MÓDULO PERL APACHE: "RewritingProxyTME"

Román Medina-Heigl Hernández

Revisión 5 (16/09/2005)

"RewritingProxyTME es un módulo Perl para Apache que provee la funcionalidad de proxy inverso con características avanzadas. Ha sido desarrollado en Telefónica Móviles España (TME), a partir de otro módulo incluido en el proyecto PAPI de RedIRIS, y se distribuye bajo licencia GNU."

ÍNDICE

1. OBJETIVO

Este documento describe detalladamente las funcionalidades provistas por el nuevo módulo Perl de Apache llamado "RewritingProxyTME" así como todas las directivas necesarias para su configuración y correcta utilización en un proxy inverso. Se sugerirá asimismo diferentes formas de uso basándonos en escenarios reales que nos podemos encontrar a la hora de integrar webs.

2. ÁMBITO DE APLICACIÓN

Un **proxy inverso** se utiliza comúnmente para dar servicios web de forma segura, enmascarando y protegiendo a los servidores web reales. Para ello es situado en primera línea y atiende las peticiones HTTP / HTTPS del "exterior" (ej. Internet). Internamente mantiene una lista de "mapeos", que relacionan una o varias URLs del proxy (visibles externamente) con los servidores web "internos", de forma que cuando un usuario accede a una de estas URLs "mapeadas" el proxy primero contacta con el servidor web adecuado, para obtener la página pedida, y posteriormente devuelve el contenido al usuario. El proceso es totalmente transparente: el usuario no se entera de que hay uno o varios servidores web detrás del proxy inverso ni de los mapeos existentes; el único elemento visible es el propio proxy, el cual deberá estar correctamente securizado y actualizado, y que puede incluir funcionalidades adicionales como "caching" (guardar temporalmente las páginas en caché para poder servirlas directamente y así ahorrar tiempo y recursos en el servidor web), "IPS" (protección ante intrusiones), autenticación (sin tener que alterar el código de las aplicaciones web finales y de forma transparente lo que facilita la implantación de políticas de seguridad) o encriptación.

Este documento va dirigido a:

- Personal encargado de la integración y mantenimiento de proxies inversos.
- Toda aquella persona interesada en mejorar el código del módulo.

3. GLOSARIO DE TÉRMINOS

Proxy-web: máquina que, realizando la función de proxy inverso, se utiliza como pasarela intermedia para acceder a servicios finales, añadiendo funcionalidades que securizan el servicio (ej.: cumplimiento de LOPD, encriptación, etc).

Módulo Perl Apache: programa realizado en Perl que añade una funcionalidad extra al software servidor Apache y que se ejecuta gracias a la extensión "mod_perl", que debe estar activada en el mismo.

Módulo "RewritingProxyTME": es el módulo tratado en este documento. No es más que un módulo Perl Apache que provee la funcionalidad de proxy inverso. Se integra en los proxies-web, junto a otros módulos (como el de autenticación y autorización).

4. ANTECEDENTES

Antes de desarrollar el presente módulo se contaba con dos opciones diferentes para llevar a cabo una integración en los proxies-web: ReverseProxy.pm y ProxyRewrite.pm. Se trata de dos módulos Perl que implementan la funcionalidad de proxy inverso y que, por desgracia, se ha demostrado que resultan insuficientes para mantener una tasa de éxito en integraciones adecuada (con estos módulos se ronda el 60-70%). Entre sus deficiencias podemos destacar las principales:

- Incorrecto tratamiento de las cookies de la aplicación final.
- Reescritura de URLs en muchos casos incorrecta o insuficiente así como falta de flexibilidad a la hora de su configuración.

5. MÓDULO "REWRITINGPROXYTME"

5.1. INTRODUCCIÓN

El módulo "RewritingProxyTME" se presenta como la alternativa a los dos módulos anteriores, corrigiendo todas las deficiencias observadas, incluidas las descritas anteriormente e incorporando nuevas opciones y mejoras. Utilizando únicamente el nuevo módulo se puede conseguir un proxy inverso completo.

El módulo está fuertemente basado en el "RewritingProxy.pm", desarrollado por RedIRIS, bajo el proyecto PAPI (http://papi.rediris.es), y con licencia GPL. El original dependía de otros módulos y tenía también otras carencias. Lo que se ha hecho es adaptarlo a las necesidades de TME e incluir a su vez bastantes mejoras.

5.2. MODIFICACIONES REALIZADAS

Se han llevado a cabo las siguientes modificaciones sobre el módulo original de PAPI:

- Lee la configuración de directivas PerlSetVar / PerlAddVar en vez del objeto PoA y funciona de forma autónoma sin necesidad de otros módulos PAPI.

- El nombre de la cookie de autenticación se especifica en la variable $TMECookie. Por defecto, se usa: "Apache::AuthCookieLDAP_APACHE".
- Soporte HTTPS para Remote_Domain (añade la opción "Remote_Domain_HTTPS").
- Rewrite funciona tanto para HTTP como HTTPS (antes siempre asumía HTTP).
- Opción "Raw_Redirect_All" (se salta el parseador TokeParser y aplica el rewrite directamente al contenido de la página).
- Opción "Debug" para facilitar la depuración de este modulo.
- Opción "Post_Redirect", que permite la reescritura de los datos enviados en peticiones POST.
- Opción "Cookie_AddString", para realizar una "traducción" de cookies que permita utilizar diferentes servicios que hacen uso de cookies con igual nombre, evitando conflictos.
- Opción "NoCache", para evitar que el servidor final haga "caching" (útil a la hora de depurar posibles problemas, nuevas integraciones, etc.).
- Mayor flexibilidad en la reescritura de contenido (URLs, etc.). Nuevas directivas: "PAPI_Redirect_Delim", "PAPI_Redirect_Extended", "Post_Redirect_Delim", "Post_Redirect_Extended", "Rewrite_MIME_Types_Extended" y "Rewrite_URL_Pattern_Extended".
- Numerosos "bug-fixes" que causaban incompatibilidades en las integraciones.

Nota: *Puede consultar la lista exhaustiva de cambios en el "Changelog" incluido en el propio módulo.*

5.3. VENTAJAS Y MEJORAS

El módulo "RewritingProxyTME" presenta diversas mejoras, que solucionan una buena parte de los problemas de incompatibilidad que suelen surgir en la integración de webs cuando se utilizan los módulos antiguos –ReverseProxy y ProxyRewrite-) y consigue elevar la tasa de éxito en integración por encima del 95%. Algunas de estas mejoras ya se incluían en el módulo original de PAPI y otras se derivan directamente de las modificaciones implementadas por TME. Se resumen a continuación:

- El tratamiento de cookies ha sido revisado. Los módulos anteriores presentaban problemas cuando el servidor enviaba varias líneas de cabecera Set-Cookie. Además ahora la cookie de autenticación no se envía a las aplicaciones finales. Esto soluciona problemas observados como por ejemplo las aplicaciones web que no tratan bien las cookies y fallan al recibir una cookie que no esperaban. También se descubrieron fallos debidos a que la cookie de autenticación que actualmente se usa contiene el carácter ":", que normalmente es prohibido y provoca malfuncionamiento en ciertas aplicaciones. Todo eso se evita simplemente no enviando dicha cookie a la aplicación final (no la necesita para nada). Por último,

ahora es posible evitar conflictos con cookies de similar nombre en diferentes servicios, sin más que usar la nueva característica de traducción de cookies.

- Configuración de reglas de reescritura de URLs totalmente flexible. Se permite incluir un número indeterminado de reglas, usar expresiones regulares e incluso definir a qué tipo de contenido afectarán esas reglas (por defecto, afectan sólo a contenido HTML). Para esto último se pueden definir los MIME-Types que se verán afectados y/o patrones de URLs (basados también en expresiones regulares). Se ha añadido además la posibilidad de traducir datos que son enviados en las peticiones POST.
- Se permite incluir variables del módulo Perl como parte de directivas de configuración (reglas de reescritura, etc). De esta forma, el grado de flexibilidad es aún mayor.
- Facilita la integración de diferentes servidores web bajo un mismo dominio. Para ello, con una sóla directiva es posible que el módulo traduzca las URLs de la forma http://proxy/host/location/... a http://host.remote.domain/location/.... Esta característica normalmente no la utilizaremos pero ya se encontraba en PAPI, por lo que se ha respetado y mantenido.
- Todas las directivas del módulo original de PAPI han sido conservadas tal cual (en la medida de lo posible), de forma que exista la mayor compatibilidad posible entre este módulo y el proyecto PAPI. Gracias a esto, podemos hacer uso del repositorio que RedIRIS ofrece gratuitamente (http://papi.rediris.es/comu/proxies/), donde se muestran diferentes y variados casos de integración, que pueden servir como modelo a la hora de configurar adecuadamente el módulo y conseguir con éxito integrar una web dada. A partir de la versión 0.6 del módulo el comportamiento por defecto ha cambiado de forma que la reescritura de reglas se realiza de forma más simple; para obtener un comportamiento similar al anterior (y al que realiza PAPI) es necesario activar el modo "extendido", que admite el uso de expresiones regulares, etc.
- A su vez, la aplicación de las reglas de escritura se puede realizar en dos modalidades. Por defecto, se realiza una reescritura "inteligente", de forma que sólo se sobreescriben URLs en aquellos campos de los correspondientes tags HTML donde normalmente van los enlaces. Por si lo anterior falla (el parser HTML no es perfecto) existe una segunda modalidad (Raw_Redirect_All) que aplica las reglas sobre todo el documento, sin tener en cuenta el código y etiquetas HTML.
- Es posible especificar un proxy a través del cual alcanzar el servidor web final, incluso aún siendo éste autenticado, resultando transparente al usuario.
- Auto-relleno y posterior envío de formularios web. Si por ejemplo la aplicación final contiene una página de login basada en un formulario, podemos hacer que el proxy, al cargar la página, rellene automáticamente el formulario, lo envíe y muestre directamente la respuesta al usuario. De esta forma podemos integrar directamente en el proxy inverso webs finales que están autenticadas, pero que de cara al usuario no lo estarán, y por tanto, la única autenticación que ha necesitado el mismo para entrar en la aplicación es la de los proxies-web. Con este método se puede ahorrar la gestión de usuarios en el servidor web final, a la vez que éste se mantiene protegido (si alguien se conecta directamente al mismo sin pasar por los proxies,

necesitará conocer el usuario y contraseña de la aplicación). Se pueden incluir un número indeterminado de estas reglas de procesamiento de formularios.

- Autenticación HTTP automática (tanto "Basic" como "Digest"). Al igual que en el caso anterior (formularios), el módulo puede responder automáticamente a peticiones de "HTTP Auth", de forma transparente al usuario.
- Todas las directivas de configuración se pueden incluir en la configuración Apache del vhost correspondiente, mediante "PerlSetVar" y/o "PerlAddVar". Así es posible personalizar cada vhost de forma rápida, sencilla y elegante.
- El módulo añade bastantes líneas de "debug" (desactivadas por defecto), de forma que en caso de dificultades cuando se está integrando una web nos puedan dar pistas acerca de cómo resolver el problema y orientar al integrador.

5.4. CONFIGURACIÓN

Las siguientes directivas de configuración están disponibles:

- **Remote_URL**

  La usaremos en la gran mayoría de los casos ya que define la ubicación de la web final. El proxy conectará a la misma para obtener la información que posteriormente servirá a los clientes. Se deberá escribir sin la "/" del final. Ejemplo:

  ```
  PerlSetVar Remote_URL http://10.253.65.16:81
  ```

- **Remote_Domain**

  Esta característica se usa en alternancia con la directiva anterior (en toda configuración deberá existir una u otra). Provoca que se traduzcan las URLs de la forma:

  http://proxy/host/location/... <--> http://host.remote.domain/location/....

  Ejemplo:

  ```
  PerlSetVar Remote_Domain ieee.org
  ```

- **Remote_Domain_HTTPS [1|0]**

  Por defecto (valor 0) cuando se utiliza la directiva Remote_Domain todas las URLs que se construyen son de la forma http://. Es decir, se traduce:

  http://proxy/host/location/... <--> http://host.remote.domain/location/....

  https://proxy/host/location/... <--> http://host.remote.domain/location/....

  Activando esta opción (valor 1) forzamos a que sean https://. Así:

  http://proxy/host/location/... <--> https://host.remote.domain/location/....

  https://proxy/host/location/... <--> https://host.remote.domain/location/....

Ejemplo:

```
PerlSetVar Remote_Domain_HTTPS 1
```

Nota: *el tipo de URL (http o https) con la que se accede al proxy, es decir, la URL origen en las "traducciones", depende únicamente de la configuración del Apache que implementa el proxy inverso.*

- **Redirect_All [1|0]**

  Por defecto (valor 0) las reglas de reescritura sólo se aplican a los enlaces activos, como por ejemplo una URL dentro de una etiqueta HREF o dentro de parámetros SRC de una etiqueta IMG o SCRIPT. Para ciertos casos muy particulares puede ser conveniente que la reescritura se haga en todo tipo de enlaces, para lo cual se deberá activar esta directiva, siempre con extremo cuidado. Ejemplo:

  ```
  PerlSetVar Redirect_All 1
  ```

- **Raw_Redirect_All [1|0]**

  Por defecto (valor 0) el motor de reescritura se aplica de forma inteligente, haciendo un "parsing" del contenido HTML de la página, y analizando si ciertos campos/tags pueden contener URLs o no. En algunos casos este análisis puede fallar o dar resultados no esperados. **Sólo para estos casos** se deberá entonces activar esta directiva, que fuerza a que la reescritura se haga sobre todo el contenido de la página, saltándose el parser. Ejemplo:

  ```
  PerlSetVar Raw_Redirect_All 1
  ```

- **Strip_Location [1|0]**

  Por defecto (valor 0) cuando el proxy recibe una petición pasa la URI sin alterar a la web final. Si se activa esta opción, el proxy prescindirá del valor de la localización ("location") y ésta no será enviada a la web final. Esta característica resulta útil si se pretende mapear varias webs remotas a diferentes localizaciones del proxy. Ejemplo de uso:

  ```
  <Location /web>
    PerlSetVar Remote_URL                 http://one.dom.ain
    PerlSetVar Strip_Location             1
  </Location>
  <Location /anotherweb>
    PerlSetVar Remote_URL                 http://two.dom.ain
    PerlSetVar Strip_Location             1
  </Location>
  ```

  Se traducirían:

  http://proxy/web/location/... <--> http://one.dom.ain/location/...

  http://proxy/anotherweb/location/... <--> http://two.dom.ain/location/...

- **Proxy_Server / Proxy_User / Proxy_Pass**

  Especifica un servidor proxy (en forma de URL) que será usado para alcanzar la web final. El proxy puede requerir autenticación, en cuyo caso deberemos utilizar además las directivas Proxy_User y Proxy_Pass. Ejemplo:

  ```
  PerlSetVar Proxy_Server http://192.168.0.9:8080
  PerlSetVar Proxy_User "usuario"
  PerlSetVar Proxy_Pass "password"
  ```

- **HTTP_Auth**

  Válido para responder automáticamente a cualquier petición de autenticación HTTP (tanto "Basic" como "Digest"). Puede haber múltiples directivas de este tipo (se distinguen por el campo "realm", que especifica el recurso remoto). Ejemplo:

  ```
  PerlAddVar HTTP_Auth "realm::username::password"
  ```

- **Form_Processor**

  Configura el "auto-relleno" de formularios. Puede haber varias directivas de este tipo. Cada una deberá incluir la URL de la página donde reside el formulario en cuestión así como los pares campo-valor que el módulo rellenará automáticamente. Ejemplo:

  ```
  PerlAddVar Form_Processor "http://192.168.0.3/dir/login.php usuario
  => roman, pass => secret"
  ```

- **PAPI_Redirect_Extended [1|0]**

  Por defecto (valor 0) se deshabilita el uso de expresiones regulares en las directivas "PAPI_Redirect" (lo cual simplifica la sintaxis de las reglas ya que no hay que "escapar" caracteres especiales). Si activamos el modo "extendido" (valor 1) el comportamiento será el habitual de PAPI, es decir, se pueden usar expresiones regulares, etc (con el inconveniente de que habrá que utilizar el carácter "\" de escape). Ejemplo:

  ```
  PerlSetVar PAPI_Redirect_Extended 1
  ```

- **PAPI_Redirect_Delim**

  Expresión regular que define el delimitador usado para separar la expresión a reemplazar por la cadena de reemplazo, en las directivas PAPI_Redirect. Por defecto es '\s+', es decir uno o más caracteres en blanco (espacio, tabulador, etc). Ejemplo:

  ```
  PerlSetVar PAPI_Redirect_Delim '\s+'
  ```

- **PAPI_Redirect**

  Especifica una regla de reescritura. En modo "extendido", admite expresiones regulares e incluso variables internas del proxy (éstas serán evaluadas antes de

realizar la reescritura). Puede haber diferentes líneas PAPI_Redirect, especificadas mediante PerlAddVar (en vez de PerlSetVar).

Ejemplo modo "normal" (por defecto):

```
PerlAddVar PAPI_Redirect "http://192.168.0.3 https://proxy:12638"
```

Ejemplo modo "extendido":

```
PerlAddVar PAPI_Redirect "([\w]+).ieee.org MPROXYNAME/$1/"
```

- **Post_Redirect_Extended [1|0]**

  Por defecto (valor 0) se deshabilita el uso de expresiones regulares en las directivas "Post_Redirect" (lo cual simplifica la sintaxis de las reglas ya que no hay que "escapar" caracteres especiales). Si activamos el modo "extendido" (valor 1) se podrá usar expresiones regulares, etc (con el inconveniente de que habrá que utilizar el carácter "\" de escape). Ejemplo:

  ```
  PerlSetVar Post_Redirect_Extended 1
  ```

- **Post_Redirect_Delim**

  Expresión regular que define el delimitador usado para separar la expresión a reemplazar por la cadena de reemplazo, en las directivas Post_Redirect. Por defecto es '\s+', es decir uno o más caracteres en blanco (espacio, tabulador, etc). Ejemplo:

  ```
  PerlSetVar Post_Redirect_Delim '\s+'
  ```

- **Post_Redirect**

  Especifica una regla de reescritura que se aplicará a las peticiones POST que el proxy reciba. Su sintaxis es similar a la de "PAPI_Redirect".

  Ejemplo modo "normal" (por defecto):

  ```
  PerlAddVar Post_Redirect "cadena nuevacadena"
  ```

  Ejemplo modo "extendido":

  ```
  PerlAddVar Post_Redirect "^(validaOper\#.*\#.*) $1\n"
  ```

- **Rewrite_MIME_Types_Extended [1|0]**

  Por defecto (valor 0) se deshabilita el uso de expresiones regulares en la directiva "Rewrite_MIME_Types" (lo cual simplifica la sintaxis ya que no hay que "escapar" caracteres especiales). Si activamos el modo "extendido" (valor 1) el comportamiento será el habitual de PAPI, es decir, se pueden usar expresiones regulares, etc (con el inconveniente de que habrá que utilizar el carácter "\" de escape). Ejemplo:

  ```
  PerlSetVar Rewrite_MIME_Types_Extended 1
  ```

- **Rewrite_MIME_Types**

  Por defecto el módulo sólo aplica el motor de reescritura sobre contenido HTML / XHTML. Si se desea que además lo haga sobre otro tipo de contenido

adicional, deberá especificarse éste con esta directiva (admite una lista de MIME-Types separadas por espacios). Ejemplo:

```
PerlSetVar Rewrite_MIME_Types "application/x-javascript tipo/mime"
```

- **Rewrite_URL_Pattern_Extended [1|0]**

  Por defecto (valor 0) se deshabilita el uso de expresiones regulares en las directivas "Rewrite_URL_Pattern" (lo cual simplifica la sintaxis de las reglas ya que no hay que "escapar" caracteres especiales). Si activamos el modo "extendido" (valor 1) el comportamiento será el habitual de PAPI, es decir, se pueden usar expresiones regulares, etc (con el inconveniente de que habrá que utilizar el carácter "\" de escape). Ejemplo:

  ```
  PerlSetVar Rewrite_URL_Pattern_Extended 1
  ```

- **Rewrite_URL_Pattern**

  Las reglas de reescritura "PAPI_Redirect", como hemos visto, se aplican sobre contenido HTML y adicionalmente sobre los tipos MIME especificados con la directiva Rewrite_MIME_Types. Además la directiva Rewrite_URL_Pattern permite especificar URLs (de web final, no de proxy) sobre los cuales se aplicará el motor de reescritura. Se pueden usar expresiones regulares (en modo "extendido") y varias líneas con esta directiva (mediante PerlAddVar).

  Ejemplo modo "normal" (por defecto):

  ```
  PerlAddVar Rewrite_URL_Pattern "http://www.tsm.es/file.js"
  ```

  Ejemplo modo "extendido":

  ```
  PerlAddVar Rewrite_URL_Pattern "\.js$"
  ```

- **Cookie_AddString**

  La cadena especificada (normalmente el puerto del vhost) será añadida al nombre de todas las cookies (excepto a la cookie de autenticación de Apache), separadas por el carácter "_". Esta modificación sólo es visible en el tramo navegador-proxy (se deshace antes de alcanzar el tramo proxy-aplicación final). De esta forma, evitamos el conflicto de cookies entre diferentes aplicaciones que usen una misma nomenclatura para las mismas, y todo ello de forma transparente en lo que a la aplicación final respecta. Ejemplo:

  ```
  PerlSetVar Cookie_AddString 10713
  ```

- **NoCache [1|0]**

  Si se activa (valor 1) esta directiva, el proxy inverso incluirá en la cabecera de las peticiones HTTP enviadas a la web final las líneas "Pragma: no-cache" y "Cache-Control: no-cache", pidiendo así que no se realice "caching". Puede ser útil para depurar a la hora de realizar nuevas integraciones. Ejemplo:

  ```
  PerlSetVar NoCache 1
  ```

- **Debug [0|1|2]**

Por defecto (valor 0) el módulo sólo registra ciertos eventos mínimos (como la URL a la que se está accediendo). Sin embargo, ante cualquier tipo de problema esta información resulta insuficiente. En estos casos resultará útil activar la depuración (valor 1), la cual está principalmente orientada al motor de reescritura, que suele ser el componente más dificultoso del módulo. Podemos obtener un grado mayor de detalle utilizando el valor 2 (imprimirá el contenido de las peticiones y las correspondientes respuestas). Ejemplo:

```
PerlSetVar Debug 1
```

5.5. EJEMPLOS

Veamos algunos ejemplos y casos de integraciones típicas:

- Un primer caso simple: la web final se encuentra en un puerto no estándar:

```
<Location />
        SetHandler perl-script
        PerlHandler Apache::RewritingProxyTME
        PerlSetVar Remote_URL http://192.168.4.53:81
</Location>
```

- Web final en puerto estándar HTTP, con una regla de reescritura. Activamos la depuración para ver posibles fallos.

```
<Location />
        SetHandler perl-script
        PerlHandler Apache::RewritingProxyTME
        PerlSetVar Remote_URL http://192.168.2.60
        PerlAddVar PAPI_Redirect "http://w3.algo.com
        https://192.168.4.3:12638"
        PerlSetVar Debug 1
</Location>
```

- Web final en puerto HTTPS no estándar, con dos reglas de reescritura, afectando tanto al contenido HTML como a los ficheros con extensión .js (JavaScript) y con reescritura realizada sobre todo el documento sin parsear (modo "raw"):

```
<Location />
        SetHandler perl-script
        PerlHandler Apache::RewritingProxyTME
        PerlSetVar Remote_URL https://192.168.1.3:8000
        PerlSetVar Rewrite_URL_Pattern_Extended 1
        PerlAddVar Rewrite_URL_Pattern "\.js$"
        PerlAddVar PAPI_Redirect "http://w3.algo.com
        https://192.168.4.3:12638"
        PerlAddVar PAPI_Redirect "http://w2.algo.com
        https://192.168.4.3:12639"
        PerlSetVar Raw_Redirect_All 1
```

```
</Location>
```

5.6. EL MOTOR DE REESCRITURA

Uno de los componentes clave del módulo (quizás el que más potencia le confiere) es el motor de reescritura. Éste se encarga de analizar el contenido de las páginas devueltas por el servidor web final y realizar los cambios ("traducciones") pertinentes, antes de entregarlo al navegador web del usuario que está accediendo al proxy inverso. Principalmente se trata de convertir "URLs de web final" (donde se haga referencia a esta última) a "URLs de proxy inverso" (donde se referencia el proxy). Normalmente sólo estas últimas URLs son accesibles directamente por el usuario, de forma que si no hiciéramos la traducción de URLs correspondiente el navegador llegaría a un callejón sin salida (por ejemplo, tratando de acceder a un enlace que apunta directamente al servidor web final).

Por defecto, la reescritura se limita a "traducir" URLs en base a "Remote_URL" o "Remote_Domain", y sólo se aplica a páginas HTML y ni siquiera a su totalidad, sino sólo a aquellas partes que son susceptibles de contener enlaces activos (por ejemplo, junto a una etiqueta HREF). Existen diferentes directivas que podemos configurar para ampliar y/o modificar este comportamiento. Debido a la complejidad que puede llegar a alcanzar detallaremos a continuación la **lógica de reescritura** para una petición arbitraria que alcance a nuestro proxy inverso:

1. El proxy recibe una petición HTTP en la "URL del proxy".
2. Se analiza la cabecera HTTP y se llevan a cabo algunas alteraciones tanto en la línea de cabecera "Cookie" (por ejemplo, se evita que se envíe la cookie de autenticación) como en otras (más ejemplos: se bloquea la cabecera "Accept-Encoding" para que la web final no nos devuelva respuestas codificadas que serían difícilmente reescribibles, se añade "Pragma: no-cache" si está activada la directiva "NoCache", se añade la línea correspondiente de autenticación HTTP si la directiva "HTTP_Auth" está definida, etc.).
3. Si la petición es POST, se aplican las reglas de reescritura "Post_Redirect".
4. Se construye la "URL de web final" y se lanza la petición correspondiente.
5. Se recibe la respuesta del servidor web final. Si existen directivas "Form_processor" (auto-relleno de formularios) en vigor se analiza el contenido devuelto y se rellena el formulario generando automáticamente una nueva petición hacia el servidor web final.
6. El proxy analiza la última respuesta recibida y realiza la reescritura (basándose en las directivas "Remote_URL", "Remote_Domain" y "PAPI_Redirect"), de la siguiente manera:
   6.1. Si es una página HTML (el campo "Content-Type" contiene la cadena "html"):
      6.1.1. Si "Raw_Redirect_All" se realiza una reescritura "raw", es decir, aplica la reescritura a todo el documento HTML.

6.1.2. Si no:

6.1.2.1. Si "Redirect_All" se realiza una reescritura inteligente que afecta únicamente a los lugares encerrados entre etiquetas HTML susceptibles de contener URLs, ya sean activos o no.

6.1.2.2. Si no, se realiza una reescritura inteligente que afecta únicamente a los lugares encerrados entre etiquetas HTML susceptibles de contener URLs (activas).

6.2. Si no es contenido HTML:

6.2.1. Si "Content-Type" está incluido en "Rewrite_MIME_Types" se realizará una reescritura "raw".

6.2.2. Si la "URL de web final" coincide con algún patrón dado por las directivas "Rewrite_URL_Pattern" se realizará una reescritura "raw".

7. Finalmente, se devuelve la respuesta al usuario que lanzó la petición, que será mostrada en su navegador.