-fail-open 选项。

既然该协议相对复杂，你最好从包含在 squid 源代码发布里的 2 个基本验证器起步。no_check 辅助器用 perl 写的，fakeauth 用 C 写的。可以在 helpers/ntlm_auth 目录找到它们。

12.5 外部 ACL

在版本 2.5，Squid 包含了一个新功能，叫做外部 ACL。这些 ACL 元素在外部辅助器进程里被执行。你指示 squid 将某些信息写往辅助器，然后辅助器以 OK 或 ERR 来响应 squid。请参考 6.1.3 章关于 external_acl_type 语法的描述。这里，我仅仅讨论一些特殊的外部 ACL 辅助器程序，它们随着 squid 的源代码发布。

12.5.1 ip_user

./configure —enable-external-acl-helpers=ip_user

该辅助器读取用户名和客户端 IP 地址作为输入。它根据配置文件来检查这 2 个值，以决定其是否有效。为了使用这个 ACL 辅助器，要在 squid.conf 里增加如下行：

external_acl_type ip_user_helper %SRC %LOGIN
/usr/local/squid/libexec/ip_user -f /usr/local/squid/etc/ip_user.conf

acl AclName external ip_user_helper

对每个请求，%SRC 替换成客户端的 IP 地址，%LOGIN 替换成用户名。ip_user.conf 配置文件有如下格式：

ip_addr[/mask] user|@group|ALL|NONE

例如：

127.0.0.1 ALL
192.168.1.0/24 bob
10.8.1.0/24 @lusers
172.16.0.0/16 NONE

该配置文件导致 ip_user 对任何来自 127.0.0.1 的请求返回 OK，对来自 192.168.1.0/24 网络的 Bob 的请求返回 OK，对来自 10.8.1.0/24 网络的，位于 luser 组里的任何用户名返回 OK。对来自 172.16.0.0/16 网络的任何请求返回 ERR。它也对任何不在这个列表里出现的地址和用户名对返回 ERR。

12.5.2 ldap_group

./configure —enable-external-acl-helpers=ldap_group

该辅助器决定是否某个用户属于一个特殊的 LDAP 组。在 acl 行里指定 LDAP 组名。它可能在你的配置文件里看起来如下：

external_acl_type ldap_group_helper %LOGIN /usr/local/squid/libexec/squid_ldap_group
-b "ou=people,dc=example,dc=com" ldap.example.com

acl AclName external ldap_group_helper GroupRDN ...

注意为了编译 squid_ldap_group 辅助器程序，你必须在系统中安装 OpenLDAP 库 (http://www.openldap.org)。

12.5.3 unix_group

./configure —enable-external-acl-helpers=unix_group

该辅助器在 Unix 组数据库（例如/etc/group 文件）里查找用户名。在辅助器的命令行指定要检查的组：

external_acl_type unix_group_helper %LOGIN
    /usr/local/squid/libexec/check_group -g group1 -g group2 ...

acl AclName external unix_group_helper

另外，可在 acl 行指定组。这样就允许对不同的组使用相同的辅助器：

external_acl_type unix_group_helper %LOGIN /usr/local/squid/libexec/check_group

acl AclName1 external unix_group_helper group1 ...
acl AclName2 external unix_group_helper group2 ...

12.5.4 wbinfo_group

./configure —enable-external-acl-helpers=wbinfo_group

该辅助器是个简短的 perl 脚本，它利用了 Samba 包的 wbinfo 程序。wbinfo 是 winbindd 服务的客户端。对每个请求，该脚本期待一个单一的 Unix 组名跟随在用户名后。这样，必须在 acl 行里放置组名：

external_acl_type wbinfo_group_helper %LOGIN /usr/local/squid/libexec/wbinfo_group.pl

acl AclName external wbinfo_group_helper group

12.5.5 winbind_group

./configure —enable-external-acl-helpers=winbind_group

该辅助器用 C 写成，也要求 winbindd 服务提供的 windows NT 用户名的组成员关系。它基于基本验证和 NTLM 验证的 winbind 辅助器。可在 acl 命令行指定多个组名：

external_acl_type winbind_group_helper %LOGIN /usr/local/squid/libexec/wb_check_group

acl AclName external winbind_group_helper group1 group2 ...

12.5.6 编写自己的外部 ACL 辅助器

外部 ACL 接口提供了许多兼容性，可以用它来执行几乎任何不被 squid 内在支持的访问控制检测。编写外部 ACL 分 2 步走。首先，你必须决定辅助器程序需要对什么样的请求信息来作出决定。在 external_acl_type 行上放置相应的关键字，紧跟着辅助器程序的路径。例如，假如你想编写一个外部 ACL 辅助器，它使用了客户端 IP 地址，用户名，和 Host 头部的值，那就可这样写：

```
external_acl_type MyAclHelper %SRC %LOGIN %{Host}
    /usr/local/squid/libexec/myaclhelper
```

第 2 步是编写 myaclhelper 程序。它必须在 stdin 里读取请求元素，作出它自己的决定，然后将 OK 或 ERR 写往 stdout。继续以前的示例，该 perl 脚本描述了如何去做：

```
#!/usr/bin/perl -wl

require 'shellwords.pl';
$|=1;

while (<>) {
    ($ip,$name,$host) = &shellwords;
    if (&valid($ip,$name,$host)) {
        print "OK";
    } else {
        print "ERR";
    }
}

sub valid {
    my $ip = shift;
    my $name = shift;
    my $host = shift;
    ...
}
```

参考 6.1.3 章关于从 squid 传递到辅助器的元素列表（%SRC, %LOGIN 等）。注意当某个元素包含空格时，squid 会在双引号里封装它。如同示例显示的那样，可以使用 perl 的 shellwords 库来解析被双引号封装的元素。

当然，为了利用外部 ACL，你必须在某个 acl 行里引入它。无论何时，若外部辅助器返回 OK，则 ACL 元素匹配成功。

外部 ACL 辅助器接口允许从辅助器提交附加的信息到 squid（在 OK/ERR 行）。这些以

keyword=value 对的形式出现。例如：

OK user=hank

当前 squid 了解的唯一关键字是 error 和 user。假如设置了 user 值，squid 将它用于 access.log。squid 当前没有用到 error 值。