Mozilla Security Review

General Information

Project Name:	Oauth WordPress Plugin
Security Reviewer:	
Date of Review:	
Bugzilla Bug #:	769502

Background Information:

This review comes to us after a prior request for a WordPress plugin called “SurveyGizmo” on the hacks.mozilla.org blog. In order for SurveyGizmo to work, the Oauth.php plugin is needed.

From the plugin home page:
An open protocol to allow secure API authorization in a simple and standard method from desktop and web applications.

Helpful URLs:

Plugin Page: http://wordpress.org/extend/plugins/oauthphp/
Code Location: http://oauth.googlecode.com/svn/code/php/
Oauth Security Info: http://software-security.sans.org/blog/2011/03/07/oauth-authorization-attacks-secure-implementation
The Blog: http://hacks.mozilla.org/

Access Information:
How is the application accessed?
Is the application internal or publicly available?
If it is internal, what mechanism prevents non-members from accessing it?

The plugin is not accessed, rather used by other plugins to communicate with third-party oauth apps. The “plugin” is actually just a single PHP page that provides functions (such as key generation, etc) that can be called by the other applications.

Infrastructure and Backends:
What languages do the applications use?
What database language is used if applicable?
Are the running versions up to date?
What server is it running on?

It is a PHP page. The current version is 1.1.

Accounts and Passwords:
If the mechanism to prevent general access is a password, how is the signup process handled?
How is account information stored?
Are passwords properly stored within databases if applicable?

Is a password policy in place?
Are accounts locked-out after a number of invalid logins?
How long are session cookies stored?

There is no account information used to access the plugin. It is simply a page of functions to generate other keys used by other plugins and apps.

Third-Party Resources:
Are third-party resources used (i.e. JavaScript libraries, images, CSS, etc.)?
Can those resources be trusted / are they from reputable sources?
Is there a chance the resource could be compromised?
Is it possible to host the resources locally to mitigate risks?
Is a third-party responsible for storage of user data?

No third party resources are used by this plugin, but the plugin is used by other plugins to communicate with third party resources. By itself, it poses no threat. However, when it is used with other plugins, it needs to be evaluated in tandem.

Data Handling:
What kind of data is transferred between the user and the application?
Is this data generated by the user or generated automatically?
Can the data be trusted?
How is the data sent to the application (i.e. JSON format, plain-text, GET/POST, etc.)?
How is the data handled by the server/application?
Can the data be manipulated in transit?
What happens if the data is altered?
What is done with the data once it is received (i.e. stored in a database, displayed to users)?
Is any data storage done via cookies? If so, what kind of data is stored via this method?

The user does not interact with this application. All work is done in the background by other plugins which communicate with this plugin.

Data Sensitivity:
What kind of data is being stored and/or manipulated by the application?
Does this data need to be encrypted in transit? In storage?
What is the impact if this data is lost/stolen?

The data manipulated by this application can be sensitive depending on its use. The primary purpose is to generate keys for oauth authentication so it could be used to maliciously attack a third party resource (such as Mozilla's SurveyGizmo account). However, the security of oauth is handled by the third party resource (SurveyGizmo, for example) and this plugin merely provides the means to communicate with that resource.

If the data is intercepted or stolen, it could be possible for someone to login as our account on the third party resource. However, the main use of this plugin is done in the WordPress Admin console (via adding third party accounts), so that would need to be compromised first.

Application Administration:
Is there an administration console?
Can it be accessed publicly?
How is it secured if so?

Are correct methods used to prevent admin actions from being performed outside of the admin console (i.e. using CSRF tokens)?
Are there any configuration pages that should not be made public?

The only admin console is the plugin installation / edit page of WordPress which is handled by WordPress security.

Security Coding Flaws:
Have all user inputs been sanitized?
Do all URL variables pass through sanitization?
Is data from databases escaped properly?
Are CSRF tokens used to prevent POSTs from outside websites?
If a database is used, are protections against SQL injection in place?

This is a “background” plugin. It is used by the third party plugin to communicate with its own servers and authenticate the wordpress admin user to the third party account. There are no inputs.

Testing:
List all tests performed on the application

1. Checked for user inputs with outputs that could potentially enable XSS.

Results:
List the results of the above tests

1. No user inputs via POST, GET, etc. or output of data directly to page.

Meetings and Notes:

This plugin does very little on its own. It just adds a page of PHP functions that enable Oauth to work in tandem with a different plugin communicating to its third-party resource. The final security review will need to be conducted when this plugin is used with other plugins (for example, the SurveyGizmo plugin which prompted this review).