Mozilla Security Review

General Information

Project Name:	Artiss Code Embed (Simple Code Embed) WordPress Plugin
Security Reviewer:	Matt Fuller (mfuller)
Date of Review:	7/5/2012
Bugzilla Bug #:	771050

Background Information:

The authors of the hacks.mozilla.org blog have asked for the Simple Code Embed plugin (Artiss Code Embed) to be installed. The plugin allows code snippets (videos, IFRAMES, JavaScript, etc). to be saved and inserted easily into a post using shorthand such as %CODE1%.

This review will analyze the behavior of this plugin and possible security issues.

From the plugin home page:
Artiss Code Embed (formally Simple Code Embed) allows you to embed code - JavaScript and HTML primarily - in a post. This is incredibly useful for embedding video, etc, when required. It cannot be used for server side code, such as PHP.
Features include..

- Add HTML or JavaScript to posts or pages - particularly useful for embedding videos!
- Embed in widgets using the Widget Logic plugin
- Global embedding allows you set up some code in one post or page and then access it from another
- Modify the keywords or identifiers used for embedding the code to your own choice
- Search for embedding code via a simple search option
- Add a simple suffix to the embed code to convert videos to responsive output
- Embed an external script directly using just the URL
- Fully internationalized ready for translations.

Helpful URLs:

Plugin Homepage: http://wordpress.org/extend/plugins/simple-embed-code/
Bugzilla Bug: https://bugzilla.mozilla.org/show_bug.cgi?id=771050
Another Bugzilla Bug with more info: https://bugzilla.mozilla.org/show_bug.cgi?id=742146
Plugin official page: http://www.artiss.co.uk/code-embed

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?
Are there links to user-only resources displayed to non-users?
Are login pages secured with SSL over HTTPS?

The plugin is “accessed” by blog authors by simply adding a custom field with custom code to a post. Then, in the post itself, they add a reference such as %CODE1% and it inserts the custom code each time that reference is used. The plugin is only accessible internally (within the WP login) but renders information within publicly accessible posts.

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?

The plugin uses PHP and CSS. The current version is 2.0.1. All other questions apply to WP and are out of scope for this plugin.

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?
Are passwords 8 characters or greater?
Do passwords use both numbers and letters (and possibly symbols)?
Is there a blacklist of common passwords?
Do passwords expire after X days and require a reset?
Are invalid logins logged?
Is there a lockout after X invalid attempts?
Is the error message for lockout generic (does not include if user/pass is valid)?
How are password resets handled (i.e. email, security question, etc.)?
Do emails sent after signup/reset contain a session link? (should not)
Do email verification codes expire after one use or 8 hours?
Is password reuse limited/prevented?

N/A - all account information (such as blog admins and accounts) is handled by WordPress. This plugin does enable more restricted accounts to post privileged elements within a post but does not enable any other admin functionality.

Session Management:
How long are session cookies stored?
Are session tokens 128-bit or greater?
Is session ID token creation handled by the server (or cryptographically if locally)?
Do authenticated sessions timeout after 15 minutes?
Is the Secure Flag enabled for all set cookies?
Is the HTTP-Only flag used to disable malicious script access to the session ID?
Are new session ids created on login?
On logout, are session ids invalidated?

N/A

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?
Does the application connect with services like Facebook, Twitter, etc?

The plugin itself does not use third-party resources, but does allow blog authors to include scripts, videos, code, etc. from third-party resources. The security aspect of this is entirely handled by the authors. It is their decision to include code hosted on other servers, but this plugin merely enables that ability.

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 “data” is whatever code the blog author chooses to include in his/her post. The plugin only interprets it as code. Only logged-in authors can post code.

The data may come from a third-party (if a blog author chooses to include a script from a different server). We do not control this information, so it may be altered and we are entirely at the will of that server once the code is loaded. This aspect is handled by the blog authors. They should be instructed on the correct content types to include.

Uploaded Data:
Can the user upload data to the application?
Are extensions, file size, file type (not only based on extension), etc. checked?
Are files renamed upon upload?
Is the correct content-type used?

N/A

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?
Is secure/sensitive data sent over SSL?

The sensitivity is low - all data being displayed is posted publicly by the blog author.

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 admin console is within the WordPress admin console.

All config and other pages for the plugin are inaccessible (rather, do not display content) unless used within WP (prevents direct plugin page access).

Security Coding Flaws:
Have all user inputs been sanitized?
Is a maximum size for data (input or uploads) defined?
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?
Is validation done on the server side (not just client-side)?
Is output encoded in addition to sanitization of input?
Does the user ever send data to the OS level?
Are x-frame options sent to “deny” or “sameorigin”?
Is debug mode disabled?

The only “user inputs” are scripts / code inserted by the post authors. These should not be sanitized/escaped/encoded because doing so would break the functionality and defeat the purpose of this plugin.

There is a search page for blog authors to search for previously used code snippets. This page, although behind a login, is vulnerable to XSS.

http://site.com/wp-admin/admin.php?page=ace-search&suffix="/><script>alert(1);</script>

I submitted bug 771315 that blocks this.

Testing:
List all tests performed on the application

1. Attempted direct access to plugin pages
2. Checked for parameters via GET
3. Checked for CSRF issues
4. Reviewed plugin pages for code flaws
5. Attempted use of keywords via comments (%CODE1%)

Results:
List the results of the above tests

1. Blank - only accessible via admin console
2. See XSS issue above
3. No issues
4. No major issues besides #2.
5. N/A - only works in posts themselves - good

Meetings and Notes:

Contacted developer about XSS - David Artiss