HTACCESS: BILBAO METHOD EXPOSED

FraMe (frame at kernelpanik.org)
MaDj0kEr (mad at j0ker.net)
http://www.kernelpanik.org

0.- Introduction

This paper explains a brief form to avoid .Htaccess authentification in Apache, configured according to criteria of multiple references[1]. In order to understand this document it requires a deep knowledge of protocol HTTP/1.1[2]

1.- LIMIT directive in .htaccess: Significant Interpretations.

Directive LIMIT, despite written in other papers[3], where it is clearly exposed that should not be used in a general way, is included as a common use directive, this seems to be a historical mistake[4], in a great number of configuration manuals and therefore in many websites

1.1.- WEB.MIT.EDU: Limited and secure access to web content over https

Usually, http clients uses only this two methods: GET and POST. GET method is often used to request files during navigation session. On the other hand, POST method is used to send a server any client´s informations, by example, as sending a form.

A good assumption would be that limiting GET method on an area, therefore we will avoid that a http client can browse it. Massachusetts Institute of Technology is using this method, in their area of secure access to web contents.

An example of htaccess will be exposed in the following caracteristics, described above and its behavior before requesting GET and POST.

Example 1: .htaccess with <LIMIT GET>

```
AuthType Basic
AuthName Private
AuthUserFile /etc/httpd/pass/private
<LIMIT GET>
require valid-user
</LIMIT>

[frame@localhost] $ telnet localhost 80
GET /frame/private/index.html HTTP/1.0
HTTP/1.1 401 Authorization Required

POST /frame/private/index.html HTTP/1.0
HTTP/1.1 200 OK
<html><head></head><body> You shouldn't be here</body></html>
```

1.2. W3C: Protecting Confidential Documents at Your Site

Sometimes, we can find the system administrator decides to limit the POST method as well, perhaps for himself, or maybe protecting scripts they really use POST method.

In the second case, i.e. when the directory contains scripts, written in perl, php, or other kind of script language defined in the Apache configuration as interpreted by a module (LoadModule/AddModule[5]), we have to consider the way that calls to an Apache module work.

In overview, Apache after have checked that the method which asked for content is not restricted by a <LIMIT> clause, delivers the request to the matching module. That module does NOT perform any check about the used method, leaving this duty to the script developer. If the developer has not worried about any kind of restriction or control over the method used (REQUEST_METHOD), result of this call will be the execution of a plain GET method.

As an example we'll use the proposed configuration by the World Wide Web Consortium, in the "The World Wide Web Security FAQ", section 7, "Protecting Confidential Documents at Your Site"[6].

Example 2: <LIMIT GET POST>

```
AuthType Basic
AuthName Private
AuthUserFile /etc/httpd/pass/private
<LIMIT GET POST>
require valid-user
</LIMIT>

[frame@localhost] $ telnet localhost 80
POST /frame/private/index.html HTTP/1.0
HTTP/1.1 401 Authorization Required

PUT /frame/private/index.html HTTP/1.0
HTTP/1.1 405 Method Not Allowed

PUT /frame/private/index.php HTTP/1.0
HTTP/1.1 200 OK
You shouldn't be here
```

As we show in this example, a method not allowed by default in Apache configuration (the PUT method), can be executed over a content that uses an Apache module, in this time mod_php.

1.3.- Limiting the unlimitable: PUT and DELETE.

There are lot of organizations[7], like important ISPs, universities, government institutions, and international mailing lists, which trying to get maximum protection, they suggest to disallow two methods, PUT and DELETE, and although they are not supported by default in Apache, as we've seen in the point before, they could bring us troubles with some kind of contents.

Example 3: <LIMIT GET POST PUT DELETE>

```
AuthType Basic
AuthName Private
AuthUserFile /etc/httpd/pass/private
<LIMIT GET POST PUT DELETE>
require valid-user
</LIMIT>

[frame@localhost] $ telnet localhost 80
OPTIONS /frame/private/index.html HTTP/1.0
HTTP/1.1 200 OK
Allow: GET,HEAD,OPTIONS,HEAD,POST,TRACE

OPTIONS /frame/private/index.php HTTP/1.0
HTTP/1.1 200 OK
You shouldn't be here
```

Perhaps, until this time, the reader of this paper, and obviously all that have configured any LIMIT directive in the way we showed before, they didn't realized that when in the point 1.2 we told that Apache doesn't perform any kind of validation in the method that calls for a dinamic content, more than checking if that content is restricted by a <LIMIT> clause, it's only that which it performs, not making any other kind of validation, excepting the TRACE method.

Hence an OPTIONS method, that on a static content should return an output like the first OPTIONS in our example, asking for it on a dinamic one returns an output like the second example.

As we don't want to make this paper very long and full of useless examples, we want to say we've found lot of examples using configurations that even try to limit all the supported methods of the web server, writing limitations clauses like the next one:

<Limit GET HEAD POST PUT DELETE OPTIONS>.

2.- Extending HTTP/1.1: The BILBAO method.

In this place, we have all the HTTP/1.1's methods, including the ones provided by DAV, restricted with a LIMIT clause. Now it comes the following question: Is it imposible to get a protected content in this way?. The answer is if the content we want to get is served by one of Apache modules, and not for Apache itself, no type of limitations with LIMIT clauses will avoid access to this content.

To demonstrate that fact we have created the BILBAO method, although you can name it as you prefer, anyway. With this method, any type of content served by an Apache module will be shown like it was called using a GET method, overriding any restricción imposed by the <LIMIT> clause. As we told in point 1.2, is the module who must check the method used... even if that method doesn't exist in the HTTP/1.1 protocol.

Example:

AuthType Basic
AuthName Private
AuthUserFile /etc/httpd/pass/private
<Limit GET POST PUT DELETE OPTIONS PROPFIND PROPPATCH MKCOL COPY MOVE LOCK UNLOCK>
require valid-user
</LIMIT>

[frame@localhost] $ telnet localhost 80
BILBAO /frame/private/index.php HTTP/1.0
HTTP/1.1 200 OK
You shouldn't be here

BILBAO /cgi-bin/index.cgi HTTP/1.0
HTTP/1.1 200 OK
You shouldn't be here

3- Protection measures: The end of LIMIT directive.

After all we've explained, it seems obvius to think that nobody should never use a <LIMIT> clause as a method to protect the general access to a restricted zone. Use of this clause is only recommended for times when knowing how does it work, you want to limit methods in an area properly made.

4.- Conclusions

These document is applicable to any Apache version, 1.3.x series and 2.0.X series. Futhermore, we have detected that the mod_dav module disallow using unimplemented methods and its own restricted methods.

The <LimitExcept> directive on Apache is not affected by no one of the features descripted in this paper.

The use of the AddType directive on non-dinamic files with non-dinamic content, like .html, .htm or other extensions, turn into accesible the static content applying methods not implemented by HTTP according this paper.

There's no proof over httpd´s different than Apache. Nevertheless, and although is pretty possible that other httpd´s doesn't answer to unimplemented methods, it is quite possible that using the <LIMIT GET> directive could be bypassed in the same way we told in this paper. We recommend the revision of this directive in all the servers using it.

5.- Grettings

Kernelpanik Labs, Zhodiac, !dsr Staff and Huno.

6.- References

[1] http://www.google.es/search?q=%22LIMIT+GET%22
[2] http://www.w3.org/Protocols/rfc2616/rfc2616.html
[3] http://httpd.apache.org/docs/mod/core.html#limit
[4] http://www.apacheweek.com/issues/97-09-05#configerrors
[5] http://httpd.apache.org/docs/mod/core.html#addmodule
[6] http://www.w3.org/Security/faq/wwwsf5.html#CON-Q6
[7] http://www.google.es/search?&q=%22LIMIT+GET+POST+PUT%22

7.- License