A Tangled Web of Deceit

Andy Whitcroft
Support Group
Computer Science Department
City University

Tim Wilkinson
Systems Architecture Research Centre
City University

Abstract

When good web clients first became available we decided that the Web provided enormous potential for the dissemination of information both within our department and throughout the university. We therefore decided to join the Web. At that time the only server available was a very primitive version of the CERN HTTPD, so we decided to create our own. This server was the basis for our current server which has provided the backbone for the CityCS Web for well over a year.

Despite the potential offered by the Web, we encountered much resistance from the author base. This was due to the lack of good authoring tools and the difficulties involved in presenting the information through the server. Also, the more information we added and the more authors we encouraged, the greater our workloads became. At that time we embarked or a project to find reasons for, and solutions to the problems encountered. We have found that only through flexibility provided by Deceit have we, as maintainers, been able to shift responsibility for information onto its author, thus reducing our workload and increasing the range of information available.

1 Introduction

The Web provides enormous potential for the dissemination of information both within an organisation and throughout the world. It also provides a consistent interface to almost all local information services. However, much of this potential has yet to be realised as authors have been unwilling or unable to provide access to their information. The unwillingness of authors is difficult to overcome, although exposure to the Web and the advantages it can bring helps to sway opinion. The inability of authors to provide information is a direct result of a lack of good authoring tools and the lack of a suitable mechanism for information dissemination. If we are going to have the level of author participation we need in order for information to be kept up to date and relevant then it is essential that the author can directly introduce and maintain information at the local Web node. To this end we embarked on two projects, one to design an effective HTML editor[1] and the second to create a Web server which provided the necessary user configurability required. The latter is described here.

Deceit is an HTTP/1.0[2] compliant http server written entirely in Perl. It is used to serve the web in the Computer Science Department of City University, UK. Deceit has one basic philosophy: total devolution of control. This means that information providers can create and manage their own areas of the Web. Additionally, any user is able to add dynamic extensions without administrative intervention, such as generated pages and search engines. These might include usage graphs or WAIS based indexing. These are used at City to provide services such as 'fingering' users, as well as dictionary, thesaurus and bibliography lookups.

In this paper we will discuss the problems associated with serving information via the web, both from the web maintainer's and the author's point of view. The design and implementation of the server is described, including the user authentication and access control mechanisms. We present an overview of a number of other freely available web servers highlighting the features and limitations of each. We then examine our solution and compare it with the others examined.

Since the introduction of deceit as our web server, its greater flexibility and simplicity has encouraged authors to become directly involved in the provision and maintenance of their information. Additionally, we have had a number of local users and research groups providing information about themselves, their groups and their research activities.

2 The Problem

Our original server, in common with many others, imposes severe restrictions. From an author's point of view it is difficult to gain access to the document hierarchy as access had to be authorised by the web maintainer. Even after access is achieved, there is a constant need to contact the web maintainers to change protection or add document generators as these were configured centrally. Control of access to documents is very primitive, only allowing control at the host level. This prevented the use of this system with any private or sensitive information.

From the web administrators point of view the constant need to maintain configuration information on behalf of authors made what should have been an almost self maintaining system into an administrative nightmare. The problem was compounded by the tendency for users to find the system to be more trouble than it was worth such that the responsibility for maintaining information defaulted to the maintainer, and this quickly led to information becoming out of date.

Although the server was fast and provided a powerful environment it suffered from a number of significant limitations. These centred on the authors' ability to use the server easily to provide information. Firstly, all information accessible through the server was located in a single directory hierarchy on the server. This meant that it was only possible to place information in the web if you were allocated an area in the server hierarchy and had access to the server machine. Secondly, all server configuration was located in a central configuration file. This configuration file provided access control information as well as specifying all search engines and document generators. This meant that all user requests to add document generators or to change protection over their information had to be done by the web maintainer. Finally, the server had no mechanism for linking or moving information within the document hierarchy. This meant that the namespace very quickly become full of badly named directories which could not be moved.

Most of the problems stemmed from a lack of flexibility. These factors conspired to prevent potential authors from utilising the system anywhere near its potential.

3 Our Solution: Deceit

Based on our experiences we decided that in order to improve author participation, much greater flexibility was required from the information server. The other freely available servers seemed to suffer from similar limitations to our initial server, and so we embarked on a project to create a new Web server which provided the necessary flexibility and configurability.

We had a number of specific aims in mind when designing the new server. Firstly, it had to be flexible, we needed to ensure that users were able to take a significant role in the authoring and maintenance of their information. Secondly, authentication was vital so that sensitive or restricted information could be included within the same framework. Thirdly, the server had to be easily extensible; we had quickly found ourselves unable to add new features cleanly to the old implementation. Finally, the server had to be secure, we did not wish to reduce the overall system security any further by its use.

3.1 Extensibility

Deceit is extensible in two ways: by 'handlers', which allow the addition of a new methods of response generation; and by 'libraries', which allow common code libraries to be incorporated. Handlers and libraries are loaded by the server at boot-time and also during a server restart (which causes a re-read of the server configuration).

Documents within the document hierarchy may have a number of different 'source forms', which represent possible starting points from which a valid response may be made. The handlers (described below) provide alternative mappings between these source forms and the final responses. The first reasonable mapping between an existing source form and an acceptable response is used. Such a mechanism makes it possible to add new source forms or new response types simply by adding additional handlers.

3.1.1 Source Forms

We wished to represent document source forms within the document hierarchy in a type independent manner. If we look at the sample request below we can see that the request consists of a number of parts. The document to be operated on, '/guide/index', the method to apply, 'GET', and the acceptable response types, 'text/html'.

```
GET /guide/index HTTP/1.0
Accept: text/html
```

The name of the document does not imply the form in which it is stored nor its type. We are therefore at liberty to use any suitable form to produce an acceptable response.

As the name of a document does not need to specify its type we do not require an extension on the document name. However, as HTTP/0.9 clients use this to determine the return type of the document, we strip any supplied extension from the name of the document. Any extension is used to augment the 'Accept:' field of the request for HTTP/0.9 clients. The extension is still specified in local anchors.

Source forms are represented in the document hierarchy using the name of the document plus a source form extension. For example, the document above may exist in both 'text/html' and 'text/plain' formats. These would be represented by sources 'index.html' and 'index.txt'. The mappings between the source types and source extensions is managed by the handlers and the central configuration file.

3.1.2 The Document Hierarchy

Having identified the document references, the first problem is locating the document in the local file-systems. The Deceit document hierarchy is based on a collection of volumes. Each volume is similar to a filesystem mount-point, that is it indicates where a particular section of the hierarchy may be found in the local filesystems. For example, the following volume map shows that '/' may be found in the directory '/usr/local/web' and '/research' in '/usr/research/web'.

```
volumes {
    /               /usr/local/web;
    /research       /usr/research/web;
}
```

Volumes are matched in a largest leftmost match order; this means that it is possible to mount multiple overlapping areas of the filesystem without problems. For example, '/research/index' would be found in '/usr/research/web' rather than '/usr/local/web'.

Having determined the volume from the volume map, the directory hierarchy is then examined to find the deepest matching directory to the request. Having located the directory in which the document should exist, we attempt to locate the object using the remainder of the request.

For example, let us examine '/research/guide/index.html'. Firstly, this is, mapped using the volumes map, to '/usr/research/web/guide/index.html'. Next we search for the deepest existing directory, which in this case would be '/usr/research/web/guide'. The document name is then stripped and we end up with a directory location of '/usr/research/web/guide' and a document of 'index'.

3.1.3 Handlers

Having located the document referred to, the server must now attempt to produce a response in an acceptable format as specified in the 'Accept:' and 'Accept-Encoding:' headers. Deceit 'handlers' provide the mechanisms for converting source forms into a response. The server achieves this by applying the handlers to the document until one is able to produce a response. If none is able to do so then the server rejects the request as 'Not Found'. Once a handler has been selected successfully, the HTTP response headers are generated and the handler is invoked to generate the body of the response message.

These handlers provide most of the basic functionality of the server. This means that as functional requirements change, different versions or additional handlers can be loaded. All standard handlers are optional, it is possible to configure a server with no handlers. However, that server will only ever return 'Not Found' to all requests. Deceit comes with a number of standard handlers which provide access control, ismap support, link support as well as search and generator interfaces.

Deceit has the following standard handlers:

acl Access control handler, providing object protection based on hosts, methods, user agents, users and groups (including Basic password authorisation). Additionally it provides conditional linkage and redirect facilities.

generate Generator handler, provides user level generation for methods, including searches and forms.

ismap Ismap handler, providing image map facilities, including matching of circles and boxes, linking to local and off-server documents.

link Link handler, providing local and off-server document linkage.

file File handler, returning plain documents and images based on required and available document source types.

Handlers are composed of two basic parts: the validator, which permits the examination of the request context thus allowing the handler to decide what if any response it is able to make; and the generator which actually produces the response. All active handlers are loaded into the server context at server boot-time. A typical handler would take the document requested and attempt to find one or more source forms which it is capable of rendering. If none was present or none can be rendered in an acceptable response type the handler indicates 'no interest' in that document. If one was found then preparations are made to generate the response, including specification of the return type, its size, modification date etc. Following generation of an appropriate header by the server, dependent to the protocol in use, the handler is called to generate the body of the response from the source form.

3.1.4 Libraries

Document generators and handlers often provide similar functionality. For example, many document generators provide searching facilities on simple databases, also most produce HTML documents as their output. To allow code reuse, Deceit provides the concept of loadable code libraries. These are available to all handler code and all user document generators. A number of basic libraries are supplied:

html The HTML page layout library, providing routines to format anchors, headers, footers, etc.

post The posting parser, which provides routines to parse the result of POST methods and forms.

srch The searching support library, which provides routines to parse and search a simple text based database and produce ordered matches.

Libraries are loaded at server boot and restart time and configured centrally.

3.2 Flexibility

The document source form interface allows all information about a document to be stored in the document hierarchy. Access control information, document generators, search engines as well as alternative formats of a document all reside together in the same directory. The available forms are designated by their source form extensions. By having all information together in this manner, authors are able to keep control over their documents.

3.2.1 Search Engines and Generators

Search engines and generated documents are particularly important if we are to make full use of the potential of the Web. Deceit provides both through the same interface, the document generator. Deceit document generators reside in the document space much as a standard document. Generators are of two basic types: method generators, which are only applied for a specific method; and general generators, which are applied for all methods. They are represented by the source forms, '.xxx' which represents the method specific generator for the method 'xxx' and '.gen' which represents a general generator. The method specific generator is used in preference to the general generator.

Document generators are similar to handlers and are composed of two basic parts: the validator, which permits the examination of the request context thus allowing the generator to decide what, if any, response it is able to make; and the generator which actually produces the body of the response. Document generators are implemented in PERL, which allows them to be directly loaded into the server context. Generators are not 'Common Gateway Interface' (CGI) conformant although they are very similar; this stems from the need to be backwards compatible with generators from older versions of the server. It is our intention to provide an additional CGI compatible interface through a CGI handler. Due to the similarity between our local generator interface and the CGI definition we have been able to convert a number of CGI gateways for use with Deceit, notably we have used the 'wwwwais' interface to provide keyword searching of our local node.

Below is an example of a document generator. This generator produces a simple document containing the current time on the server host when the page is requested:

```
#
# Validate
#
sub Validate {
```

```
        local(*info) = @_;

        $info{type} = 'text/html';
        &main'HOK; #' We are able to render.
    }

    #
    # Generate.
    #
    sub Generate {
        local(*info) = @_;

        print "<head><title>Server Date</title></head>";
        print "<body>\n";
        print "Current server time: " . time . "\n";
        print "</body>\n";

        &main'HOK;
    }
```

As we can see from this, the generator is split into two parts: the routine 'Validate', which sets the return type to 'text/html' and indicates that this generator is able to render a response; and the routine 'Generate', which produces a very simple HTML document containing the time. This server could be installed as 'time.GET' in the root document hierarchy, thus it would be invoked when a client requested a GET of '/time' or '/time.html'.

We have used document generators to provide a number of information services locally. These include the 'wwwwais' interface mentioned previously and a simple finger interface allowing location of local users.

3.2.2 Links

In order to allow simpler maintenance of the document hierarchy, we need to be able to move information within the document hierarchy as the information base grows. Deceit provides these via the source form '.lnk'. These contain linkage information indicating the type and destination of the link.

Links are of two types. The first is a local link and allows a local document to be referenced under a different name. If the link source 'local.lnk' in the research hierarchy contained the following example then any reference to the document '/research/local.html' would apply to the document '/research/index'. It is possible to reference any local document in this manner. However, as the client is responsible for the calculation of relative links within documents it is important to exercise care using these 'hidden' links.

```
link index.html
```

The second form is a remote link and allows a local document to reference any other document, either local or remote. If the link source 'remote.lnk' in the research hierarchy contained the following example then any reference to the document '/research/remote.lnk' would cause a client redirect to be returned, thus causing the client to retrieve the referenced document.

```
redirect /research/index.html
```

These kinds of links are used to allow centrally named documents to reside in local user directories.

3.3 Authentication and Protection

Protection of documents and authentication of clients is essential if we are to be able to place secret or licenced information within the web. Also, authors may wish to render different documents based on 'who' is accessing a document. For example, they may wish to only include specific sections of a document if the client is local. Deceit provides a number of features to allow authors to restrict access to documents and to indicate what is presented to different users or sites. The access control mechanism specifies who may access a document, what methods are acceptable on that document and which version of a document a particular request should use.

The access control mechanism allows a number of criteria to be taken into account when examining a request:

host The host from which the request was received, either the hostname or its IP address for machines on the Internet. This is determined from the connection between the client and the server.

method The method specified in the client request.

user The user from which the request came.

group The group of which the user may be a member.

client The client in use by the user.

These criteria may be combined using logical operators (AND, OR and NOT) to make more complex selections. The 'user' and 'group' specifiers are currently only available when using the 'Basic' authentication scheme.

Access control is provided on a per directory basis. Access control information is located in the file '.acl' and resides either in the directory in which the document is located or in one of its parent directories. This file defines access information via a set of predicates. For example, the access control file shown below defines the protection for both the '/guide' hierarchy as well as for local documents.

```
/guide/*  (
            (host *.cs.city.ac.uk) |
            (basic authors authors.pwd &amp; group admin);
               methods GET HEAD POST
          )
          (host *.gov; fail)
*         (methods GET HEAD)
```

The above access control file defines three basic things:

- users from the domain 'cs.city.ac.uk' have GET, HEAD and POST access to the guide hierarchy.
- users who have provided a valid username and password in the 'authors' realm and are members of the 'admin' group as specified in authors password file have GET, HEAD and POST access to the guide hierarchy.
- users from the 'gov' domain have no access to the guide hierarchy.
- Otherwise (indicated by the '*') GET and HEAD access is permitted.

The example shows the use of the 'Basic' authentication scheme. Under this scheme users are required to supply a username and password with their request. These are verified in a local password file, 'authors.pwd' in this example. The password file provides a list of valid username and password pairings. Additionally, it provides the real name of the user and any groupings of which they are a member. For example, the following password file indicates three users, andy, tim and njw.

```
andy:password:Andy Whitcroft:admin
tim:password:Tim Wilkinson:admin
njw:password:Nick Williams:general
```

This password file specifies that andy and tim are both in the admin group and that njw is in the general group.

3.4 Security

It is very important that as we add flexibility we do not reduce the security of the system on which it is run. Also, user ignorance should not make them vulnerable. We looked at security from the point of view that the system should be no easier to compromise with the server than without. For example, we do not believe that it is our responsibility to prevent a user from making the system password file available via the server, as if they have an account on the system they could just as easily post it to the Usenet News.

Obviously the provision of user level generators causes a number of security problems. It is very important that if we are to execute document generators on behalf of users that we do not allow their id to be used if there is any chance that the generator has been modified by another user. At the same time we may wish to be able to maintain a directory by members of a local group and so we need to be able to execute generators on behalf of that group not any one member.

Deceit uses the user and group ownership on a directory to check read permissions on source forms and access control information. These are only used if a directory is deemed secure. If the directory is writable by the group or world then the user id 'nobody' is substituted, and if the directory is world writable then the group is also substituted with the group id 'nobody'. In this way only a file which is 'secure' from tampering will be executed using a user's id.

When attempting to execute generators, a similar process is applied. Only if the generator source is protected from group and world modification is the generator executed as the user, if the generator is protected from world modification only it will be executed as the group of the source. Hence we allow users or groups to own generators and have them executed on their behalf.

4 Other Servers

Before we examine the suitability of our solution to the requirements of authors and web administrators we will examine a selection of other servers, outlining their strengths and weaknesses.

4.1 CERN HTTPD

The CERN HTTPD[3] server is written in C and is available for UNIX, VMS and VM/CMS. This server has a number of interesting features worth noting.

Multiple source forms This allows storage of multiple versions of a document, in different formats or languages, with the most appropriate being selected based on client requests.

Name mapping This allows names to be mapped within the server so that document names can persist as documents move physically.

Authorisation Documents may be protected based on the hosts, users and groups. These are determined using the 'Basic' authorisation scheme.

Document Generators Both their own 'Pre-Parsed' and CGI/1.0 format document generators are supported.

Proxy server The server is also capable of acting as a proxy server for use on a firewall.

Document Cache The server can provide a remote document cache through the proxy mechanism.

The CERN server has a number of desirable features, especially the proxy cache service. However, a number of the most desirable features to an author are centrally configured. Specifically, the authorisation scheme and the document generators are specified in this manner. Also, there does not seem to be any mechanism for extending the server externally.

4.2 Plexus

Plexus[4] is written entirely in PERL. Most functionality is added through configurable PERL functions called 'gateways' which are loaded under the control of a central configuration file. The server boasts the following features:

Extensible New functionality can be added to the server via the central configuration.

Document Generators Its own form of document generators are supported in the form of loadable 'gateways' which are included at server boot-time.

Authentication Document protection is provided via a script which is executed for each document.

Plexus provided a great deal of potential for extensibility through the use of loadable scripts. These 'gateways' provide document generators for particular documents, however, they are centrally configured and loaded into the server at boot-time, thus users are unable to add personal generators. Document generators are expected to produce the entire document including all headers and response codes, routines are provided to support this process, for example, 'MIME_header(...)'. The protection and access control mechanism is very flexible and based on the execution of a script. However, the supplied script only provides very simple access control. Although Plexus provides a flexible and extensible starting point, it lacks the necessary functionality to provide the level of author flexibility or protection required.

4.3 NCSA HTTPD-1.1

The NCSA HTTPD[5] (version 1.1) is written in C and is available for the UNIX platform. It provides a combined document hierarchy, supports personal user directories, provides authentication and document generators. The following features are of particular interest:

Server Scripts These provide generated documents, searches and form support.

Security Directory based authentication including support for 'Basic' authentication.

Personal directories Users may have their own files stored in their own areas.

Server side includes It is possible to specify other documents or command output in your documents.

The NCSA server has a number of desirable features, notably the server scripts and the directory based authentication. However, a number of the features are centrally configured. This applies particularly to document generators which are aliased into a centrally maintained binary directory. Also, there does not seem to be any mechanism for extending the server externally and the methods supported seem to be fixed.

5 Evaluating Deceit

Deceit has now been used locally for about two months. We have taken the opportunities offered by the introduction of the new server to review and revise the current information base and to improve its presentation. Currently, most of the old information is now hosted by the new server. We believe that we have achieve much to reduce the problems we encountered in the past.

5.1 Extensibility

Through the provision of handlers, libraries and generators we can now extend the server easily. For example, we have a prototype cache generator for use with clients supporting proxy - this was written in an afternoon. Also we do not restrict the range of methods understood by the server. Thus, if new methods are introduced they will be instantly usable in handlers and generators.

5.2 Flexibility

Through document generators and also the provision of local control of authentication we achieve almost totally localised control over document hierarchies. Also, through the use of document 'link's we have been able to maintain the document hierarchy more easily.

5.3 Authentication and Protection

With authentication provided at the document level we are able to offer different protection and authentication schemes over each document. Also through the provision of 'basic' authentication support we can provide controlled access to sensitive information.

5.4 Security

Through controlled use of user and group identities we have managed to obtain flexibility without reducing security. Also we have tried to make it difficult for users to compromise their own security accidentally.

6 Conclusion

Since the introduction of the new server we have been approached by a number of members of the university interested in placing information both about themselves and about their work into the web. Additionally, we have been able to persuade people to take over information which we had originally placed into the web and to take responsibility for keeping it up to date. A number of 'strands' have been moved away from the central document hierarchy and are now stored in the personal area of their maintainers. Also, many people are taking advantage of the new server to start personal 'strands' located in their personal directories. Based on these preliminary experiences we believe that the mechanisms we have discussed provide an exceptionally flexible and extensible platform for information dissemination.

In conclusion, we have found that only through the flexibility provided by Deceit have we, as maintainers, been able to shift responsibility for information onto its author, thus reducing our workload and increasing the range of information available.

Glossary

Document Generators provide a mechanism for producing documents which have no fixed form - these might include server usage and searches.

Handlers validate requests against the available source forms and generate appropriate responses from them.

Libraries provide related groups of common routines for use in handlers and document generators.

Search Engines are a special sub-section of document generators, associated with the GET method containing a search.

Source Forms local source representations for documents - these are converted by handlers into suitable request responses.

Volume Map the point of mapping between document hierarchies and physical storage.

References

[1] Williams, N. and Wilkinson, T. *Experiences in Writing a WYSIWYG Editor for HTML*, Proceedings of WWW'94, Switzerland.

[2] IETF, *HTTP: A protocol for networked information*, Working Draft.

[3] Luotonen, Ari *CERN httpd*, Online documentation.

[4] Sanders, Tony *Plexus HTTPD Overview*, Online documentation.

[5] McCool, Rob *NCSA httpd overview*, Online documentation.