Dvas0004's Blog

If at first you don't succeed; call it version 1.0

SQUID + GreasySpoon : enhancing your proxy deployment with content adaptation

Filed under: BlueCoat, Linux, Open Source — 1 Comment
February 28, 2011

i
2 Votes

When comparing the two proxy solutions I am most familiar with, these being BlueCoat ProxySG and SQUID, the most striking difference is the capability of the bluecoat to easily change and modify the traffic passing through it. For the Bluecoat-savvy of you, adding a "Web Access" and "Web Content" layer in policy allows you to modify traffic such as adding Headers, cookies, notify pages, and so on. This sort of modification is what is known as "Content Adaptation". A SQUID article explains the various options available to SQUID users in doing this:

http://wiki.squid-cache.org/SquidFaq/ContentAdaptation

It definitely doesn't look very easy to do this. The easiest way I've found is using an Internet Content Adaptation Protocol (ICAP) server to modify the traffic for SQUID. I wont go into much details on ICAP, in a nutshell the SQUID proxy sends traffic of interest (such as HTTP) over to the ICAP server, which then parses it, modifies it, and sends it back to the SQUID server. This opens up a lot of opportunities for achieving the same sort of Bluecoat functionality I mentioned previously… such as adding headers, cookies, inserting company headers within a website, and much more.

The easiest and most flexible open source ICAP server i've come across is GreasySpoon:

http://greasyspoon.sourceforge.net/

It requires some programming knowledge so it's not as easy for first-timers but the upside is the possibilities are endless… apart from having a good performance and being cross-platform. If you are going to go through with setting this up, I advise reading through the website, they have some good documentation and script samples.

In this article I'll be logging my test setup where I've used a CentOS 5 machine to host a SQUID proxy and a GreasySpoon server. As a test case, I wanted to instruct GreasySpoon to insert a header into YouTube server responses to force clients to use the HTML5 version of the YouTube site.

- Setting up SQUID proxy server

The first steps is installing a SQUID proxy on the server. In order to include ICAP functionality you need a later SQUID version (3.x). The SQUID versions I found in the CentOS repositories where v2.x, so this necessitated building SQUID from source. The only two pre-requisite packages I needed to download to do this were *gcc* and *gcc-c++*. From there, the process is quite normal:

- Download latest package (v3.1 in my case)
- Run **./configure –enable-icap-client** to enable ICAP functionality on the SQUID proxy
- Run **make** and **make install**
- It should install successfully. Modify the squid conf file to your needs and start the proxy
- Test the proxy by using a browser pointing to the proxy IP

- Setting up the GreasySpoon ICAP server

The installation of GreasySpoon is a breeze, the only issue to point out is to make sure you have the correct Java version installed, else the inbuilt JavaScript engine will not be enabled, leaving you with no languages to use to modify the traffic.

In my case, CentOS already had OpenJDK installed, but I ran into problems anyway because the JavaScript language was not selectable. I needed to download the Sun Java package and install that.

Download the tar.gz package from greasyspoon, extract this, and modify the file "greasyspoon" to point the JAVA_HOME variable to the java home directory, in my case **/usr/java/jre1.6.0/**.

To start greasyspoon give executable permission to the greasyspoon file: **chmod +x greasyspoon**. Then start the server: **./greasyspoon start**

You should now be able to reach the admin interface of the server by visiting http://localhost:8088 on the server.

To double check, run netstat to make sure the server is listening on port **1344**.

- Set up the SQUID + GreasySpoon interaction

This part of the setup informs SQUID to send traffic over to the GreasySpoon server. The basic instructions to do this are already explained here:

http://greasyspoon.sourceforge.net/setup_proxy.html

In a production environment you may want to modify this configuration a bit so not all traffic is sent to the ICAP server, usually only a subset of traffic should be sent.

- Writing a GreasySpoon script to modify HTTP traffic to youtube.com

In my case, I wanted GreasySpoon to check for the presence of a certain Cookie (pref=f2) and if not there, instruct the client to set this cookie using the Set-Cookie HTTP header. This cookie controls if youtube should be seen in HTML5 or not.

In the greasyspoon admin interface, navigate to the tab "greasyspoon scripts" > responses scripts > new script. Add a name and leave the language as ECMAScript. Here's the script itself:

```
//————————————————————————————-
// This is a GreasySpoon script.
// ————————————————————————————
// WHAT IT DOES:force HTML5 version of youtube
// ————————————————————————————
// ==ServerScript==
// @name          youtube_HTML_5
// @status on
// @description    force browser to request the HTML5 version of youtube
// @include       .*youtube.*
// @exclude
// @responsecode    200
// ==/ServerScript==
// ————————————————————————————
// Available elements provided through ICAP server:
// ——————
// requestedurl  :   (String) Requested URL
// requestheader  :  (String)HTTP request header
// responseheader :  (String)HTTP response header
// httpresponse   :  (String)HTTP response body
// user_id       :  (String)user id (login or user ip address)
// user_group     :  (String)user group or user fqdn
// sharedcache    :  (hashtable<String, Object>) shared table between all scripts
// trace         :  (String) variable for debug output – requires to set log level to FINE
// ——————
   headerstring = "Cookie: ";
   c = requestheader.indexOf(headerstring) + headerstring.length;
   c1 = requestheader.indexOf("\r\n", c);
   var Cookiestring = requestheader.substring(c,c1);

        if (Cookiestring.indexOf("f2")<0){
           responseheader = responseheader + "Set-Cookie: PREF=f2=40000000; path=/;
domain=.youtube.com;\r\n";
          }
```

Most of it is just comments but pay attention to the **include** and **exclude** comments since they control which sites the script will be applied to.

The rest of the comments describe which variables are available to you the programmer to use.

The actual script (starting headerstring="Cookie: ") is an adaptation of a sample script on the site which basically just checks for the presence of the cookie, and if not found, sends the Set-Cookie header to the client.

Save and enable the script.

That's about it. You can see if the script is being applied from the "Data" tab > logs > access logs section of the greasyspoon interface.

This was just an example, just to show that with a bit of persistence and programming a free, open-source solution can match the functionality and flexibility of much more expensive commercial solutions. Of course, it's not as easy to setup, use and maintain, but I still think this is a fantastic tool and setup that gives any network admin great granularity of control over his proxy traffic

PS : greasyspoon can serve as a very flexible ICAP server for Bluecoat also… all that's needed is a web content rule that forwards traffic via ICAP to the GreasySpoon server.

Tags: bluecoat, centos, icap, Linux, proxySG, squid

Comments RSS (Really Simple Syndication) feed

1 Trackback or Pingback for this entry:

SQUID transparent SSL interception « Dvas0004's Blog

0

0

i
Rate This

[...] http://dvas0004.wordpress.com/2011/02/28/squid-greasyspoon-enhancing-your-proxy-deployment-with-cont… [...]
