Mozilla Security Review

General Information

Project Name:	WordPress Table Reloaded Plugin
Security Reviewer:	Matt Fuller (mfuller)
Date of Review:	6/28 - 6/29
Bugzilla Bug #:	721818

Background Information:

The plugin “Table Reloaded” has been requested to be added to the WordPress blog. Below is information taken from the plugin description page:

WP-Table Reloaded enables you to create and manage tables in your WP's admin area. No HTML knowledge is needed. A comfortable backend allows to easily edit table data. Tables can contain any type of data and additional JavaScript libraries can be used to extend it with features like sorting, pagination, filtering, and more. You can include the tables into your posts, on your pages or in text widgets by using a shortcode or a template tag function. Tables can be imported and exported from/to CSV, XML and HTML.

Helpful URLs:

Infrasec Review Bug: https://bugzilla.mozilla.org/show_bug.cgi?id=721818
WordPress Plugin Page: http://wordpress.org/extend/plugins/wp-table-reloaded/

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 accessed two ways: 1) the author of a post in WordPress creates a new table and then adds it to the blog via a special tag. 2) The viewers of the blog see the table rendered on the post page. The plugin admin console is only visible to post admins or authors and is hidden from the public via the WordPress login console.

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?

PHP, HTML, CSS, JavaScript

The current version (as of testing) is: 1.9.3 which is installed via WordPress’ “Add Plugin” page.

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 are no accounts besides the WordPress accounts.

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.

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 only data transferred is data entered by the post author. When the author uses a shortcode such as [table id=<ID> /], the table is called from the database.

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 would only be information published by a blog author. There is no private or sensitive data.

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 via the WordPress management page which requires a separate login.

CSRF is prevented using nonces.

All directory listings are hidden via an empty index.php but also the internal configuration files do not reveal sensitive data.

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?

The input from the blog author is NOT sanitized. This could be an issue if he or she is tricked into entering extraneous code in the tables.

Nonces prevent POSTs from outside the console.

Testing:
List all tests performed on the application

1. Test for third-party resource use via proxy
2. Investigate code for input parameters that could be used maliciously with direct page access (i.e. admin.php?input=script)
3. Check for sanitization of input
4. Can users call the shortcode tag to display a table in the comments ([table id=<ID> /])?
5. Tested sanitization of other parameters entered by post author (such as table name, description, etc.)

Results:
List the results of the above tests

1. No third party resource use
2. General tests passed (fuzzed all found parameters) with no alerts()
3. Does not sanitize input that the post author makes (i.e. if the author can be tricked into adding <script> into the table, it will be shown and executed for everyone).
4. No
5. Those are sanitized

Meetings and Notes:

In the answers above, some mention that data is not sanitized when coming from the post author. The table plugin (like WordPress itself) allows authors to add HTML and other elements to their posts. The only issue here would be a self-xss attack, but is not a large concern for this plugin because if an author can be tricked into pasting unknown code into the table plugin, they could just as easily paste it into the blog post itself. This is a non-issue.