Getting Tooled Up:

Showing Robots the Door

Do you have concerns about robots crawling your site? Are they overloading your web server? Do you know what they are indexing? Can you control them? **Ian Peacock** describes the Robots Exclusion Protocol and reports on a analysis of the use of this protocol by UK Universities and Colleges. This article appears in the Web, and not the print, version of Ariadne.

What is Robots Exclusion Protocol?

The robot exclusion protocol (REP) is a method implemented on web servers to control access to server resources for robots that crawl the web. Ultimately, it is up to the designer or user of robot-software to decide whether or not these protocols will be respected. However, the criteria defining an *ethical* robot includes stipulation that a robot should support REP.

This article refers to the established REP [1] acredited to Martijn Koster [2]. This involves creating a *server-wide* set of directives contained within the top-level `/robots.txt` plain-text file (e.g. corresponding to `http://my.server.foo-domain/robots.txt`). The currently deployed protocol allows multiple `Disallow` fields, one per-line, to be followed by a directory path. Robots parsing a `/robots.txt` file will not retrieve any resource with a URL path below the path specified by the `Disallow` directive. A `Disallow` field without a value is interpreted to mean no restrictions. Groups of `Disallow` directives must be associated with a particular `User-agent` (corresponding to the HTTP `User-agent` request header, which a robot should use to identify itself). This is done by inserting a `User-agent` field above the directives associated with it. The values for the `User-agent` field are allowed to be a particular user-agent (e.g. `RogueRobot/v3.0`), a list of user-agents or '`*`' which specifies all robots. Figure 1 gives an example of a `/robots.txt` file.

```
# This is an example robots.txt file for the site
# http://my.site.ac.uk/
# Note that you can insert comments by preceeding them with a hash

# and that blank lines are also valid.

User-agent: *                     # All user-agents (except others
                                    specified in this file)
Disallow: /cgi-bin/               # Dont look at stuff in
                                    http://my.site.ac.uk/cgi-bin/

User-agent: WebWatch/v3.0
Disallow:                         # The WebWatch robot is allowed to look
                                    at everything :-)

User-agent: BadRobot1, BadRobot2
Disallow: /                       # These BadRobots are denied access to
                                    everything

User-agent: IndexingRobot
Disallow: /cgi-bin/               # Binaries are no good for indexing
Disallow: /images/                # Images are no good for indexing
Disallow: /home-pages/            # Privacy issues with home-pages??
Disallow: /site/admin/stats/webstats # Web stats are no good for indexing
                                    (they may also be
                                  # sensitive)
```

Figure 1 - An example `robots.txt` file

UK Universities and Colleges `/robots.txt` files

The WebWatch project has recently undertaken an analysis of the `/robots.txt` files of UK Higher Education Institutions (HEIs) main web sites, as defined by the list of institutional web services [3] maintained by NISS. From a list of 163 institutional web servers, 53 `/robots.txt` files were retrieved by a WebWatch robot. This is around 33% of the servers we looked at; the remaining servers did not have a `/robots.txt` (i.e. the server returned a 404 response code) or the connection timed-out.

The robot wrote each `robots.txt` file to disk for subsequent analysis. This was achieved with two Perl scripts, one to produce analysis information on the file, and another to perform basic error-checking.

The first script output records containing the following information:

- File-size in bytes
- File-size as total number of lines
- Number of lines starting with a comment
- Number of lines containing a comment
- Number of `Disallow` directives corresponding to each `User-agent`
- Number of `Allow` directives corresponding to each `User-agent`
- Total number of `User-agent` fields

where a line break was defined as `NL`. The `Allow` directive is not strictly part of the currently deployed protocol but was checked for.

The error-checking script scans for common errors. This has been subsequently been made into a web-based service for server administrators to check their own `/robots.txt` files.

Analysis of UK Universities and Colleges `/robots.txt` files

Size of file

The mean size of a `/robots.txt` files is around 427 bytes. This corresponds to a mean total number of lines of about 15. Figure 2 shows the distribution of the total number of lines of text in a `/robots.txt` file.

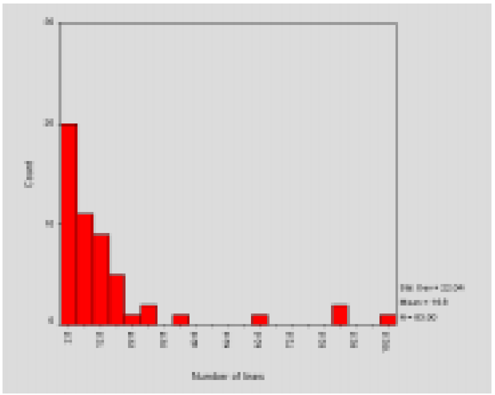

Figure 2 - Distribution of total number of lines in a sample of `/robots.txt` files.

The distribution of size in bytes is roughly the same shape as Figure 2. The two measurements of size are approximately proportional, the average number of bytes per line being about 28.

Figure 2 indicates that the majority of files contain less than 22 lines, with the outliers each representing one or two sites containing more. The large intervals between these corresponds to a visual inspection that the "average" `robots.txt` contains "typical" lines such as `Disallow: /cgi-bin` and `Disallow: /images` and a small number of sites list large (site-specific) lists of directories. It will be interesting to monitor the shape of this distribution as sites tailor `robots.txt` files to reflect their own web site.

The range of total number of lines is from 0 lines (an empty `robots.txt`, of which there were 2 cases) to 101 lines.

Stripping the `/robots.txt` file of comments and blank lines and comparing this to the original, indicated that a large number were similar - i.e. many files contained few comments or blank lines. For those files that contained no comments and no blank lines, over 80% contained less than 6 lines in total. There were no cases of files containing only comments.

On average, 21% of a `/robots.txt` file is composed of non-parsed lines. Further analysis indicates that this corresponds to an average of approximately 1 blank line and 2 comment lines per file. The distribution of the total number of non-parsed lines is of roughly the same shape as the size distribution of the `/robots.txt` file. This suggests that comments and blank lines grow in rough proportion to the total number of lines contained in a file.

Use of `User-agent`

The mean number of `User-agent` fields included in a `/robots.txt` file is just over 1. There were no cases of multiple user-agents referred to by a single `User-agent` field. The distribution of number of `User-agent` fields per file is spread over 0, 1, 2, 3 and 7 occurrences. Those `/robots.txt` files with 0 occurrences are syntactically incorrect since each file must contain at least one reference to a `User-agent`.

Use of directives

The mean number of directives per line is around 9. These were all `Disallow` directives - no `Allow` directives were found. Figure 3 shows a frequency distribution of the number of directives found per file.

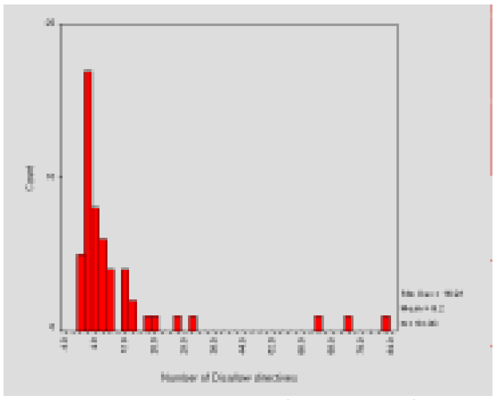

Figure 3 - Distribution of number of directives per `/robots.txt` file

Figure 3 shows that most sites are using less than 12 directives per file and that the most frequent number of directives is actually two. This is due to the large number of "standard" `/robots.txt` files which `Disallow` a couple of "standard" locations (e.g. `/cgi-bin` and `/images`). Note the logical correlation between the outliers in Figure 3 and in Figure 2 - the larger files contain more directives. There were 4 cases of 0 directives. These correspond to the zero-length files, and two invalid `/robots.txt` files.

Calculated from the above approximate means, the number of directives per `User-agent` is approximately 9. Further analysis shows this is closer to 8 and the distribution is shown in Figure 4.

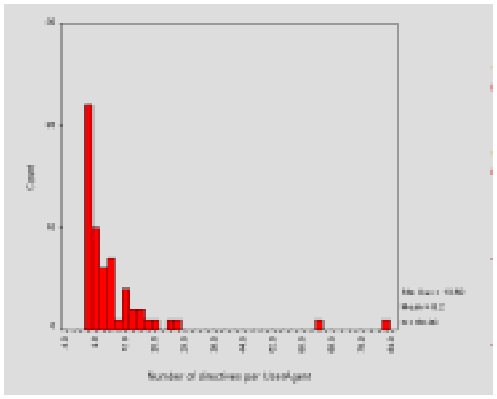

Figure 4 - Distribution of number of directives per `User-agent` field

Note that in Figure 4 compared to Figure 3, some outliers have shifted left. This implies that the shifted outlier sites had their `/robots.txt` file organised into more than one `User-agent` field. The static outliers contain only one `User-agent` field with many directives, showing that the site administrators dont recognise individual robots. This is wise unless the administrator can keep up with additions and changes to the current pool of robots.

Error checking

The second script performed some simple error checking on each of the retrieved `/robots.txt` files. The common errors are shown in Figure 5.

Errors	**No. Occurences**
`User-agent` field without value	5
No User-agent value corresponding to `Disallow`	5
No `Disallow` fields corresponding to `User-agent`	2
Warnings	
Unknown fields	4
`CR` DOS-style end of line	4
Empty file	2
Optimise	
Multiple `User-agent` fields refering to the same value	1

Figure 5 - Errors encountered in `/robots.txt` files

The three errors shown here are strictly illegal, each non-zero-length file must contain at least one `User-agent` field with a corresponding value and at least one `Disallow` directive. The latter error was triggered by the files that also triggered the "Unknown field" warning mentioned below. Interestingly, there were no cases of a file without an *attempt* at inserting a `Disallow` directive (apart from those of zero-length, which *is* valid).

The unknown fields warnings refer to a field that is not one of `User-agent`, `Disallow` or `Allow`. Closer examination of these warnings reveals that two sites put spaces or tab stops at the start of a line, before otherwise valid fields. The remaining cases failed to postfix valid fieldnames with a colon. DOS end of lines are valid, but are mentioned because some UNIX robots may have problems with this.

The optimisation remark refers to a file which uses multiple `User-agent` fields referring to the same user-agent. All directives referring to the same user-agent can be inserted under the one field.

Conclusions

Almost all of the 'mainstream' robots (i.e. those run by the major search-engines) and many other ethical robots respect REP. This means that having a site `/robots.txt` file will control access to your server for many robotic visitors. It is recommended that sites implement REP, if possible, in order to exercise some degree of control over server accesses and corresponding server load. Implementation will also aid the production of useful index-spaces on the web and cut-down on the proportion of 'spam' that is indexed. There are also benefits for the users of robots. Site administrators may direct indexing robots away from irrelevant material and point out 'black-holes' and other stumbling blocks for robots. As REP becomes ever-more widespread, the number of robots implementing the standard will probably increase.

It should be borne in mind that REP relies in the cooperation of a (possibly unethical) robot user. More reliable exclusion can be enforced via HTTP authentication or lower-level access controls.

The open nature of the `/robots.txt` file means that it should not contain confidential information (directly or indirectly). Disallowing a robot to index material stored in certain directories should *not* be an indication that the material contained within is 'secret' or sensitive. The protocol is not a security mechanism. In cases where material must be protected from indexing, password-protection should be used. For information not-requiring particularly fascist protection, it is worth remembering that a URL not-linked anywhere on the site (or other sites) will not be stumbled upon by a robot. Also within HTML forms, submit buttons are rarely followed by robots.

The design of a `/robots.txt` file should direct task-specific robots away from areas that are irrelevant to their work. For example, a hyperlink maintenance robot needs access to the whole site, except perhaps 'black-holes' and CGI scripts (most of the time you should `Disallow` indexing of scripts). An indexing robot, on the other hand, needs access only to relevant indexable documents (i.e. you should also `Disallow` things like images). Our observations show that there tends to be a 'standard' `/robots.txt` file similar to that shown in Figure 6.

```
User-agent: *
Disallow: /cgi-bin/
Disallow: /images/
```

Figure 6 - A typical `robots.txt` file

This file is fine, though does not address the characteristics of the web-space served. There is almost certainly other material which is unsuitable for indexing, for example, collections of web-logs.

There have been some reports of very large `/robots.txt` files causing errors when parsed by some robots.

One disadvantage of the `/robots.txt` method is that it is server-wide and should be maintained by an individual on behalf of the servers information-providers. Note that it is *not* valid in terms of the protocol to have a `/robots.txt` file in a subdirectory of the root ('/') directory, although employing this technique may be a useful strategy in maintaining a cross-departmental (or similar) exclusion file, perhaps with a script collecting all of these and forming the top level file.

A recent, less widely supported exclusion protocol [4] overcomes the problem mentioned above, but is restricted in other ways. The method involves directives embedded within HTML documents and allows the page author to specify whether the page should be indexed and/or followed (parsed for hyperlinks or links to inline objects). This method is implemented with the HTML `META` element using the `NAME="ROBOTS"` attribute-value pair. The `CONTENT` attribute of the `META` element then includes a list of non-conflicting directives that should be implemented by a robot. The possibilities are `INDEX` or `NOINDEX` and `FOLLOW` or `NOFOLLOW`. Alternatively, the convenience keywords `ALL=` or `NONE=` may be used to preceed a list of directives that should all be set on or off respectively.

Example 1

```
<META NAME="ROBOTS" CONTENT="NOINDEX,FOLLOW">
```

This document should not be indexed, but should be parsed for links.

Example 2

```
<META NAME="ROBOTS" CONTENT="ALL=INDEX,FOLLOW">
```

This document should be indexed and parsed for links.

The current `/robots.txt` exclusion protocol is currently being revised and written as an internet draft [5]. This draft clarifies a number of points originating from the previous defining document and gives a more detailed description of the relationship between robot and `/robots.txt` file. From the server administrators point of view, the new directive `Allow` is added. Our above analysis would indicate the lack of `Allow` directives to imply that this revision has not yet been widely adopted. It is not a recommendation to do - the draft is, at present, uncompleted.

The error-checking script used in the above analysis has been turned into a WebWatch service so that site-administrators can check for common errors in their own `/robots.txt` files. The service runs as a CGI script at <URL: http://www.ukoln.ac.uk/web-focus/webwatch/services/robots-txt/>. An example session is shown in Figure 7.

We hope to continue monitoring the use of `/robots.txt` files as part of the WebWatch project.

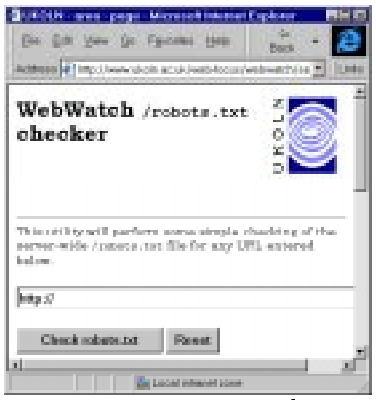
WebWatch /robots.txt checker
UKOLN
This utility will perform some simple checking of the server-wide /robots.txt file for any URL entered below.
http://
Check robots.txt
Reset

Figure 7 - WebWatch `robots.txt` checking service

References

1. Koster, M:A Standard for Robot Exclusion,
   <URL: http://info.webcrawler.com/mak/projects/robots/norobots.html>
2. Koster, M: m.koster@webcrawler.com,
   <URL: mailto:m.koster@webcrawler.com>
3. NISS-maintained List of UK HE Campus Information Services, <URL: http://www.niss.ac.uk/education/hesites/cwis.html>
4. HTML Author's Guide to the Robots META tag,
   <URL: http://info.webcrawler.com/mak/projects/robots/meta-user.html>
5. Koster, M: Internet Draft specification of robots.txt,
   <URL: http://info.webcrawler.com/mak/projects/robots/norobots-rfc.html>

This article was originally published in the web version of Ariadne, issue 15.
See <URL: `http://www.ariadne.ac.uk/issue15/robots/`>