Squid 中文权威指南

（第 11 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 11 章 重定向器

重定向器是squid的外部程序，它重写来自客户请求的URI。例如，尽管某个用户请求这个页面：http://www.example.com/page1.html ，重定向器可以将请求改变到别的地方，例如：http://www.example.com/page2.html 。squid自动抓取新的URI，就像是客户端的原始请求一样。假如响应可被缓存，squid将它存储在新的URI下。

重定向功能允许你执行与 squid 相关的许多有趣事情。许多站点使用它们实现如下目的：访问控制，移除广告，本地镜像，甚至用以绕开浏览器的 bug。

关于使用重定向器进行访问控制的好处之一是，你可以将用户的请求重定向到某个页面，这个页面详细解释为何她的请求被拒绝。你也会发现重定向器比 squid 内建的访问控制提供更多的弹性。然而不久你会看到，重定向器并不能访问包含在客户请求里的完整信息。

许多人使用重定向器来过滤 web 页面广告。大部分情形下，可以将对 GIF 或 JPEG 广告图片的请求，改变为请求位于本地服务器上的，小而空的图片。这样，广告就消失了，然而不会影响页面布局。

所以在本质上，重定向器其实就是一个程序，它从标准输入里读取 URI 和其他信息，并将新的 URI 写往标准输出。Per 和 Python 是写重定向器的流行语言，尽管某些作者使用编译性语言（例如 C）以求更好的性能。

Squid 的源代码没有包含任何重定向程序。作为管理员，你有责任编写自己的重定向器，或者下载别人编写的。该章开头部分描述在 squid 和重定向进程之间的接口。我也提供几个简单的 Perl 重定向器示例。假如你志在使用别人的重定向器，而不是自己编写，请跳到 11.3 章。

11.1 重定向器接口

重定向器在其标准输入里，每次一行的接受来自 squid 的数据。每行包括下列四个元素，以空格分开：

1)请求 URI
2)客户 IP 地址和完全可验证域名
3)用户名，通过 RFC 1413 ident 或代理验证
4)HTTP 请求方式

例如：
http://www.example.com/page1.html 192.168.2.3/user.host.name jabroni GET

请求 URI 取自客户请求，包括任何查询条件。然而，分段标记（例如#字符和随后的文

本）被移除了。

第二个元素包含客户 IP 地址，和可选的完整可验证域名（FQDN）。假如激活了 log_fqdn 指令或使用了 srcdomain ACL 元素，FQDN 才会设置。尽管那样，FQDN 也许仍未知，因为客户网络管理员没有在其 DNS 里正确的设置反向指针区域。假如 squid 不知道客户的 FQDN，它用一个短横线(-)代替。例如：

http://www.example.com/page1.html 192.168.2.3/- jabroni GET

假如 squid 了解请求背后的用户名，客户 ident 域才会设置。假如使用了代理验证，ident ACL 元素，或激活了 ident_lookup_access，这点才会发生。然而请记住，ident_lookup_access 指令不会导致 squid 延缓请求处理。换句话说，假如你激活了该指令，但没有使用访问控制，squid 在写往重定向进程时，也许仍不知道用户名。假如 squid 不知道用户名，它显示一个短横线(-)。例如：

http://www.example.com/page1.html 192.168.2.3/- - GET

Squid 从重定向进程里读回一个元素：URI。假如 squid 读取一个空行，原始 URI 保留不变。

重定向程序永不退出，除非在标准输入里发生 end-of-file。假如重定向进程确实过早退出，squid 在 cache.log 里写一条警告信息：

WARNING: redirector #2 (FD 18) exited

假如 50%的重定向进程过早退出，squid 会以致命错误消息退出。

11.1.1 处理包含空格的 URI

假如请求 URI 包含空格，并且 uri_whitespace 指令设置为 allow，那么任何在 URI 里的空格被递交到重定向器。如果重定向器的解析器很简单，那它在这种情况下会很困惑。在使用重定向器时，有 2 个选项来处理 URI 里的空格。

一个选项是设置 uri_whitespace 指令为任何值，除了 allow。默认的设置 strip，在大多数情况下可能是个好的选择，因为 squid 在解析 HTTP 请求时，它简单的从 URI 里删除空格。该指令的其他值的信息，请见附录 A。

假如上述方法不可行，你必须确保重定向器的解析器足够巧妙，以检测额外的元素。例如，假如它发现接受自 squid 的行里的元素不止 4 个，它会假设最后 3 个元素是 IP 地址，ident，和请求方式。在最后 3 个元素之前的任何东西，组成请求 URI。

11.1.2 产生 HTTP 重定向消息

当某个重定向器改变客户的 URI 时，它通常不知道 squid 决定抓取新的资源。也就是说，这点违背了 HTTP RFC。假如你想友好而保留兼容性，有一个小窍门可让 squid 返回 HTTP 重定向消息。简单的让重定向器在新的 URI 前面插入 301:, 302:, 303:, 或 307:。

例如，假如重定向器在其标准输出里写如下行：

301:http://www.example.com/page2.html

Squid 返回类似如下的响应到客户端：

HTTP/1.0 301 Moved Permanently

Server: squid/2.5.STABLE4

Date: Mon, 29 Sep 2003 04:06:23 GMT

Content-Length: 0

Location: http://www.example.com/page2.html

X-Cache: MISS from zoidberg

Proxy-Connection: close

11.2 重定向器示例

示例 11-1 是用 perl 写的非常简单的重定向器。它的目的是，将对 squid-cache.org 站点的 HTTP 请求，发送到位于澳洲的本地镜像站点。对看起来是请求 www.squid-cache.org 或其镜像站点之一的 URI，该脚本输出新的 URI，将主机名设为 www1.au.squid-cache.org

重定向程序遇到的通用问题是缓存 I/O。注意这里我确保 stdout 不可缓存。

Example 11-1. A simple redirector in Perl

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

$|=1;      # don't buffer the output

while (<>) {
```

```
        ($uri,$client,$ident,$method) = ( );

        ($uri,$client,$ident,$method) = split;

        next unless ($uri =~ m,^http://.*\.squid-cache\.org(\S*),);

        $uri = "http://www1.au.squid-cache.org$1";

} continue {

        print "$uri";

}
```

示例 11-2 是另一个稍微复杂点的脚本。在这里我做了个初步尝试，当 URI 包含不当词汇时，拒绝该请求。该脚本论证了解析输入域的另一个方法。假如没有得到所有 5 个请求域，重定向器返回一个空行，请求保留不变。

该示例也优待某些用户。假如 ident 等于"BigBoss," ，或来自 192.168.4.0 子网，请求就直接通过。最后，我使用 301:窍门来让 squid 返回 HTTP 重定向消息到客户端。注意，本程序既非有效的，又非足够巧妙的，来正确拒绝坏请求。

Example 11-2. A slightly less simple redirector in Perl

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

$|=1;     # don't buffer the output



$DENIED = "http://www.example.com/denied.html";

&load_word_list( );



while (<>) {

        unless (m,(\S+) (\S+)/(\S+) (\S+) (\S+),) {

                $uri = '';

                next;
```

```
        }

        $uri = $1;

        $ipaddr = $2;

        #$fqdn = $3;

        $ident = $4;

        #$method = $5;

        next if ($ident eq 'TheBoss');

        next if ($ipaddr =~ /^192\.168\.4\./);

        $uri = "301:$DENIED" if &word_match($uri);

} continue {

        print "$uri";

}


sub load_word_list {

        @words = qw(sex drugs rock roll);

}


sub word_match {

        my $uri = shift;

        foreach $w (@words) { return 1 if ($uri =~ /$w/); }

        return 0;

}
```

关于编写自己的重定向器的更多主意，我推荐阅读 11.5 章里提到的重定向器的源代码。

11.3 重定向器池

重定向器可能经过任意长的时间才返回应答。例如，它可能要查询数据库，搜索正则表达式的长列表，或进行复杂的计算。squid 使用重定向进程池以便它们能并行工作。当某个重定向器忙时，squid 将请求递交给另一个。

对每个新请求，squid 按顺序检查重定向进程池。它将请求提交给第一个空闲进程。假如请求率非常低，第一个重定向器也许自己能处理所有请求。

可以使用 redirect_children 指令来控制重定向器池的 size。默认值是 5 个进程。注意 squid 不会根据负载来动态的增或减进程池的 size。这样，建议你适当的放宽 size 限制。假如所有的重定向器忙碌，squid 会将请求排队。假如队列变得太大（大于进程池 size 的 2 倍），squid 以致命错误消息退出：

FATAL: Too many queued redirector requests

在该情形下，你必须增加重定向器池的 size，或改变其他东西以让重定向器能更快的处理请求。你可以使用 cache 管理器的 redirector 页面来发现是否有太少，或太多重定向器在运行。例如：

% squidclient mgr:redirector

...

Redirector Statistics:

program: /usr/local/squid/bin/myredir

number running: 5 of 5

requests sent: 147

replies received: 142

queue length: 2

avg service time: 953.83 msec

FD PID # Requests Flags Time Offset Request

1	10	35200	46	AB	0.902	0 http://...
2	11	35201	29	AB	0.401	0 http://...
3	12	35202	25	AB	1.009	1 cache_o...
4	14	35203	25	AB	0.555	0 http://...
5	15	35204	21	AB	0.222	0 http://...

在该示例里，假如你见到最后一个重定向器的请求数量，几乎和倒数第二个一样多，就应该增加重定向器池的 size。另一方面，假如你见到许多重定向器没有请求，就该减少进程池的 size。

11.4 配置 Squid

下列 5 个 squid.conf 指令，控制 squid 里的重定向器的行为。

11.4.1 redirect_program

redirect_program 指令指定重定向程序的命令行。例如：

redirect_program /usr/local/squid/bin/my_redirector -xyz

注意，重定向程序必须能被 squid 的用户 ID 执行。假如因为某些理由，squid 不能执行重定向器，你将在 cache.log 里见到错误消息。例如：

ipcCreate: /usr/local/squid/bin/my_redirector: (13) Permission denied

因为 squid 的工作方式，主 squid 进程可能不知道执行重定向程序的问题所在。squid 不会检测到错误，直到它试图写一个请求和读到一个响应。然后它打印：

WARNING: redirector #1 (FD 6) exited

这样，假如你见到发送给 squid 的第一个请求的如此错误，请仔细检查 cache.log 的其他错误，并确保重定向程序可被 squid 执行。

11.4.2 redirect_children

redirect_children 指令指定 squid 应该开启多少重定向进程。例如：

redirect_children 20

当所有重定向器同时忙碌时，squid 会通过 cache.log 发出警告：

WARNING: All redirector processes are busy.
WARNING: 1 pending requests queued.

假如见到这样的警告，你应该增加子进程的数量，并重启（或 reconfigure）Squid。假如队列的 size 变成重定向器数量的 2 倍，squid 以致命错误退出。

不要试图将 redirect_children 设为 0 来禁止 squid 使用重定向器。简单的从 squid.conf 里删除 redirect_program 行就可以了。

11.4.3 redirect_rewrites_host_header

正常情况下，squid 在使用重定向器时，会更新请求的 Host 头部。也就是说，假如重定向器返回的新 URI 里包含不同的主机名，squid 将新的主机名放在 Host 头部。假如使用 squid 作为代理人（surrogate，见 15 章），你也许想将 redirect_rewrites_host_header 指令设为 off 来禁止这种行为：

redirect_rewrites_host_header off

11.4.4 redirector_access

正常情况下，squid 将每个请求发送往重定向器。然而，可以使用 redirector_access 规则来有选择的发送某些请求。该语法与 http_access 相同：

redirector_access allow|deny [!]ACLname ...

例如：

acl Foo src 192.168.1.0/24
acl All src 0/0

redirector_access deny Foo
redirector_access allow All

在该情形里，对任何匹配 Foo ACL 的请求，Squid 跳过重定向器。

11.4.5 redirector_bypass

假如激活了 redirector_bypass 指令，squid 在所有重定向器忙碌时，会绕过它们。正常情况下，squid 将未处理请求排队，直到某个重定向进程可用。假如该队列增长得太大，squid 以致命错误退出。激活该指令确保 squid 永不会达到那种状态。

当然，折衷点是当负载高时，某些用户请求可能不会被重定向。假如这样对你没问题，简单的激活该指令即可：

redirector_bypass on

11.5 流行的重定向器

我已经提过，squid 的源代码未包含任何重定向器。然而，通过 http://www.squid-cache.org 的 Related Software 页面的链接，可以找到许多有用的第三方重定向器。如下是一些流行的重定向器：

11.5.1 Squirm

http://squirm.foote.com.au/

Squirm 出自 Chris Foote 之手。它用 C 编写，并在 GNU 公用许可证（GPL）下发布源代码。Squirm 的功能包括：

1)非常快速，最少的内存使用
2)完全正则表达式匹配和替换
3)对不同的客户组应用不同的重定向列表
4)命令行的交互式模式的测试
5)防故障模式，假如配置文件包含错误，它不对请求作任何改变
6)将 debug 信息，错误信息，和其他更多信息写往不同日志文件

11.5.2 Jesred

http://www.linofee.org/~elkner/webtools/jesred/

Jesred 出自 Jens Elkner 之手。它用 C 编写，基于 Squirm 而来，也在 GNU GPL 下发行。其功能包括：

1)比 Squirm 更快，但内存使用稍多
2)在运行时能重读配置文件

3)完全正则表达式匹配和替换
4)防故障模式，假如配置文件包含错误，它不对请求作任何改变
5)可选择的记录重写请求到日志文件

11.5.3 squidGuard

http://www.squidguard.org/

squidGuard 出自 Tele Danmark InterNordia 的 Pal Baltzersen 和 Lars Erik Haland。它在 GNU GPL 下发行。作者确保 squidGuard 在现代 Unix 系统上能轻松编译。他们的站点包含了许多好文档。如下是 squidGuard 的一些功能：

1)高度可配置；你能在不同的时间，应用不同的规则到不同的客户组
2)URI 置换，而非仅仅替换
3)printf 形式的置换，允许递交参数给 CGI 脚本来定制消息
4)支持重定向器的 301/302/303/307 HTTP 重定向状态码功能
5)可选择的重写规则日志记录

在 squidGuard 的站点，还可以找到超过 100,000 个站点的黑名单，它们以色情，暴力，毒品，黑客，广告，和其他更多形式来分类。

11.5.4 AdZapper

http://www.adzapper.sourceforge.net

AdZapper 是个流行的重定向器，因其明确的目标是从 HTML 页面里移除广告。它是 Cameron Simpson 所写的 Perl 脚本。AdZapper 能阻止横幅图片，弹出式窗口，flash 动画，页面计数器，和 web bug。该脚本包含正则表达式列表，用以匹配某些已知包含广告，弹窗等的 URI。Cameron 定期更新该脚本的模式匹配。你也能维护你自己的模式匹配列表。