HTACCESS: BILBAO METHOD EXPOSED

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

0.- Introducción.

El presente documento explica de forma breve un procedimiento para sobrepasar la autenticacion de Apache mediante el mecanismo de .htaccess, configurado según el criterio de múltiples referencias[1]. La óptima comprensión de este texto requiere conocer el protocolo HTTP/1.1[2]

1.- Directiva LIMIT en .htaccess: Interpretaciones relevantes.

La directiva LIMIT, a pesar de lo escrito en otros textos[3], donde claramente se expone que no debe ser usada de manera general, es incluida como directiva de uso común, por lo que parece ser un fallo histórico[4], en un gran número de manuales de configuración y por extensión en multitud de sitios web.

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

Por norma general, los clientes del protocolo HTTP únicamente utilizan dos de sus métodos, estos son GET y POST. El método GET es el usado habitualmente para realizar las peticiones de archivos a lo largo de la navegación. Por otra parte el método POST tiene como misión hacer llegar al servidor algún tipo de información adicional desde el cliente, por ejemplo, el envío de datos de un formulario.

Una buena suposición sería que limitando el método GET sobre un área, evitaremos que un cliente de HTTP pueda tener acceso a la misma. De hecho así parece suponerlo el Instituto Tecnológico de Massachusets, en su área[5] dedicada a cómo securizar el acceso a contenido web.

A continuación se expondrá un ejemplo de .htaccess siguiendo este criterio y su comportamiento ante las peticiones GET y POST.

Ejemplo 1: .htaccess con <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>Usted no deberia ver este contenido</body></html>

1.2. W3C: Protecting Confidential Documents at Your Site

En ocasiones, podemos encontrarnos con que el administrador del sistema, decide limitar también el uso del método POST, ya sea por iniciativa propia, o por estar protegiendo scripts que hagan uso real del mismo.

En el segundo caso, es decir cuando el directorio contiene scripts, bien sean perl, bien php, bien cualquier otro tipo definido en la configuración de Apache como procesable por un módulo (LoadModule/AddModule[6]), debemos considerar el funcionamiento de llamadas a módulos de Apache.

A grandes rasgos, Apache tras haber comprobado que el método con el que se ha solicitado el contenido no viene limitado por una cláusula <LIMIT>, entrega la petición al módulo correspondiente, el cual NO efectúa ningún chequeo sobre el método empleado, dejando esta tarea en manos del desarrollador del script. Si éste desarrollador no ha aplicado ningún tipo de restricción o control sobre el método con el que se le invoca (REQUEST_METHOD), algo que es totalmente común, el resultado de esta llamada será a todos los efectos, la ejecución de un método GET.

Como muestra de ejemplo usaremos la configuración propuesta por el World Wide Web Consortium, en su documento “The World Wide Web Security FAQ”, sección 7, “Protecting Confidential Documents at Your Site”[7].

Ejemplo 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
Usted no deberia ver este contenido

Como se puede comprobar en este ejemplo, un método que no está permitido por defecto en Apache, como es el método PUT, puede ser ejecutado sobre un contenido que hace uso de un módulo de Apache, en este caso mod_php.

1.3.- Limitando lo ilimitable: PUT y DELETE.

Existe abundante literatura[8], entre la que aparecen importantes proveedores de servicios, universidades, organizaciones gubernamentales, y listas de usuarios de ámbito internacional, donde en un intento de precaución máxima, deniegan dos métodos, PUT y DELETE, que si bien por defecto no están soportados por Apache, como hemos visto en el punto anterior pueden traer complicaciones ante determinados contenidos.

Ejemplo 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
Usted no debiera ver este contenido
```

Quizá hasta este momento el lector de este documento, y obviamente todos aquellos que han configurado alguna directiva LIMIT de la forma descrita anteriormente, no se habían percatado de que cuando en el punto 1.2 hemos dicho que Apache no efectúa ningún tipo de comprobación sobre el método que invoca un contenido dinámico, más allá de verificar si está restringido por una cláusula <LIMIT>, implica efectivamente eso, que no hace ningún tipo de comprobación, excepción del método TRACE.

Por tanto un método OPTIONS, que sobre un contenido estático debiera devolver una salida como la del primer OPTIONS de nuestro ejemplo, sobre uno dinámico devuelve una salida como la del segundo.

Para no extender en exceso el documento, decir que hemos encontrado numerosos ejemplos de configuración que incluso llegan a limitar todos los métodos soportados por su servidor web, llegando a escribir limitaciones como la siguiente:

<Limit GET HEAD POST PUT DELETE OPTIONS>.

2.- Extendiendo HTTP/1.1: El método BILBAO.

Llegados a este punto, todos los métodos de HTTP/1.1, incluidos los proporcionados por DAV, han sido limitados en una cláusula LIMIT. Ahora surge la siguiente pregunta: ¿Es imposible acceder a un contenido protegido de esa forma?. La respuesta es que si el contenido al que queremos acceder depende de alguno de los módulos de Apache, y no de Apache en sí, ningún tipo de limitación impuesto en la cláusula LIMIT impedirá que el contenido pueda ser accedido.

Para demostrarlo hemos creado para la ocasión el método BILBAO, aunque cada cual le puede dar el nombre con el que se sienta más cómodo. Con este método, cualquier tipo de contenido servido por un módulo de Apache será mostrado como si de un método GET se tratara, saltando cualquier restricción impuesta por la cláusula <LIMIT>, dado que, y como ya hemos comentado en el apartado 1.2, es el módulo en el que se delega el chequear cual es el método que lo invoca, incluso si el método no existe en el protocolo HTTP/1.1.

Veámoslo con un ejemplo:

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
Usted no deberia ver este contenido

BILBAO /cgi-bin/index.cgi HTTP/1.0
HTTP/1.1 200 OK
Usted no deberia estar viendo esto

3- Medidas de protección: El fin de la directiva LIMIT.

Después de todo lo expuesto, parece obvio pensar, que la medida de protección pasa por que nadie vuelva a usar nunca una directiva <LIMIT> como método para limitar el acceso general a una zona restringida. Limitando el uso de la misma a casos en los que concretamente, y previo conocimiento de las implicaciones de la misma, se quiera limitar un determinado método sobre un área.

4.- Consideraciones finales

Este documento es aplicable a cualquier versión de Apache, tanto de su rama 1.3.x, como de su rama 2.0.x. Así mismo, hemos detectado que el módulo mod_dav impide la aplicación de métodos no implementados, y de métodos restringidos por él.

La directiva <LimitExcept> sobre Apache no es afectada por ninguna de las características descritas en ste documento.

El uso de la directiva AddType sobre ficheros de contenido no dinámico, como .html, .htm u otras extensiones, convierten al contenido estático en accesible aplicando métodos no implementados por HTTP según lo descrito en este documento.

No se han realizado pruebas sobre httpd´s distintos a Apache. No obstante, y aunque es bastante probable que otros httpd´s no contesten a métodos no implementados, sí es probable que el uso de la directiva <LIMIT GET> pueda ser sobrepasada de la misma forma que la descrita en este documento. Es por ello por lo que aconsejamos la revisión de esta directiva en todos aquellos servidores que hagan uso de ella.

5.- Agradecimientos

- Kernelpanik: Por seguir todos más o menos vivos.
- Zhodiac & !DSR Staff: Por los comentarios a la versión no pública.
- Huno: Por el intento de traducción.

6.- Enlaces del documento

[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://web.mit.edu/cwis/web/htaccess-usage.shtml
[6] http://httpd.apache.org/docs/mod/core.html#addmodule
[7] http://www.w3.org/Security/faq/wwwsf5.html#CON-Q6
[8] http://www.google.es/search?&q=%22LIMIT+GET+POST+PUT%22

7.- Licencia