Simpletons Guide to Web Server Analysis

Welcome to the wonderful world of web server usage analysis! This guide is intended to provide the necessary background and insight to how web server analysis works, things to look for and things to watch out for. Specifically, this guide is intended for the users of the Webalizer, but can be applied to most any analysis package out there. If you are new to web server analysis, or just want to find out how things work, then this guide is for you.

Hit me Please!

Ok, so you got a web site and you want to know if anybody is looking at it, and if so, what they are looking at and how many times. Lucky for you, (most) every web server keeps a log of what it's doing, so you can just go look and see. The logs are just plain ASCII text files, so any text editor or viewer would work just fine. Each time someone (using a web browser) asks for one of your web pages, or any componant thereof (known as URLs, or *Uniform Resource Locator*), the web server will write a line to the end of the log representing that request. Unfortunately, the raw logs are rather crpytic for everyday humans to read. While you might be able to determine *if* anybody was looking at your web site, any other information would require some sort of processing to determine. A typical log entry might look something like the following:

```
192.168.45.13 - - [24/May/2005:11:20:39 -0400] "GET /mypage.html HTTP/1.1" 200 117
```

This represents a request from a computer with the IP address 192.168.45.13 for the URL **/mypage.html** on the web server. It also gives the time and date the request was made, the type of request, the result code for that request and how many bytes were sent to the remote browser. There will be a line similar to this one for each and every request made to the web server over the period covered by the log. A '**Hit**' is another way to say '*request made to the server*', so as you may have noticed, each line in the log represents a '**Hit**'. If you want to know how many Hits your server received, just count the number of lines in the log. And since each log line represents a request for a *specific* URL, from a *specific* IP address, you can easily figure out how many hits you got for each of your web pages or how many hits you received from a particular IP address by just counting the lines in the log that contain them. Yes, it really is that simple. And while you *could* do this manually with a text editor or other simple text processing tools, it is much more practical and easier to use a program specifially designed to analyze the logs for you, such as the Webalizer. They take the work out of it for you, provide totals for many other aspects of your server, and allow you to visualize the data in a way not possible by just looking at the raw logs.

How does it all work?!?

Well, to understand what you can analyze, you really should know what information is provided by your web server and how it gets there. At the very least, you should know how the HTTP (*HyperText Transport Protocol*) protocol works, and it's strengths and weaknesses. At it's simplest, a web server just sits there listening on the network for a web browser to make a request. Once a request is received, the server processes it and then sends something back to the requesting browser (and as explained above, the request is logged to a log file). These requests are typically for some URL, although there are other types of information a browser can request, such as server type, HTTP protocol versions supported, modification dates, etc., but those types are not as common. To visualize the interaction between server, browser and web pages, lets use an example to illustrate the information flow. Imagine a simple web page, '**mypage.html**', which is a HTML web page that contains two graphic images, '**myimage1.jpg**', and '**myimage2.jpg**'. A typical server/browser interaction might go something like this:

- The web browser asks for the URL **mypage.html**.
- The server sees the request and sends back the HTML page.
- The web browser notices that there are two inline graphic links in the HTML page, so it asks for the first one, **myimage1.jpg**.
- The server sees the request and sends back the graphic image.
- The web browser then asks for the second image, **myimage2.jpg**.
- The server sees the request and sends back the graphic image.

- The browser displays the web page and graphics for the user.

In the web server log, the following lines would be added:

```
192.168.45.13 - - [24/May/2005:11:20:39 -0400] "GET /mypage.html HTTP/1.1" 200 117
192.168.45.13 - - [24/May/2005:11:20:40 -0400] "GET /myimage1.jpg HTTP/1.1" 200 231
192.168.45.13 - - [24/May/2005:11:20:41 -0400] "GET /myimage2.jpg HTTP/1.1" 200 432
```

So what can we gather from this exchange? Well, based on the what we learned above, we can count the number of lines in the log file and determine that the server recveived 3 hits during the period that this log file covers. We can also calculate the number of hits each URL received (in this case, 1 hit each). Along the same lines, we can see that the server received 3 hits from the IP address 192.168.45.13, and when those requests were received. The two numbers at the end of each line represent the *response code* and the *number of bytes* sent back to the requestor. The response code is how the web server indicates how it handled the request, and the codes are defined as part of the HTTP protocol. In this example, they are all 200, which means everything went OK. One response code you may be very familiar with is the all too common '**404 - Not Found**', which means that the requested URL could not be found on the server. There are several other response codes defined, however these two are the most common.

And that, in a nutshell, is about all you can accurately determine from the logs. "But wait!" you might be screaming, "most analysis program have lots of other numbers displayed!", and you would be right. Some more obscure numbers can be calculated, like the number of different response codes, number of hits within a given time period, total number of bytes sent to remote browsers, etc.. Other numbers can be implied based on certain assumptions, however those cannot be considered entirely accurate, and some can even be wildly inaccurate. Other log formats might be used by a web server as well, which provide additional information above what the CLF format does, and those will be discussed shortly. For now, just realize that the only thing you can really, accurately determine is what IP address requested which URL, and when it requested that URL, as shown in the example above.

The Good, the Bad and the Ugly

So now you have a good grasp of how your web server works and what information can be obtained from its logs, like number of hits (to the server and to individual URLs), number of IP addresses making the requests (and how many hits each IP address made), and when those requests were made. Given just that information, you can answer questions such as "What is the most popular URL on my site?", "What was the next most popular URL?", "What IP address made the most requests to my server?", and "How busy was my server during this time period?". Most analysis programs will also make it easy to answer such questions as "What time of day is my web server the most active?", or "What day of the week is the busiest?". They can give you an insight into usage patterns that may not be apparent by just looking at the raw logs. All of these questions can be answered with completely accurate answers, based just on the simple analysis of your web server logs. That's the good news!

The bad news? Well, with all the things you can determine by looking at your logs, there are a lot of things you can't acccurately calculate. Unfortunately, some analysis programs lead you to believe otherwise, and forget to mention (particularly commercial packages) that these are not much more than assumptions and cannot be considered at all accurate. Like what? you ask.. well, how about those things that some programs call 'user trails' or 'paths', that are supposed to tell you what pages and in what order a user travelled through your site. Or how about the length of time a user spends on your site. Another less than accurate metric would be that of 'visits', or how many users 'visited' your site during a given time period. All of these cannot be accurately calculated, for a couple of different reasons.. lets look at some of them:

- **The HTTP protocol is stateless**

  In a typical computer program that you run on your own machine, you can always determine what the user is doing. They log in, do some stuff, and when finished, they log out. The HTTP protocol however is different. Your web server only sees requests from some remote IP address. The remote address connects, sends a request, receives a response and then disconnects. The web server has no idea what the remote side is doing

between these requests, or even what it did with the response sent to it. This makes it impossible to determine things like how long a user spends on your site. For example, if an IP address makes a request to your server for your home page, then 15 minutes later makes a request for some other page on your site, can you determine how long the user had been at your site? The answer is of course **No!**. Just because 15 minutes expired between requests, you have no idea what the remote address was doing between those two requests. They could have hit your site, then immediately gone somewhere else on the web, only to come back 15 minutes later to request another page. Some analysis packages will say that the user stayed on your site for at least 15 minutes plus some 'fudge' time for viewing the last page requested (like 5 minutes or so). This is actually just a guess, and nothing more.

- **You cannot determine individual users**

  Web servers see requests and send results to IP addresses only. There is no way to determine *what* is at that address, only that some request came from it. It could be a real person, it could be some program running on a machine, or it could be lots of people all using the same IP address (more on that below). Some of you will note that the HTTP protocol *does* provide a mechanism for user authentication, where a username and password are required to gain access to a web site or individual pages. And while that is true, it isn't something that a normal, public web site uses (otherwise it wouldn't be public!). As an example, say that one IP address makes a request to your server, and then a minute later, some other IP address makes a request. Can you say how many people visited your site? Again, the answer is **No!**. One of those requests may have come from a search engine 'spider', a program designed to scour the web looking for links and such. Both requests could have been from the same user, but at different addresses. Some analysis program will try to determine the number of users based on things like IP address plus browser type, but even so, these are nothing more than guesses made on some rather faulty assumptions.

- **Network topology makes even IP addresses problematic**

  In the good old days, every machine that wanted to talk on the internet had it's own unique IP address. However, as the internet grew, so did the demand for addresses. As a result, several methods of connecting to the internet were developed to ease the addressing problem. Take, for example, a normal dial-up user sitting at home. They call their service provider, the machines negotiate the connection, and an IP address is assigned from a re-usable 'pool' of IP addresses that have been assigned to the provider. Once the user disconnects, that IP address is made available to other users dialing in. The home user will typically get a different IP address each time they connect, meaning that if for some reason they are disconnected, they will re-connect and get a new IP address. Given this situation, a single user can appear to be at many different IP addresses over a given time. Another typical situation is in a corporate environment, where all the PCs in the organization use private IP addresses to talk on the network, and they connect to the internet through a gateway or firewall machine that translates their private address to the public one the gateway/firewall uses. This can make all the users within the organization appear as if they were all using the same IP address. Proxy servers are similar, where there can be thousands of users, all appearing to come from the same address. Then there are reverse-proxy servers, typical of many large providers such as AOL, that can make a single machine appear to use many different IP addresses while they are connected (the reverse-proxy keeps track of the addresses and translates them back to the user). Given this situation, can you say how many users visited your site if your logs show 10 requests from the same IP address over an hour? Again, the answer is **No!**. It could have been the same user, or it could have been multiple users sitting behind a firewall. Or how about if your logs show 10 requests from 10 different IP addresses? Think it was from 10 different users? Of course not. It could have been 10 different users, could have been a couple of users sitting behind a reverse proxy, could have been one or more users along with a search engine 'spider', or it could be any combination of them all.

But wait, theres more!

Ok, so what have we learned here? Well, in short, you don't know who or what is making requests to your server, and you can't assume that a single IP address is really a single user. Sure, you can make all kinds of assumptions and guesses, but that is all they realy are, and you should not consider them at all accurate. Take the following example; IP address A makes a request to your server, 1 minute later, IP address B makes a request, and then 10 minutes later, address A makes another request. What can we determine from that sequence? Well, we can assume that two users visited. But what if address A was that of a firewall? Those two requests from address A could have been two different users. What if the user at address A got disconnected and dialed back in, getting a different address (address B) and someone else dialed in at the same time and got the now free address A? Or maybe the user was sitting behind a reverse-proxy, and all three requests were really from the same user. And can we tell what 'path' or 'trail' these users took while at the web site or how long they remained? Hopefully, you should now see that the answer to all these things is a big resounding **No!** we can't! Without being able to identify individual unique users, there is no way to tell what an individual unique user does. All is not lost however. Over time, people have come up with ways to get around these limitations. Systems have been written to get around the stateless nature of the HTTP protocol. Cookies and other unique identifiers have been used to track individuals, as has various dynamic pages with back end databases. However, these things are all, for the most part, external to the protocol, not logged in a standard web server log, and require specalized tools to analyze. In all other cases, any programs that claim to analyze these types of metrics should just be considered guesses based on certain assumptions. One such example can be found within the Webalizer itself. The concept of a 'visit' is a metric that cannot be accurately reported, yet that is one of the things that the Webalizer does show. It was added because of the huge number of requests received from individuals using the program. It is based on the assumption that a single IP address represents a single user. You have already seen how this assumption falls flat in the real world, and if you read through the documentation provided with the program, you will see that it clearly says the 'visit' numbers (along with 'entry' and 'exit' pages) are not to be considered accurate, but more of a rough guess. We haven't touched on entry and exit pages yet, but they are based on the concept of a 'visit', which we have already seen isn't accurate. These are supposed to be the first and last page a user sees while at the web site. If a request comes in that is considered a new 'visit', then the URL of that request would be, in theory, the 'Entry' page to the site. Likewise, the last URL requested in a visit would be the 'Exit' page. Similar to user 'paths' or 'trails', and being based on the 'visit' concept, they are to be treated with the same caution. One of the funniest metrics I have seen in one particular analysis program was supposed to tell you where the user was geographically, based on where the domain name of the requesting remote address was registered. Clever idea, but completely worthless. Take for example AOL, which is registered in Virginia. The program considered all AOL users as living in Virginia, which we know is not the case for a provider with access points all over the globe.

Other metrics you CAN determine

Now that you have seen what is possible, you may be thinking that there are some other things these programs display, and wondering about how accurate they might be. Hopefully, based on what you have already seen thus far, you should be able to figure them out on your own. One such metric is that of a 'page' or 'page view'. As we already know, a web page is made up of an HTML text document and usually other elements such as graphic images, audio or other multimedia objects, style sheets, etc.. One request for a web page might generate dozens of requests for these other elements, but a lot of people just want to know how many web pages were requested without counting all the stuff that makes them up. You can get this number, if you know what type of files you may consider a 'page'. In a normal server, these would be just the URLs that end with a .htm or .html extension. Perhaps you have a dynamic site, and your web pages use an .asp, .pl or .php extension instead. You obviously would not want to count .gif or .jpg images as pages, nor would you want to count style sheets, flash graphic and other elements. You could go through the logs and just count up the requests for whatever URL meets your criteria for a 'page', but most analysis programs (including the Webalizer) allows you to specify what you consider a page and will count them up for you.

Other information

Up to now, we have just discussed the CLF (Common Log Format) log format. There are others. The most common is called 'combined', and takes the basic CLF format and adds two new pieces of information. Tacked on the end is the 'user agent' and 'referrer'. A user agent is just the name of the browser or program being used to generate the

request to the web server. The 'referrer' is supposed to be the page that *referred* the user to your web server. Unfortunately, both of these can be completely miseading. The user agent string can be set to anything in some modern browsers. One common trick for Opera users is to set their user agent string to that of MS Internet Explorer so they can view sites that only allow MSIE visitors. And the referrer string, according to the standards document (RFC) for the HTTP protocol, may or may not be used at the browsers choosing, and if used, does not have to be accurate or even informative. The apache web server (which is the most common on the interet) allows other things to be logged, such as cookie information, length of time to handle the request and lots of other stuff. Unfortunately, the inclusion and placement of this information in the server logs are not standard. Another format, developed by the W3C (world wide web consortium), allows log records to be made up of many different pieces of information, and their location can be anywhere in the log entry with a header record needed to map them. Some analysis programs handle these and other formats better than others.

Analysis techniques

The only true way to get an accurate picture of what your web server is doing is to look at it's logs. This is how most of the analysis packages out there get their information, and is the most accurate. Other methods can be used, with different results. One common method, which was widely popular for a while, was the use of a 'page counter'. Basically, it was a dynamic bit included in a web page that incremented a counter and displayed it's value each time the page was requested. Normally, it was included in the page as if it were a standard image file. One problem with this method was that you had to include a different 'image' file for each page you wanted to track. Another problem occured if the remote user had image display turned off in their browser, or could not display images at all (such as in a text based web browser). You could also easily inflate the number by just hitting the 'reload' button on your browser over and over again. Similar methods were developed using java and javascript, in an attempt to get even more information about the visiting browser, such as screen resolution and operating system type. Of course, these can easily be circumvented as well. Some companies set up systems that claim to track your server usage remotely, by including an image or javascript element on your site which would then contact the companies system each time the image or javascript element was requested. These all have the same problems and limitations. In all of these, you can simply turn off images and java/javascript and then browse the web site completely uncounted and unseen (except in the web server logs). Beware of these types of counters and remote usage sites, they are not quite as accurate as they may lead you to believe.

Conclusion

It should now be obvious that there are only certain things you can determine from a web server log. There are some completely accurate numbers you can generate without question. And then, there are some wildly inaccurate and misleading numbers you can garner depending on what assumptions you make. Want to know how many requests generated a 404 (not found) result? Go right ahead and count them up and be completely confident with the number you get. Want to know how many 'users' visited your web site? Good luck with that one.. unless you go 'outside the logs', it will be a hit or miss stab in the dark. But now you should have a good idea of what is and isn't possible, so when you look at your usage report, you will be able to determine what the numbers mean and how much to trust them. You should also now see that a lot can depend on how the program is configured, and that the wrong configuration can lead to wrong reqults. Take the example of 'pages'.. if your analysis software thinks that only URLs with a .htm or .html extension is a page, and all you have is .php pages on your site, that number will be completely wrong. Not because the program is wrong, but because someone told it the wrong information to base its calculations on. Remember, knowlege is power, so now you have the power to ask the proper questions and get the proper results. The next time you look at a server analysis report, hopefully you will see it in a different light given your new found knowlege. For a description of the common terms used by the Webalizer, Click here!

Copyright (C)2002-2010 by Bradford L. Barrett - Last modified: 12-Jun-2008