3

Logs and Monitoring

Introduction to Logging in Apache

In addition to the error logging functionality described in the previous chapter, Apache provides extensive facilities for recording information about every aspect of a request. This chapter covers the most common issues found when logging requests, such as conditional logging, log rotation, resolution of IP addresses, and piped logging. It also covers a number of bundled and third-party modules and utilities for monitoring the status of your Apache server and to analyze its logs.

Default Apache Log Files

Apache provides a number of monitoring and logging facilities to track the correct operation of the server. The default Apache configuration provides two log

files, placed inside the `logs` directory of the installation directory:

- The `access_log` file (`access.log` in Windows) contains information about the requests that have been served by the server, such as the URL requested, the IP address of the client, and whether the request completed successfully or not.
- The `error_log` file (`error.log` in Windows) contains information related to error conditions, as well as different events in the lifecycle of the server.

Creating Log Formats

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%h %l %u %t \"%r\" %>s %b"
    \"%{Referer}i\" \"%{User-agent}i\"" combined
```

The `LogFormat` directive allows you to tell Apache which aspects of the request you want to record. You will still need additional directives to tell Apache where to log that information, but that is addressed in the next section. This example shows the configuration for the two most popular formats, the Common Log Format and the Combined Log Format. When Apache receives a request, it will substitute each one of the fields prefixed by a % with the corresponding request attribute. If you are using the CLF, each entry in your log file will look like this:

```
192.168.200.4 - someuser [12/Jun/2005:08:33:34
    +0500] "GET /example.png HTTP/1.0" 200 1234
```

If you are using the combined common format, each entry in your log file will look like this:

```
192.168.200.4 - someuser [12/Jun/2005:08:33:34
     +0500] "GET /example.png HTTP/1.0" 200 1234
     http://www.example.com/index.html "Mozilla/5.0
     (Windows; U; Windows NT 5.1; en-US; rv:1.7.7)"
```

Although the appendix provides a comprehensive logging format reference, this list describes the most important fields:

- `%h`: The IP address of the client that sent the request to the web server, or the client's hostname if you have `HostNameLookups` enabled (192.168.200.4 in this example.)
- `%u`: The user id of the user who sent the request determined by HTTP authentication (`someuser` in the example). See Chapter 6 for more details on how to configure HTTP-based authentication.
- `%t`: Time when the request was received by the server.
- `%r`: Text of the original request line from the client including the HTTP method used, the resource requested, and the HTTP protocol version used by the client's browser (`"GET /example.png HTTP/1.0"` in the example).
- `%>s`: The final HTTP request status code that the web server sends back to the client (200 in the example, indicating that the request was completed successfully).
- `%b`: The size in bytes of the object sent to the client in response to the request excluding the response headers (1234 in the example).

The combined log format extends the common log format with two additional fields. It is defined as

- `%{Referer}i`: The `Referer` HTTP request header; that is, the web page that referred to the current document (`http://www.example.com/index.html` in the example).
- `%{User-agent}i`: The `User-agent` HTTP request header. It includes information about the client's browser (`"Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.7)"` in the example).

Creating a Custom Log File

```
CustomLog logs/access_log common
TransferLog logs/sample.log
```

You may want to create new log files in addition to the ones included with Apache. This example uses `CustomLog` to create a new log file and store the information defined by a previously defined log format named `common`, as seen in the previous section. You can replace the nickname with the format definition itself. An additional, simpler directive is `Transferlog`, which will just take the definition provided by the latest `LogFormat` directive.

Redirecting Logs to an External Program

```
TransferLog "|bin/rotatelogs /var/logs/apachelog
86400"
```

You can also use `CustomLog` or `TransferLog` to redirect ("pipe") the log output to an external program instead of a file. To do this, you need to begin with the pipe character "|", followed by the path to a program that will receive the log information on its standard input. This example uses the `rotatelogs` program included with Apache, which is described in a later section.

When an external program is used, it will be run as the user who started `httpd`. This will be root if the server was started by root; be absolutely sure that the program is secure. Also, when entering a file path on non-Unix platforms, care should be taken to make sure that only forward slashes are used, even though the platform may allow the use of backslashes. In general, it is a good idea to always use forward slashes throughout the configuration files.

Logging Requests Conditionally

```
SetEnvIf Request_URI "(\.gif|\.jpg)$" image
CustomLog logs/access_log common env=!image

SetEnvIf Remote_Addr  192\.168\.200\.5 specialma-
chine
CustomLog logs/special_access_log common env=spe-
cialmachine
```

You can decide whether or not to log a request based on the presence of an environment variable. This variable can be previously set based on a number of parameters, such as the client's IP address or the presence of a certain header in the request. As shown in this example, the `CustomLog` directive can accept an environment variable as a third argument. If the environment variable is present, the entry will be logged;

otherwise, it will not. If the environment variable is negated by prefixing it with an "!", the entry will be logged if the variable is *not* present. The example shows you how to avoid logging images in GIF and JPEG format and how to log requests from a particular IP address to a separate log file. See the next section for another example.

Monitoring Who Is Linking to Your Website

```
SetEnvIfNoCase Referer www\.example\.com internalre-
ferral
LogFormat "%{Referer}i -> %U" referer
CustomLog logs/referer.log referer env=!internalre-
ferral
```

In order to monitor who is linking to your website, you can log the `Referer:` header from the request. This header contains the URL that linked to the page being requested. While not always present or accurate, it works for the majority of cases. This example shows how to use an environment variable to log the referrer information to a separate file. In this particular case, we are only interested in logging external referers, not those that come from an internal web page. To do so, in this example we check whether the referrer matches our own domain.

Monitoring Apache with mod_status

```
<Location /server-status>
  SetHandler server-status
  Order Deny,Allow
  Deny from all
  Allow from 192.168.0
</Location>
```

The `mod_status` module provides information on server activity and performance, allowing a server administrator to find out how well their server is performing. An HTML page is presented that gives the current server statistics in an easily readable form, such as the number of workers serving requests, the number of idle workers, the time the server was started/restarted, and so on.

If you include an `ExtendedStatus On` directive, additional information will be displayed, such as individual information about each worker status, total number of accesses, current requests being processed, and so on.

Bear in mind that, depending on your server load, this extended statistics recording may have a significant impact on server performance.

This example shows how to enable the `mod_status` monitoring, while restricting access to this information to only certain IP addresses. You can now access server statistics by using a Web browser to access the page at http://www.example.com/server-status.

Monitoring Apache with SNMP

There are a couple of open-source modules that add Simple Network Management Protocol (SNMP) capabilities to the Apache web server. This protocol is commonly used to manage network servers and equipment from a central console such as HP OpenView and Tivoli. With this module, you can easily monitor Apache performance in real time, including server uptime, load average, number of errors in a certain period of time, number of bytes and requests served, and many other metrics. The SNMP modules can also generate alarms when a certain threshold or error condition is met, such as a sudden increase in the number of simultaneous client connections.

For Apache 1.3, you can use `mod_snmp`, which can be found at http://www.mod-snmp.com/ and supports SNMP version 1 and 2. It requires patching of the Apache core.

For Apache 2, you can use a similar module called `mod_apache_snmp`. It can be found at http://mod-apache-snmp.sourceforge.net/. This module supports versions 1, 2, and 3 of the SNMP protocol and can be compiled as a DSO, without the need to patch Apache.

A number of open-source tools and frameworks allow you to manage SNMP resources, such as the tools at http://www.net-snmp.org, OpenNMS (http://www.opennms.org), and Nagios (http://www.nagios.org).

Analyzing Your Logs with Open-source Tools

There are a number of commercial and open-source tools that you can use to process and display your log data. They usually take a log file, analyze its contents, and create a series of web pages with the relevant statistics.

The following are some popular, freely available, open source applications for general log analysis:

- Webalizer—http://www.mrunix.net/webalizer/
- AWStats—http://awstats.sf.net

Other tools allow you more advanced log processing, such as visually displaying the path followed by your visitors:

- Visitors—http://www.hping.org/visitors/
- Pathalizer—http://pathalizer.bzzt.net/

Monitoring Your Logs in Real Time

In addition to `mod_status` and the various SNMP modules described earlier, you can use the `apachetop` command-line tool, which can be downloaded from http://clueful.shagged.org/apachetop/.

This tool works similarly to the Unix `top` command-line tool, but instead of displaying the status of the operating system, it displays the status of the web server in real time.

If you run Apache on a Unix system and you have a website with low traffic, you can use the `tail` command-line utility to rudimentarily monitor, in real time, log entries both to your access and error logs:

```
tail -f logfile
```

There are additional programs that enable you to quickly identify problems by scanning your error log files for specific errors, malformed requests, and so on, and reporting on them:

- Logscan can be found at http://www.garand.net/security.php
- ScanErrLog can be found at http://www.librelogiciel.com/software/

Logging Requests to a Database

Apache itself does not include tools for logging to databases, but a few third-party scripts and modules are available:

- `mod_log_sql` allows you to log requests directly to a MySQL database: http://www.outoforder.cc/projects/apache/mod_log_sql/
- You can then query the database using the Apache LogView SQL tool: http://freshmeat.net/projects/apachelogviewsql/
- `pglogd` collects logs and stores log entries in a PostgreSQL database: http://www.digitalstratum.com/pglogd/.

Rotating and Archiving Logs

```
CustomLog "|bin/rotatelogs /var/logs/apachelog
86400" common
```

If you have a website with high traffic, your log files will quickly grow in size. While you can always archive the log files by hand, there are a number of mechanisms to rotate logs periodically, archiving and compressing older logs at well-defined intervals.

To avoid having to stop or restart the server when manipulating the log files, a common solution is to use an intermediate program to log the requests. The program will in turn take care of rotating, compressing, and archiving the logs.

Apache provides the `rotatelogs` tool for this purpose. You can find a similar, alternative program at http://cronolog.org/.

This example uses the `rotatelogs` tool to create a new log file and move the current log to the `/var/logs` directory daily (86400 is the number of seconds in one day). Check the Apache documentation for details on how to use `rotatelogs` to also rotate logs based on size and name archived files based on a template.

CAUTION: If the path to the log rotation program includes spaces, you might need to escape them by prefixing them with a \ (backslash). This is especially common in the Windows platform.

Controlling IP Address Resolution

```
HostNameLookups on
```

If you set the `HostNameLookups` directive to `on` then Apache will try to determine (resolve) the hostname corresponding to the client's IP-address when it logs the request.

With `HostNameLookups` set to `off`, an `access_log` entry may look like

```
192.168.200.4 - someuser [12/Jun/2005:08:33:34
     +0500] "GET /example.png HTTP/1.0" 200 1234
```

And with `HostNameLookups` set to `on`, the same entry would look like

```
unit12.example.com - someuser [12/Jun/2005:08:33:34
     +0500] "GET /example.png HTTP/1.0" 200 1234
```

The next section explains the reverse process, how to replace IP addresses in logs with hostnames.

Processing Logged IP Addresses

```
$ logresolve < access_log > resolved_log
```

Setting `HostNameLookups` to `on` can have an impact on the server's performance, slowing its response time. To avoid using this directive setting, it is possible to disable name resolution and use a separate post-processing utility that can scan the log files and replace the IP addresses with host names. These tools are more efficient because they can cache results and they do not cause any delay when serving requests to clients.

Apache includes one such tool, `logresolve` (`logresolve.exe` in Windows). It reads log entries from standard input and outputs the result to its standard output. To read to and from a file, you can use redirection, on both Unix and Windows, as shown in the example.

You should bear in mind that the result of an IP address resolution result will not always correspond to the real hostname that sent the request. For example, if there is a proxy or gateway between the client and the web server, the IP address reported by `HostNameLookups` or `logresolve` will be the IP address of the proxy or gateway and you will get the hostname of the proxy server or the IP block managed by the gateway, rather than the name of an actual host.

Restarting Apache Automatically If It Fails

```
#!/bin/bash
if [ `ps -waux | grep -v grep | grep -c httpd` -lt 1
]; then apachectl restart; fi
```

If you install Apache on Windows as a service, it can be automatically restarted by the service manager if it crashes.

In Unix, you can implement this functionality with a watchdog script. A watchdog monitors the status of another program, and if the program crashes or stops for any reason, the watchdog starts it again. The example shows a simple Linux script that will monitor the system process list to ensure that an `httpd` process exists, and will restart httpd if it crashes. To use it, you

will need to give it executable permissions and add it to your `cron` configuration so it can be run at predefined intervals.

If you are running Solaris, use `ps -ef` instead of `ps -waux`.

You can find a more sophisticated watchdog script that will send email when the server is down, and can watch specific `httpd` process ids, at the following URL: http://perl.apache.org/docs/general/control/control.html.

Most Linux distributions also include their own generic watchdog scripts that can be adapted to work with Apache.

Merging and Splitting Log Files

When you have a cluster of web servers serving the same content, it is often necessary to merge logs from all servers into a unique log file before passing it to analysis tools. Similarly, if a single Apache server handles several virtual hosts, sometimes it is necessary to split a single log file into different files, one per each virtual host. This can be done at the web server level, as explained in the next section, or by post-processing the log file. Both Apache 1.3 and 2.x come with a support script file named `split-logfile`. It can be found in the `support/` directory of the Apache source distribution.

The logtool project provides a collection of log manipulation tools, and can be found at http://www.coker.com.au/logtools/.

The vlogger tool allows splitting a single log stream into several virtualhost-specific log files, as well as being able to replace tools such as cronolog, as explained in a previous section. It can be found at http://n0rp.chemlab.org/vlogger/.

Keeping Separate Logs for Each Virtual Host

```
<VirtualHost 192.168.200.3>
ServerName vhost1.example.com
CustomLog logs/vhost1.example.com_log combined
ErrorLog logs/vhost2.example.com_log
.......
</Virtual Host>
```

You can keep separate access logs for each virtual host using a CustomLog directive inside each <VirtualHost> section, as shown in the example.

You can also choose to log the operations of all virtual hosts in the access_log defined in the global server context:

```
LogFormat "%v %h %l %u %t \"%r\" %>s %b" common_vir-
tualhost
CustomLog logs/access_log common_virtualhost
```

%v will log the name of the virtual host that serves the request. You can then use the tools described in the previous section to process the resulting log file. This may be necessary if you have a large number of virtual hosts.

If you don't want to keep track of the operations of a particular host at all, you can simply use

```
CustomLog /dev/null
```

Common Log Entries

In addition to the information provided in Chapter 2, this section describes a number of log entries for some common errors that you may find when you review your log files. You can safely ignore most of them.

File favicon.ico Not Found

Most recent web browsers support displaying a custom icon next to the browser's location bar or when storing a bookmark. To do so, the browser requests a specific file from the website (`favicon.ico`). If this file is not present, you will get this error. You can learn more about how to provide this icon in your website in Chapter 1.

File robots.txt Not Found

The robots.txt file is a file requested by certain programs, such as automatic downloaders and web crawlers, when accessing your website. These are programs that scan websites, recursively following and downloading any links that they find. They are usually associated with search engines, and their main purpose is to store and index the retrieved contents. If the `robots.txt` file is not present, you will get this error.

httpd.pid Overwritten

On Unix systems, the `httpd.pid` file contains the PID (process id) for the Apache process currently running. It is created when Apache starts and deleted when it shuts down. When Apache does not have a clean shutdown, for example because the server had to be killed manually or the machine crashed, the file will not be

deleted. In this case, it will still be present the next time the server starts, giving this error.

Long, Strange Requests

You may find strange requests such as the following in your error log:

```
"SEARCH /\x90\x02\xb1\x02\xb1\x02\xb1\x02 ..."
"GET /scripts/..%252f../winnt/system32/cmd.exe?/
c+dir HTTP/1.0..."
"GET /default.ida?NNNNNNN NNNNNNNNNNNNNNNNN ..."
```

Or requests for executable files that do not exist in your website, such as `cmd.exe`, `root.exe`, dir, and so on.

There are log entries that result from automated attempts to exploit vulnerabilities in web servers. Luckily, most of them are generated by worms or other malicious applications specific to Microsoft Internet Information Server on Windows, and Apache is not affected. However, from time to time, flaws are discovered in Apache that could leave it vulnerable to remote attacks. For this reason, you should always keep your Apache server up to date, as described in Chapter 6.

This page intentionally left blank