Tutorial 5 - Extending and analyzing the access log

What are we doing?

We are defining a greatly extended log format in order to better monitor traffic.

Why are we doing this?

In the usual configuration of the Apache web server a log format is used that logs only the most necessary information about access from different clients. In practice, additional information is often required, which can easily be recorded in the server's access log.

Requirements

- An Apache web server, ideally one created using the file structure shown in Tutorial 1 (Compiling an Apache web server).
- Understanding of the minimal configuration in Tutorial 2 (Configuring a minimal Apache server).
- An Apache web server with SSL/TLS support as in Tutorial 4 (Configuring an SSL server)

Step 1: Understanding the common log format

The *common* log format is a very simple format that is hardly ever used any more. It has the advantage of being space-saving and hardly ever writing unnecessary information.

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
...
CustomLog logs/access.log common
```

We use the *LogFormat* directive to define a format and give it a name, *common* in this case.

We invoke this name in the definition of the log file using the *CustomLog* directive. We can use these two directives multiple times in the configuration. Thus, multiple log formats with several name abbreviations can be defined next to one another and log files written in different formats. It's possible for different services to write to separate log files on the same server.

The individual elements of the *common* log format are as follows:

%h designates the *remote host*, normally the IP address of the client making the request. But if the client is behind a proxy server, then we'll see the IP address of the proxy server here. So, if multiple clients share the proxy server then they will have the same *remote host* entry. It's also possible to retranslate the IP addresses using DNS reverse lookup on our server. If we configure this (which is not recommended), then the host name determined for the client would be entered here.

%l represents the *remote log name*. It is usually empty and output as a hyphen ("-"). In fact, this is an attempt to identify the client via *ident* access to the client. This has little client support and results in the biggest performance bottlenecks which is why *%l* is an artifact from the early 1990s.

%u is more commonly used and designates the user name of an authenticated user. The name is set by an authentication module and remains empty (thus the "-"), for as long as access without authentication on the server takes.

%t means the time of access. For big, slow requests the time means the moment the server receives the request line. Since Apache writes a request in the log file only after completing the response, it may occur that a slower request with an earlier time may appear several entries below a short request started later. Up to now this has resulted in confusion when reading the log file.

By default, the time is output between square brackets. It is normally the local time including the deviation from standard time. For example:

```
[25/Nov/2014:08:51:22 +0100]
```

This means November 25, 2014, 8:51 am, 1 hour before standard time. The format of the time can also be changed if necessary. This is done using the *%{format}t* pattern, where *format* follows the specification of *strftime(3)*. We have already made use of this option in Tutorial 2. But let's use an example to take a closer look:

```
%{[%Y-%m-%d %H:%M:%S %z (%s)]}t
```

In this example we put the date in the order *Year-Month-Day*, to make it sortable. And after the deviation from standard time we add the time in seconds since the start of the Unix age in January 1970. This format is more easily read and interpreted via a script.

This example gives us entries using the following pattern:

```
[2014-11-25 09:34:33 +0100 (1322210073)]
```

So much for *%t*. This brings us to *%r* and the request line. This is the first line of the HTTP request as it was sent from the client to the server. Strictly speaking, the request line does not belong in the group of request headers, but it is normally subsumed along with them. In any case, in the request line the client transmits the identification of the resource it is demanding.

Specifically, the line follows this pattern:

```
Method URI Protocol
```

In practice, it's a simple example such as this:

```
GET /index.html HTTP/1.1
```

The *GET* method is being used. This is followed by a space, then the absolute path of the resource on the server. The index file in this case. Optionally, the client can, as we are aware, add a *query string* to the path. This *query string* normally begins with a question mark and comes with a number of parameter value pairs. The *query string* is also output in the log file. Finally, the protocol that is most likely to be HTTP version 1.1. Version 1.0 still continues to be used by some agents (automated scripts). The new HTTP/2 protocol does not appear in the request line of the initial request. In HTTP/2 an update from HTTP/1.1 to HTTP/2 takes place during the request. The start follows the pattern above.

The following format element follows a somewhat different pattern: *%>s*. This means the status of the response, such as *200* for a successfully completed request. The angled bracket indicates that we are interesting in the final status. It may occur that a request is passed off within the server. In this case what we are interested in is not the status that passing it off triggered, but the status of the response for the final internal request.

One typical example would be a request that causes an error on the server (Status 500). But if the associated error page is unavailable, this results in status 404 for the internal transfer. Using the angled bracket means that in this case we want 404 to be written to the log file. If we reverse the direction of the angled bracket, then Status 500 would be logged. Just to be certain, it may be advisable to log both values using the following entry (which is not usual in practice):

```
%<s %>s
```

%b is the last element of the *common* log format. It shows the number of bytes announced in the content-length response headers. In a request for *http://www.example.com/index.html* this value is the size of the *index.html* file. The *response headers* also transmitted are not counted. In addition, this number shows only an announcement of the number and is no guarantee that these data were actually transferred.

Step 2: Understanding the combined log format

The most widespread log format, *combined*, is based on the *common* log format, extending it by two items.

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\"" combined
...
CustomLog logs/access.log combined
```

"%{Referer}i" is used for the referrer. It is output in quotes. The referrer means any resource from which the request that just occurred was originally initiated. This complicated paraphrasing can best be illustrated by an example. If you click a link at a search engine to get to *www.example.com* and once there are automatically redirected to *shop.example.com*, then the log entry for *shop.example.com* will include the search engine as the referrer and not the link to *www.example.com*. If however a CSS file dependent on *shop.example.com* is loaded, the referer would normally be attributed to *shop.example.com*. However, despite all of this, the referrer is part of the client's request. The client is required to follow the protocol and conventions, but can in fact send any kind of information, which is why you cannot rely on headers like these when security is an issue.

Finally, *"%{User-agent}i"* means the client user agent, which is also placed in quotes. This is also a value controlled by the client and which we should not rely on too much. The user agent is the client browser software, normally including the version, the rendering engine, information about compatibility with other browsers and various installed plugins. This results in very long user agent entries which can in some cases include so much information that an individual client can be uniquely identified, because they feature a particular combination of different add-ons of specific versions.

Step 3: Enabling the Logio module

We have become familiar with the *combined* format, the most widespread Apache log format. However, to simplify day-to-day work, the information shown is just not enough. Additional useful information has to be included in the log file.

It is advisable to use the same log format on all servers. Now, instead of just propagating one or two additional values, these instructions describe a comprehensive log format that has proven useful in a variety of scenarios.

However, in order to be able to configure the log format below, we first have to enable the *Logio* module.

If the server has been compiled as described in Tutorial 1, then this module is already present and only has to be added to the list of modules being loaded in the server’s configuration file.

```
LoadModule      logio_module        modules/mod_logio.so
```

We need this module to be able to write two values. *IO In* and *IO Out*. This means the total number of bytes of the HTTP request including header lines and the total number of bytes in the response, also including header lines.

Step 4: Configuring the new, extended log format

We are now ready for a new, comprehensive log format. The format also includes values that the server is as of yet unaware of with the modules defined up to now. It will leave them empty or show them as a hyphen *"-"*. Only with the *Logio* module just enabled this won’t work. The server will crash if we request these values without them being present.

We will gradually be filling in these values in the instructions below. But, as explained above, because it is useful to use the same log format everywhere, we will now be getting a bit ahead of ourselves in the configuration below.

We’ll be starting from the *combined* format and will be extending it to the right. The advantage of this is that the extended log files will continue to be readable in many standard tools, because the additional values are simply ignored. Furthermore, it is very easy to convert the extended log file back into the basic version and then still end up with a *combined* log format.

We define the log format as follows:

```
LogFormat "%h %{GEOIP_COUNTRY_CODE}e %u [%{%Y-%m-%d %H:%M:%S}t.%{usec_frac}t] \"%r\" %>s %b \
\"%{Referer}i\" \"%{User-Agent}i\" %v %A %p %R %{BALANCER_WORKER_ROUTE}e %X \"%{cookie}n\" \
%{UNIQUE_ID}e %{SSL_PROTOCOL}x %{SSL_CIPHER}x %I %O %{ratio}n%% \
%D %{ModSecTimeIn}e %{ApplicationTime}e %{ModSecTimeOut}e \
%{ModSecAnomalyScoreIn}e %{ModSecAnomalyScoreOut}e" extended

...

CustomLog       logs/access.log extended
```

Step 5: Understanding the new, extended log format

The new log format adds 19 values to the access log. This may seem excessive at first glance, but there are in fact good reasons for all of them and having these values available in day-to-day work makes it a lot easier to track down errors.

Let's have a look at the values in order.

In the description of the *common* log format we saw that the second value, the *logname* entry, displays an unused artifact right after the client IP address. We'll replace this item in the log file with the country code for the client IP address. This is useful, because this country code is strongly characteristic of an IP address. (In many cases there is a big difference whether the request originates nationally or from the South Pacific). It is now practical to place it right next to the IP address and have it add more in-depth information to the meaningless number.

After this comes the time format defined in Tutorial 2, which is oriented to the time format of the error log and is now congruent with it. We are also keeping track of microseconds, giving us precise timing information. We are familiar with the next values.

%v refers to the canonical host name of the server that handled the request. If we talk to the server via an alias, the actual name of the server will be written here, not the alias. In a virtual host setup the virtual host server names are also canonical. They will thus also show up here and we can distinguish among them in the log file.

%A is the IP address of the server that received the request. This value helps us to distinguish among servers if multiple log files are combined or multiple servers are writing to the same log file.

%p then describes the port number on which the request was received. This is also important to be able to keep some entries apart if we combine different log files (such as those for port 80 and those for port 443).

%R shows the handler that generated the response to a request. This value may also be empty ("-") if a static file was sent. Or it uses *proxy* to indicate that the request was forwarded to another server.

%{BALANCER_WORKER_ROUTE}e also has to do with forwarding requests. If we alternate among target servers this value represents where the request was sent.

%X shows the status of the TCP connection after the request has been completed. There are three possible values: The connection is closed (-), the connection is being kept open using *Keep-Alive* (+) or the connection was lost before the request could be completed (*X*).

"%{cookie}n" is a value employed by user tracking. This enables us to use a cookie to identify a client and recognize it at a later point in time, provided it still has the cookie. If we set the cookie for the whole domain, e.g. to example.com and not limited to www.example.com, then we are even able to track a client across multiple hosts. Ideally, this would also be possible from the client's IP address, but this may change over the course of a session and multiple clients may be sharing a single IP address.

%{UNIQUE_ID}e is a very helpful value. A unique ID is created on the server for every request. When we output this value on an error page for instance, then a request in the log file can be easily identified using a screenshot, and ideally the entire session can be reproduced on the basis of the user tracking cookies.

Now come two values made available by *mod_ssl*. The encryption module provides the log module values in its own name space, indicated by *x*. The individual values are explained in the *mod_ssl* documentation. For the operation of a server the protocol and encryption used are of primary interest. These two values, referenced by *%{SSL_PROTOCOL}x* and *%{SSL_CIPHER}x* help us get an overview of encryption use. Sooner or later there will come a time when we have to disable the *TLSv1* protocol. But first we want to be certain that is it no longer playing a significant role in practice. The log file will help us do that. It is similar to the encryption algorithm that tells us about the *ciphers* actually being used and helps us make a statement about which ciphers are no longer being used. The information is important. If, for example, vulnerabilities in individual versions of protocols or individual encryption methods become known, then we can assess the effect of our measures by referring to the log file. The following statement in spring 2015 was worth its weight in gold: "Immediately disabling the SSLv3 protocol as a reaction to the POODLE vulnerability will cause an error in approx. 0.8% of requests. Extrapolated to our customer base, xx number of customers will be impacted." Based on these numbers, the risk and the effect of the measures were predictable.

%I and *%O* are used to define the values used by the *Logio* module. It is the total number of bytes in the request and the total number of bytes in the response. We are already familiar with *%b* for the total number of bytes in the response body. *%O* is a bit more precise here and helps us recognize when the request or its response violates size limits.

%{ratio}n%% means the percentage by which the transferred data were able to be compressed by using the *Deflate* module. This is of no concern for the moment, but will provide us interesting performance data in the future.

%D specifies the complete duration of the request in microseconds. Measurement takes place from the time the request line is received until the last part of the response leaves the server.

We'll continue with performance data. In the future we will be using a stopwatch to separately measure the request on its way to the server, onward to the application and while processing the response. The values for this are set in the *ModSecTimeIn*, *ApplicationTime* and *ModSecTimeOut* environment variables.

And, last but not least, there are other values provided to us by *ModSecurity* (to be handled in a subsequent tutorial), specifically the anomaly score of the request and the response. For the moment it's not important to know all of this. What's important is that this highly extended log format gives us a foundation upon which we can build without having to adjust the log format again.

Step 6: Writing other request and response headers to an additional log file

In day-to-day work you are often looking for specific requests or you are unsure of which requests are causing an error. It has often been shown to be useful to have specific additional values written to the log file. Any request and response headers or environment variables can be easily written. Our log format makes extensive use of it.

The \*"%{Referer}i\\"* and \*"%{User-Agent}i\\"* values are request header fields. The balancer route in *%{BALANCER_WORKER_ROUTE}e* is an environment variable. The pattern is clear: *%{Header/Variable}*. Request headers are assigned to the *i* domain. Environment variables to domain *e*, the response headers to domain *o* and the variables of the *SSL* modules to the *x* domain.

So, for debugging purposes we will be writing an additional log file. We will no longer be using the *LogFormat* directive, but instead defining the format together with the file on one line. This is a shortcut, if you want to use a specific format one time only.

```
CustomLog logs/access-debug.log "[%{%Y-%m-%d %H:%M:%S}t.%{usec_frac}t] %{UNIQUE_ID}e \
\"%r\" %{Accept}i %{Content-Type}o"
```

With this additional log file we see the wishes expressed by the client in terms of the content type and what the server actually delivered. Normally this interplay between client and server works very well. But in practice there are sometimes inconsistencies, which is why an additional log file of this kind can be useful for debugging. The result could then look something like this:

```
$> cat logs/access-debug.log
2015-09-02 11:58:35.654011 VebITcCoAwcAADRophsAAAAX "GET / HTTP/1.1" */* text/html
2015-09-02 11:58:37.486603 VebIT8CoAwcAADRophwAAAAX "GET /cms/feed/ HTTP/1.1" text/html,application/xhtml+x
2015-09-02 11:58:39.253209 VebIUMCoAwcAADRoph0AAAAX "GET /cms/2014/04/17/ubuntu-14-04/ HTTP/1.1" */* text/h
2015-09-02 11:58:40.893992 VebIU8CoAwcAADRbdGkAAAAD "GET /cms/2014/05/13/download-softfiles HTTP/1.1" */* t
2015-09-02 11:58:43.558478 VebIVcCoAwcAADRbdGoAAAAD "GET /cms/2014/08/25/netcapture-sshargs HTTP/1.1" */* t
...
```

This is how log files can be very freely defined in Apache. What's more interesting is analyzing the data. But we'll need some data first.

Step 7: Trying it out and filling the log file

Let's configure the extended access log in the *extended* format as described above and work a bit with the server.

We could use *ApacheBench* as described in the second tutorial for this, but that would result in a very uniform log file. We can change things up a bit with the following two one-liners.

```
$> for N in {1..100}; do curl --silent http://localhost/index.html?n=${N}a >/dev/null; done
$> for N in {1..100}; do PAYLOAD=$(uuid -n $N | xargs); \
   curl --silent --data "payload=$PAYLOAD" http://localhost/index.html?n=${N}b >/dev/null; \
   done
```

On the first line we simply make one hundred requests, numbered in the *query string*. Then comes the interesting idea on the second line: We again make one hundred requests. But this time we want to send the data using a POST request in the body of the request. We are dynamically creating this payload in such a way that it gets bigger every time it is called. We use *uuidgen* to generate the data we need. This is a command that generates an *ascii ID*. Stringed together, we get a lot of data. (If there is an error message, this could be because the *uuidgen* command is not present. In this case, the *uuid* package should be installed).

It may take a moment to process this line. As a result we see the following the log file:

```
127.0.0.1 - - [2015-10-03 05:54:09.090117] "GET /index.html?n=1a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.exa
127.0.0.1 - - [2015-10-03 05:54:09.133625] "GET /index.html?n=2a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.exa
127.0.0.1 - - [2015-10-03 05:54:09.179561] "GET /index.html?n=3a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.exa
127.0.0.1 - - [2015-10-03 05:54:09.223015] "GET /index.html?n=4a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.exa
127.0.0.1 - - [2015-10-03 05:54:09.266520] "GET /index.html?n=5a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.exa
127.0.0.1 - - [2015-10-03 05:54:09.310221] "GET /index.html?n=6a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.exa
127.0.0.1 - - [2015-10-03 05:54:09.353847] "GET /index.html?n=7a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.exa
127.0.0.1 - - [2015-10-03 05:54:09.397234] "GET /index.html?n=8a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.exa
127.0.0.1 - - [2015-10-03 05:54:09.440755] "GET /index.html?n=9a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.exa
127.0.0.1 - - [2015-10-03 05:54:09.484324] "GET /index.html?n=10a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:09.527460] "GET /index.html?n=11a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:09.570871] "GET /index.html?n=12a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:09.614222] "GET /index.html?n=13a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:09.657637] "GET /index.html?n=14a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:09.701005] "GET /index.html?n=15a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:09.744447] "GET /index.html?n=16a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:09.787739] "GET /index.html?n=17a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:09.831136] "GET /index.html?n=18a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:09.874456] "GET /index.html?n=19a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:09.917730] "GET /index.html?n=20a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:09.960881] "GET /index.html?n=21a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.004104] "GET /index.html?n=22a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.047408] "GET /index.html?n=23a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.090742] "GET /index.html?n=24a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.133714] "GET /index.html?n=25a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.176825] "GET /index.html?n=26a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.219999] "GET /index.html?n=27a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.263645] "GET /index.html?n=28a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.306908] "GET /index.html?n=29a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.351172] "GET /index.html?n=30a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.397145] "GET /index.html?n=31a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.440458] "GET /index.html?n=32a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.483683] "GET /index.html?n=33a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.529464] "GET /index.html?n=34a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.583115] "GET /index.html?n=35a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.638475] "GET /index.html?n=36a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.683748] "GET /index.html?n=37a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.727064] "GET /index.html?n=38a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.770306] "GET /index.html?n=39a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.813481] "GET /index.html?n=40a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.866573] "GET /index.html?n=41a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.924152] "GET /index.html?n=42a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:10.970115] "GET /index.html?n=43a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.013452] "GET /index.html?n=44a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.057181] "GET /index.html?n=45a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.108020] "GET /index.html?n=46a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.157339] "GET /index.html?n=47a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.210087] "GET /index.html?n=48a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.255414] "GET /index.html?n=49a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.298710] "GET /index.html?n=50a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.342002] "GET /index.html?n=51a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.385796] "GET /index.html?n=52a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.437934] "GET /index.html?n=53a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.484871] "GET /index.html?n=54a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.532164] "GET /index.html?n=55a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.576373] "GET /index.html?n=56a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.625497] "GET /index.html?n=57a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.676021] "GET /index.html?n=58a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.721580] "GET /index.html?n=59a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.771195] "GET /index.html?n=60a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.814475] "GET /index.html?n=61a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.857877] "GET /index.html?n=62a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.901023] "GET /index.html?n=63a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.949860] "GET /index.html?n=64a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:11.996345] "GET /index.html?n=65a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.043010] "GET /index.html?n=66a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.094492] "GET /index.html?n=67a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.139945] "GET /index.html?n=68a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.190450] "GET /index.html?n=69a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.239383] "GET /index.html?n=70a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.282753] "GET /index.html?n=71a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
```

```
127.0.0.1 - - [2015-10-03 05:54:12.327762] "GET /index.html?n=72a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.375769] "GET /index.html?n=73a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.419382] "GET /index.html?n=74a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.463196] "GET /index.html?n=75a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.507089] "GET /index.html?n=76a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.553814] "GET /index.html?n=77a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.597165] "GET /index.html?n=78a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.640322] "GET /index.html?n=79a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.683549] "GET /index.html?n=80a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.726859] "GET /index.html?n=81a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.770189] "GET /index.html?n=82a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.813490] "GET /index.html?n=83a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.856534] "GET /index.html?n=84a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.899494] "GET /index.html?n=85a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.946169] "GET /index.html?n=86a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:12.991259] "GET /index.html?n=87a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:13.036759] "GET /index.html?n=88a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:13.081440] "GET /index.html?n=89a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:13.129467] "GET /index.html?n=90a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:13.183269] "GET /index.html?n=91a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:13.233710] "GET /index.html?n=92a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:13.278141] "GET /index.html?n=93a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:13.325932] "GET /index.html?n=94a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:13.371602] "GET /index.html?n=95a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:13.416067] "GET /index.html?n=96a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:13.467033] "GET /index.html?n=97a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:13.520931] "GET /index.html?n=98a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:13.568819] "GET /index.html?n=99a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:54:13.613138] "GET /index.html?n=100a HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:08.192381] "POST /index.html?n=1b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:55:08.244061] "POST /index.html?n=2b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:55:08.294934] "POST /index.html?n=3b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:55:08.345959] "POST /index.html?n=4b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:55:08.396783] "POST /index.html?n=5b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:55:08.447396] "POST /index.html?n=6b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:55:08.498101] "POST /index.html?n=7b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:55:08.548684] "POST /index.html?n=8b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:55:08.600923] "POST /index.html?n=9b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.ex
127.0.0.1 - - [2015-10-03 05:55:08.651712] "POST /index.html?n=10b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:08.702620] "POST /index.html?n=11b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:08.753260] "POST /index.html?n=12b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:08.803847] "POST /index.html?n=13b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:08.854720] "POST /index.html?n=14b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:08.905325] "POST /index.html?n=15b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:08.956204] "POST /index.html?n=16b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.007565] "POST /index.html?n=17b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.058787] "POST /index.html?n=18b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.109967] "POST /index.html?n=19b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.160955] "POST /index.html?n=20b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.211941] "POST /index.html?n=21b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.263858] "POST /index.html?n=22b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.315355] "POST /index.html?n=23b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.368451] "POST /index.html?n=24b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.422182] "POST /index.html?n=25b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.476593] "POST /index.html?n=26b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.534756] "POST /index.html?n=27b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.588418] "POST /index.html?n=28b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.641792] "POST /index.html?n=29b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.695003] "POST /index.html?n=30b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.747278] "POST /index.html?n=31b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.800173] "POST /index.html?n=32b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.851537] "POST /index.html?n=33b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.903471] "POST /index.html?n=34b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:09.955182] "POST /index.html?n=35b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:10.011663] "POST /index.html?n=36b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:10.063837] "POST /index.html?n=37b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:10.124744] "POST /index.html?n=38b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:10.182238] "POST /index.html?n=39b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:10.233935] "POST /index.html?n=40b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:10.286021] "POST /index.html?n=41b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:10.338986] "POST /index.html?n=42b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:10.392424] "POST /index.html?n=43b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:10.445391] "POST /index.html?n=44b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:10.498816] "POST /index.html?n=45b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
```

```
127.0.0.1 - - [2015-10-03 05:55:10.555547] "POST /index.html?n=46b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:10.607887] "POST /index.html?n=47b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:10.659831] "POST /index.html?n=48b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:10.712089] "POST /index.html?n=49b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:10.764404] "POST /index.html?n=50b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:10.818158] "POST /index.html?n=51b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:10.873327] "POST /index.html?n=52b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:10.927217] "POST /index.html?n=53b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:10.980241] "POST /index.html?n=54b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:11.034181] "POST /index.html?n=55b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:11.089734] "POST /index.html?n=56b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:11.143863] "POST /index.html?n=57b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:11.196211] "POST /index.html?n=58b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:11.249333] "POST /index.html?n=59b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:11.304195] "POST /index.html?n=60b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:11.358419] "POST /index.html?n=61b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:11.413544] "POST /index.html?n=62b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:11.465599] "POST /index.html?n=63b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:11.517771] "POST /index.html?n=64b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:11.569863] "POST /index.html?n=65b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:11.629315] "POST /index.html?n=66b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:11.692200] "POST /index.html?n=67b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:11.744763] "POST /index.html?n=68b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:11.800476] "POST /index.html?n=69b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:11.852595] "POST /index.html?n=70b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:11.904921] "POST /index.html?n=71b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:11.957216] "POST /index.html?n=72b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.010008] "POST /index.html?n=73b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.062213] "POST /index.html?n=74b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.114455] "POST /index.html?n=75b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.168195] "POST /index.html?n=76b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.220147] "POST /index.html?n=77b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.272479] "POST /index.html?n=78b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.327103] "POST /index.html?n=79b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.386224] "POST /index.html?n=80b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.442225] "POST /index.html?n=81b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.496991] "POST /index.html?n=82b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.551043] "POST /index.html?n=83b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.606645] "POST /index.html?n=84b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.659519] "POST /index.html?n=85b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.712401] "POST /index.html?n=86b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.765312] "POST /index.html?n=87b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.817843] "POST /index.html?n=88b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.870316] "POST /index.html?n=89b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.923036] "POST /index.html?n=90b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:12.975815] "POST /index.html?n=91b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:13.028428] "POST /index.html?n=92b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:13.081251] "POST /index.html?n=93b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:13.134076] "POST /index.html?n=94b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:13.189013] "POST /index.html?n=95b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:13.241741] "POST /index.html?n=96b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:13.294453] "POST /index.html?n=97b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:13.347215] "POST /index.html?n=98b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:13.400345] "POST /index.html?n=99b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.e
127.0.0.1 - - [2015-10-03 05:55:13.453047] "POST /index.html?n=100b HTTP/1.1" 200 45 "-" "curl/7.35.0" www.
```

As predicted above, a lot of values are still empty or indicated by -. But we see that we talked to server *www.example.com* on port 443 and that the size of the request increased with every *POST* request, with it being almost 4K, or 4096 bytes, in the end. Simple analyses can already be performed with this simple log file.

Step 8: Performing simple analyses using the extended log format

If you take a close look at the example log file you will see that the duration of the requests are not evenly distributed. There are two outliers. We can identify them as follows:

```
$> egrep -o "\% [0-9]+ " logs/access.log | cut -b3- | tr -d " " | sort -n
```

Using this one-liner we cut out the value that specifies the duration of a request from the log file. We use the percent sign of the Deflate value as an anchor for a simple regular expression and take the number following it. *egrep* makes sense here, because we want to work with regex, the *-o* option results in only the match itself being output, not the entire line. This is very helpful. One detail that will help us to avoid errors in the future is the space following the plus sign. It only accepts values that have a space following the number. The problem is the user agent that also appears in our log format and which has up to now also included percent signs. We assume here that percent signs can be followed by a space and a whole number. But this is not followed by another space and this combination only appears at the end of the log file line after the *Deflate space savings* percent sign. We then use *cut* so that only the third and subsequent characters are output and finally we use *tr* to separate the closing space (see regex). We are then ready for numerical sorting. This delivers the following result:

```
...
925
932
935
939
958
1048
1093
1393
3937
```

In our example there are four values with a duration of over 1,000 microseconds, or more than one millisecond, three of which are within reason, but the one with 4 milliseconds is clearly a statistical outlier, clearly setting itself apart from the other values.

We know that we made 100 GET and 100 POST requests. But for the sake of practice, let's count them again:

```
$> egrep -c "\"GET " logs/access.log
```

This should result in 100 GET requests:

```
100
```

We can also compare GET and POST with one another. We do this as follows:

```
$> egrep  -o '"(GET|POST)' logs/access.log | cut -b2- | sort | uniq -c
```

Here, we filter out the GET and the POST requests using the method that follows a quote mark. We then cut out the quote mark, sort and count grouped:

```
    100 GET
    100 POST
```

So much for these first finger exercises. On the basis of this self-filled log file this is unfortunately not yet very exciting. So let's try it with a real log file from a production server.

Step 9: Greater in-depth analysis of an example log file

Analyses using a real log file from a production server are much more exciting. Here's one with 10,000 requests:

tutorial-5-example-access.log

```
$> head tutorial-5-example-access.log
75.249.65.145 US - [2015-09-02 10:42:51.003372] "GET /cms/tina-access-editor-for-download/ HTTP/1.1" 200 71
71.180.228.107 US - [2015-09-02 11:14:02.800605] "GET /cms/application_3_applikationsserver_aufsetzen/?q=ap
5.45.105.71 DE - [2015-09-02 11:32:39.371240] "GET /cms/feed/ HTTP/1.1" 200 35422 "-" "Tiny Tiny RSS/1.15.3
155.80.44.115 IT - [2015-09-02 11:58:35.654011] "GET /robots.txt HTTP/1.0" 404 21023 "-" "Mozilla/5.0 (comp
155.80.44.115 IT - [2015-09-02 11:58:37.486603] "GET /cms/2013/09/23/ HTTP/1.1" 200 22822 "-" "Mozilla/5.0 
155.80.44.115 IT - [2015-09-02 11:58:39.253209] "GET /cms/2013/09/23/convert-from-splashid-safe-to-keepass-
155.80.44.115 IT - [2015-09-02 11:58:40.893992] "GET /cms/2013/09/23/convert-from-splashid-safe-to-keepass-
155.80.44.115 IT - [2015-09-02 11:58:43.558478] "GET /cms/2013/10/21/ HTTP/1.1" 200 6171 "-" "Mozilla/5.0 (
155.80.44.115 IT - [2015-09-02 11:58:45.287565] "GET /cms/2013/10/21/nftables-to-replace-iptables-firewall-
155.80.44.115 IT - [2015-09-02 11:58:49.801640] "GET /cms/2013/10/21/nftables-to-replace-iptables-firewall-
```

Let’s have a look at the distribution of *GET* and *POST* requests here:

```
$> cat tutorial-5-example-access.log  | egrep  -o '"(GET|POST)'  | cut -b2- | sort | uniq -c
   9781 GET
     12 POST
```

This is a clear result. Do we actually see many errors? Or requests answered with an HTTP error code?

```
$> cat tutorial-5-example-access.log | cut -d\" -f3 | cut -d\  -f2 | sort | uniq -c
   9040 200
      5 206
    447 301
     47 304
     16 400
      3 403
    401 404
     41 408
```

Besides the sixteen requests with the “400 Bad Request” HTTP response there is a large number of 404s (“404 Not Found”). HTTP status 400 means a protocol error. As is commonly known, 404 is a page not found. This is where we should have a look at the permissions. But before we continue, a note about the request using the *cut* command. We have subdivided the log line using the *”*-delimiter, extracted the third field with this subdivision and then further subdivided the content, but this time with a space (note the \ character) as the delimiter and extracted the second field, which is now the status. Afterwards this was sorted and the *uniq function* used in count mode. We will be seeing that this type of access to the data is a pattern that repeats itself. Let’s take a closer look at the log file.

Further above we discussed encryption protocols and how their analyses was a foundation for deciding on an appropirate reaction to the *POODLE* vulnerability. In practice, which encryption protocols are actually on the server since then:

```
$> cat tutorial-5-example-access.log | cut -d\" -f9 | cut -d\  -f3 | sort | uniq -c | sort -n
     21 -
     65 TLSv1.1
   1764 TLSv1
   8150 TLSv1.2
```

It appears that Apache is not always recording an encryption protocol. This is a bit strange, but because it is a very rare case, we won’t be pursuing it for the moment. What’s more important are the numerical ratios between the TLS protocols. After disabling *SSLv3*, the *TLSv1.2* protocol is dominant, in addition to a substantial percentage of *TLSv1.0*. *TLSv1.1* can be disregarded.

We again got to the desired result by a series of *cut* commands. It would actually be advisable to take note of these commands, because will be needing them again and again. It would then be an alias list as follows:

```
alias alip='cut -d\  -f1'
alias alcountry='cut -d\  -f2'
alias aluser='cut -d\  -f3'
alias altimestamp='cut -d\  -f4,5 | tr -d "[]"'
alias alrequestline='cut -d\" -f2'
alias almethod='cut -d\" -f2 | cut -d\  -f1 | sed "s/^-$/**NONE**/"'
alias aluri='cut -d\" -f2 | cut -d\  -f2 | sed "s/^-$/**NONE**/"'
alias alprotocol='cut -d\" -f2 | cut -d\  -f3 | sed "s/^-$/**NONE**/"'
alias alstatus='cut -d\" -f3 | cut -d\  -f2'
alias alresponsebodysize='cut -d\" -f3 | cut -d\  -f3'
alias alreferer='cut -d\" -f4 | sed "s/^-$/**NONE**/"'
alias alreferrer='cut -d\" -f4 | sed "s/^-$/**NONE**/"'
alias aluseragent='cut -d\" -f6 | sed "s/^-$/**NONE**/"'
alias alservername='cut -d\" -f7 | cut -d\  -f2'
alias alservername='cut -d\" -f7 | cut -d\  -f2'
alias allocalip='cut -d\" -f7 | cut -d\  -f3'
alias alcanonicalport='cut -d\" -f7 | cut -d\  -f4'
alias alport='cut -d\" -f7 | cut -d\  -f4'
alias alhandler='cut -d\" -f7 | cut -d\  -f5'
alias albalroute='cut -d\" -f7 | cut -d\  -f6'
alias alconnstatus='cut -d\" -f7 | cut -d\  -f7'
alias altrkcookie='cut -d\" -f8'
alias alreqid='cut -d\" -f9 | cut -d\  -f2'
alias alsslprotocol='cut -d\" -f9 | cut -d\  -f3'
alias alsslcipher='cut -d\" -f9 | cut -d\  -f4'
alias alioin='cut -d\" -f9 | cut -d\  -f5'
alias alioout='cut -d\" -f9 | cut -d\  -f6'
alias aldeflateratio='cut -d\" -f9 | cut -d\  -f7 | tr -d %'
alias alduration='cut -d\" -f9 | cut -d\  -f8'
alias aldurationin='cut -d\" -f9 | cut -d\  -f9'
alias aldurationapp='cut -d\" -f9 | cut -d\  -f10'
alias aldurationout='cut -d\" -f9 | cut -d\  -f11'
alias alscorein='cut -d\" -f9 | cut -d\  -f12 | tr "-" "0"'
alias alscoreout='cut -d\" -f9 | cut -d\  -f13 | tr "-" "0"'
alias alscores='cut -d\" -f9 | cut -d\  -f12,13 | tr " " ";" | tr "-" "0"'
```

All of the aliases begin with *al*. This stands for *ApacheLog* or *AccessLog*. This is followed by the field name. The individual aliases are not sorted alphabetically. They instead follow the sequence of the fields in the format of the log file.

This list with alias definitions is available in the file apache-modsec.alias. They have been put together there with a few additional aliases that we will be defining in subsequent tutorials. If you often work with Apache and its log files, then it is advisable to place these alias definitions in the home directory and to load them when logging in. By using the following entry in the *.bashrc* file or via another related mechanism.

```
test -e ~/.apache-modsec.alias && . ~/.apache-modsec.alias
```

Let’s use the new alias right away:

```
$> cat tutorial-5-example-access.log | alsslprotocol | sort | uniq -c | sort -n
     21 -
     65 TLSv1.1
   1764 TLSv1
   8150 TLSv1.2
```

This is a bit easier. But the repeated typing of *sort* followed by *uniq -c* and then a numerical *sort* yet again is tiresome. Because it is again a repeating pattern, an alias is also worthwhile here, which can be abbreviated to *sucs*: a merger of the beginning letters and the *c* from *uniq -c*.

```
$> alias sucs='sort | uniq -c | sort -n'
```

This then enables us to do the following:

```
$> cat tutorial-5-example-access.log | alsslprotocol | sucs
     21 -
     65 TLSv1.1
   1764 TLSv1
   8150 TLSv1.2
```

This is now a simple command that is easy to remember and easy to write. We now have a look at the numerical ratio of 1764 to 8150. We have a total of exactly 10,000 requests; the percentage values can be derived by looking at it. In practice however log files may not counted so easily, we will thus be needing help calculating the percentages.

Step 10: Analyses using percentages and simple statistics

What we are lacking is a command that works similar to the *sucs* alias, but converts the number values into percentages in the same pass: *sucspercent*.

```
$> alias sucspercent='sort | uniq -c | sort -n | $HOME/bin/percent.awk'
```

Traditionally, *awk* is used for quick calculations in Linux. In addition to the above linked *alias* file, which also includes the *sucspercent*, the *awk* script *percent.awk* is also available. It is ideally placed in the *bin* directory of your home directory. The *sucspercent* alias above then assumes this setup. The *awk* script is available here.

```
$> cat tutorial-5-example-access.log | alsslprotocol | sucspercent
```

```
                         Entry        Count Percent
---------------------------------------------------
                             -           21   0.21%
                       TLSv1.1           65   0.65%
                         TLSv1         1764  17.64%
                       TLSv1.2         8150  81.50%
---------------------------------------------------
                         Total        10000 100.00%
```

Wonderful. We are now able to output the numerical ratios for any repeating values. How does it look, for example, with the encryption method used?

```
$> cat tutorial-5-example-access.log | alsslcipher | sucspercent
                         Entry        Count Percent
---------------------------------------------------
          DHE-RSA-AES256-SHA256           2   0.02%
         ECDHE-RSA-DES-CBC3-SHA           5   0.05%
                   DES-CBC3-SHA           8   0.08%
                             -           21   0.21%
      DHE-RSA-AES256-GCM-SHA384          43   0.43%
        ECDHE-RSA-AES128-SHA256          86   0.86%
           ECDHE-RSA-AES128-SHA         102   1.02%
             DHE-RSA-AES256-SHA         169   1.69%
                     AES256-SHA         565   5.65%
        ECDHE-RSA-AES256-SHA384         919   9.19%
           ECDHE-RSA-AES256-SHA        1008  10.08%
    ECDHE-RSA-AES256-GCM-SHA384        1176  11.76%
    ECDHE-RSA-AES128-GCM-SHA256        5896  58.96%
---------------------------------------------------
                         Total        10000 100.00%
```

A good overview on the fly. We can be satisfied with this for the moment. Is there anything to say about the HTTP protocol versions?

```
$> cat tutorial-5-example-access.log | alprotocol | sucspercent
                         Entry        Count Percent
---------------------------------------------------
                          quit            4   0.04%
                      **NONE**           41   0.41%
                      HTTP/1.0           70   0.70%
                      HTTP/1.1         9885  98.85%
---------------------------------------------------
                         Total        10000 100.00%
```

The obsolete *HTTP/1.0* still appears, and something seems to have gone wrong with 45 requests. In the calculation let’s concentrate on the successful requests with a valid protocol and have another look at the percentages:

```
$> cat tutorial-5-example-access.log | alprotocol | grep HTTP |  sucspercent
                         Entry        Count Percent
---------------------------------------------------
                      HTTP/1.0           70   0.70%
                      HTTP/1.1         9885  99.30%
---------------------------------------------------
                         Total         9955 100.00%
```

An additional *grep* is used here. We can narrow down the "alias field extraction -> sucs” pattern via additional filter operations.

With the different aliases for the extraction of values from the log file and the two *sucs* and *sucspercent* aliases we have come up with a handy tool enabling us to simply answer questions about the relative frequency of repeating values using the same pattern of commands.

For measurements that no longer repeat, such as the duration of a request or the size of the response, these percentages are not very useful. What we need is a simple statistical analysis. What are needed are the average, perhaps the median, information about the outliers and, for logical reasons, the standard deviation.

Such a script is also available for download: basicstats.awk. Similar to percent.awk, it is advisable to place this script in your private *bin* directory. It’s important to know that this script consists of an expanded *awk* implementation (yes, there are several). The package is normally named *gawk* and it makes sure that the `awk` command uses the Gnu awk implementation.

```
$> cat tutorial-5-example-access.log | alioout | basicstats.awk
Num of values:        10000
      Average:        15375
       Median:         6646
          Min:            0
          Max:       340179
        Range:       340179
Std deviation:        25913
```

These numbers give a clear picture of the service. With an average response size of 15 KB and a median of 6.6 KB we have a typical web service. Specifically, the median means that half of the responses were smaller than 6.6 KB. The largest response came in at 340 KB, the standard deviation of just under 26 KB means that the large values were less frequent overall.

How does the duration of the requests look? Do we have a similar homogenous picture?

```
$> cat tutorial-5-example-access.log | alduration | basicstats.awk
Num of values:        10000
      Average:        91306
       Median:         2431
          Min:           18
          Max:    301455050
        Range:    301455032
Std deviation:      3023884
```

It’s important to remember that we are dealing in microseconds here. The median was 2400 microseconds, which is just over 2 milliseconds. At 91 milliseconds, the average is much larger. We obviously have a lot of outliers which have pushed up the average. In fact, we have a maximum value of 301 seconds and less surprisingly a standard deviation of 3 seconds. The picture is thus less homogenous and we have at least some requests that should be investigated. But this is now getting a bit more complicated. The suggested method is only one of many possible and is included here as a suggestion and inspiration for further work with the log file:

```
$> cat tutorial-5-example-access.log | grep "\"GET " | aluri | cut -d\/ -f1,2,3 | sort | uniq \
| while read P; do AVG=$(grep "GET $P" tutorial-5-example-access.log | alduration | basicstats.awk \
| grep Average | sed 's/.*: //'); echo "$AVG $P"; done \
| sort -n
...
       97459 /cms/
       97840 /cms/application-download-soft
       98959 /cms/category
      109910 /cms/technical-blog
      115564 /cms/content
      146096 /cms/feed
      146881 /files/application-9-sshots-appl.png
      860889 /cms/download-softfiles
```

What happens here in order? We use *grep* to filter *GET* requests. We extract the *URI* and use *cut* to cut it. We are only interested in the first part of the path. We limit ourselves here in order to get a reasonable grouping, because too many different paths will add little value. The path list we get is then sorted alphabetically and reduced by using *uniq*. This is half the work.

We now sequentially place the paths into variable *P* and use *while* to make a loop. In the loop we calculate the basic statistics for the path saved in *P* and filter the output for the average. In doing so, we use *sed* to filter in such a way that the *AVG* variable includes only a number and not the *Average* name itself. We now output this average value and the path names. End of the loop. Last, but not least, we sort everything numerically and get an overview of which paths resulted in requests with longer response times. A path named */cms/download-softfiles* apparently comes out on top. The keyword *download* makes this appear plausible.

This brings us to the end of this tutorial. The goal was to introduce an expanded log format and to demonstrate working with the log files. In doing so, repeatedly used were a series of aliases and two *awk* scripts, which can be very flexibly arranged in sequence. With these tools and the necessary experience in their handling you will be able to quickly get at the information available in the log files.

References

- Apache Module mod_log_config documentation
- Apache Module mod_ssl documentation
- tutorial-5-example-access.log
- .apache-modsec.alias
- percent.awk
- basicstats.awk

License / Copying / Further use