phpPoA: Método simple de autorización mediante PAPI

phpPoA: Simple Authorization Method with PAPI

D. González Hernández y J. Palacios

Resumen

Para poder aprovechar todas las funcionalidades que nos ofrece PAPI (Punto de Acceso a Proveedores de Información) sin necesidad de integrar sus componentes con el servidor web Apache, hemos desarrollado un Punto de Acceso simple en PHP llamado phpPoA. Mediante este PoA en php podemos gestionar la autorización de usuarios en nuestro sitio web simplemente haciendo una llamada a un método, el cual nos proporcionará información sobre el usuario que intenta acceder. De esta modo no sólo conseguimos una forma simple de proteger nuestro sitio web, sino que además damos libertad el desarrollador para que muestre al usuario la información que quiera en función de la identidad del usuario.

Palabras clave: PAPI, Punto de Acceso, Autorización, PHP

Summary

In order to take advantage of all the functionalities provided by PAPI (Access Point to Information Providers) without integrate their components into Apache web server, we have developed a simple Point of Access in PHP called phpPoA. With this PoA in php we can manage users authorization in our web server by only doing a call to a method, which will provide us information about the accessing user. Therefore not only we have a simple way to protect our web site, but also the developer has freedom to show the user the information required based in user identity.

Keywords: PAPI, Point of Access, Authorization, PHP

1.- Introducción

Hemos desarrollado un Punto de Acceso PAPI simple en php que permite proteger nuestro sitio web sin necesidad de instalar PAPI ni integrarlo con el servidor Apache

En ocasiones nos resultaría muy útil poder controlar el acceso de usuarios a nuestro sitio web mediante PAPI sin tener que integrar su configuración en nuestro servidor Apache. De hecho, sería muy cómodo poder proteger nuestro sitio web sin tener que cumplir los requisitos de instalación de PAPI (servidor web Apache, intérprete perl y varias librerías, mod_perl, etc.). Por este motivo hemos desarrollado un Punto de Acceso PAPI simple en php (phpPoA) que permite proteger nuestro sitio web sin necesidad de instalar PAPI ni integrarlo con el servidor Apache. Desde cada una de las páginas html o php podemos realizar una llamada a un método del phpPoA que nos indicará si el usuario está o no autorizado para acceder a este recurso web. Además, si el usuario se ha autenticado correctamente, este método nos devolverá su identidad, lo cual constituye una valiosa información para el desarrollador que podrá generar dinámicamente la página web con la información adecuada a cada perfil de usuario.

Este Punto de Acceso en php implementa las funcionalidades de los PoA tradicionales de PAPI, además de tener una instalación muy simple y proporcionar flexibilidad a los desarrolladores

De momento es necesario incluir en cada página web una llamada al método php para chequear la autorización del usuario, aunque estamos planteándonos la posibilidad de desarrollar un pequeño módulo que añadido al servidor web permita controlar sitios web completos.

El phpPoA, ha sido probado con Apache 1.3.x y Apache 2.0.54, aunque al estar desarrollado en php puede funcionar en un principio con cualquier servidor web. De hecho también se han realizado algunas pruebas satisfactorias con Microsoft IIS.

Este Punto de Acceso en php implementa las principales funcionalidades de los PoA tradicionales de PAPI (ya que se comunica con GpoA y AS de la misma forma que los PoA implementados en Perl), además de tener una instalación muy simple y proporcionar flexibilidad a los desarrolladores.

Los requisitos de instalación se reducen a:

- Servidor web (probado en Apache)
- PHP compilado con módulos OpenSSL, Mcrypt y DBA

Estamos estudiando la posibilidad de eliminar el requisito de Mcrypt ya que la última versión de OpenSSL incluye soporte Mcrypt.

2.- Características

El phpPoA es simplificado ya que no implementa todas las funcionalidades de los PoAs desarrollados en perl, aunque sí las principales. De las dos cookies que emplean los PoAs tradicionales, el phpPoA sólo utiliza una de ellas, Lcook. Además la instalación y configuración es muy sencilla, basta con copiar las librerías en php a su ubicación habitual en el servidor y editar un pequeño fichero de configuración (papi.ini).

De las dos cookies que emplean los PoAs tradicionales, el phpPoA sólo utiliza una de ellas, Lcook

La comunicación del phpPoA con el GpoA definido en la configuración se hace con las mismas garantiías de seguridad que los PoAs tradicionales, de hecho emplea los mismos algoritmos de encriptación tanto para la cookie (AES) como para la comunicación (RSA). Al proteger recursos web de forma individualizada, podemos implementar políticas de seguridad independientes para cada recurso, aunque todas ellas definidas de forma centralizada en el fichero de configuración independiente.

TABLA 1: COMPARATIVA ENTRE UN PoA EN PERL Y UN phpPoA

PoA vs phpPoA		
	PoA	**phpPoA**
Lenguaje de programación	Perl	PHP
Configuración	Integrada en Apache	Fichero de configuración independiente
Ámbito de protección	Conjunto de recursos	Individualizado
Cookies	Lcook y Hcook	Lcook
Interacción	GpoA	GpoA

El phpPoa tiene dos modos de funcionamiento:

• Redirección automática

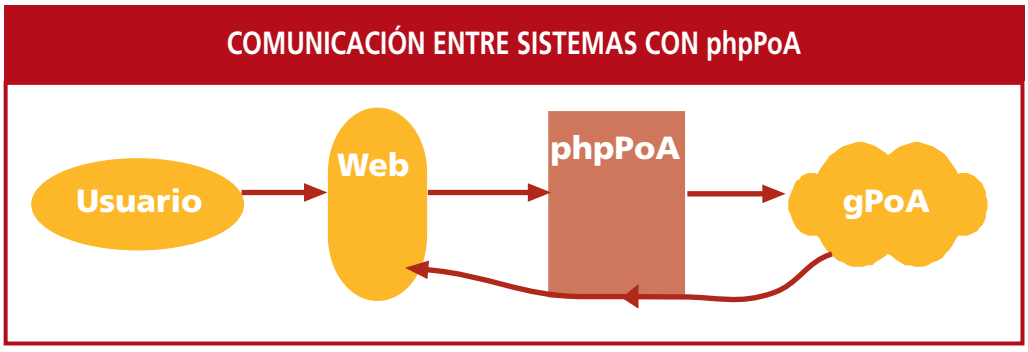
• Simple autorización

3.- Funcionamiento

El phpPoa tiene dos modos de funcionamiento:

- Redirección automática ($auto=1). En este modo de funcionamiento, cuando se produce un error, el propio phpPoA redirige al usuario a una página de error definida en el fichero de configuración.
- Simple autorización ($auto=0). En este modo de funcionamiento, no se producen redireccionamientos, siempre se devuelve un código a la página web accedida indicando si el usuario ha sido autenticado correctamente o no.

Cuando un usuario intenta acceder a un recurso web, desde el propio recurso se llama al método check_Acess del phpPoA indicándole el modo de operación y la política a utilizar en este recurso. Las políticas están configuradas en el papi.ini y consisten en filtros, GPoA a utilizar, páginas de error, etc.



Si el phpPoA no puede autorizar al usuario, realiza una petición al GPoA preguntando acerca de su autorización. Si el GPoA autoriza al usuario contesta al phpPoA con un UserAssertion que le identifica. El phpPoA comprueba si el usuario cumple con los filtros de la política, y en caso afirmativo devuelve un código a la web junto con el UserAssertion.

Si el phpPoA no puede autorizar al usuario, realiza una petición al GPoA preguntando acerca de su autorización

4.- Programación con el phpPoA

El phpPoA puede utilizarse para proteger recursos de forma tradicional, aunque basado en un fichero en lugar de en Location. Pero, al estar implementado como una clase PHP, está disponible para cualquier aplicación desarrollada en PHP. En cualquier punto del código podemos confirmar la autorización del usuario utilizando el método que nos proporciona el phpPoA y, puesto que se trata de una comprobación activa, no basada en variables de entorno, nos da una gran versatilidad. Podemos utilizar diferentes niveles de autorización dentro del mismo recurso, lo que quiere decir que habrá partes del mismo que puedan ser accesibles públicamente, mientras que otras pueden permanecer inaccesibles para todos aquellos usuarios sin credenciales suficientes.

Podemos utilizar diferentes niveles de autorización dentro del mismo recurso

5.- Adaptando aplicaciones

Una de las primeras utilidades del phpPoA es la posibilidad de integrar nuestras aplicaciones web con nuestra infraestructura PAPI. Tanto si realizamos la autenticación contra nuestro propio backend, como si empleamos algún otro método específico de la aplicación, el primer paso es localizar el punto donde se realiza la autenticación, que es donde debemos situar la comprobación de las cookies PAPI. En el caso más típico, la aplicación tendrá asociada una base de datos donde se almacenan los datos de autenticación junto con información propia del usuario, generalmente relativa a configuración y en algunos casos relacionada con la autorización. Como esta información es necesaria para el correcto funcionamiento de la aplicación, nos vemos en la necesidad de realizar una segunda modificación en el código en funcionamiento. Y ya que la aserción PAPI debe identificar de forma única al usuario frente a la aplicación, lo más sencillo es extender la tabla correspondiente para almacenar este identificador, y utilizar el uid PAPI para localizar la entrada del usuario autenticado.

6.- Conclusiones

Con este nuevo PoA en php conseguimos controlar el acceso de forma individualizada a nuestros recursos web con una sencilla instalación y un mínimo de configuración. Tendremos que tener cuidado de añadir el código php a cada uno de los recursos que queramos proteger. Además permite abstraernos del servidor web que utilicemos y proporciona flexibilidad a los desarrolladores, ya que les cede el control de la aplicación para que muestren la información más adecuada en función del perfil del usuario. Aunque se trata de una implementación simple de un PoA, aprovecha las funcionalidades de PAPI e implementa las principales características de un PoA tradicional.

Referencias

- http://papi.rediris.es/comu/news/20051027.html
- ftp://ftp.rediris.es/rediris/papi/phpPoA.1.0.tar.gz
- http://papi.rediris.es

Con este nuevo PoA en php conseguimos controlar el acceso de forma individualizada a nuestros recursos web con una sencilla instalación y un mínimo de configuración

David González Hernández
(david.gonzalez@rediris.es)
Sistemas
RedIRIS
Javier Palacios Bermejo
(javier.palacios@ific.uv.es)
IFIC - CSIC