Mozilla Security Review

General Information

Project Name:	University Recruiting WordPress Theme
Security Reviewer:	Matt Fuller (mfuller)
Date of Review:	9/10/2012
Bugzilla Bug #:	789841

Background Information:

A WordPress theme was developed to improve the layout/appearance of the university recruiting blog. It is based off the twentyeleven default theme and makes modifications using WordPress taxonomies.

Note: many of the categories below are N/A because they are handled by WordPress rather than the theme itself. For example, HTTPS is a site-controlled setting not affected by the theme.

Helpful URLs:

Blog: https://blog.mozilla.org/university/
Theme Code: https://github.com/mattbasta/moz-college-recruiting
Sample Site: http://wordpressdev.serverboy.net (note, this may change/disappear)

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?
If HTTPS is used, is Strict Transport Security set?

N/A to the theme.

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?

Theme uses PHP, everything else is N/A to the theme.

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 to the theme - theme does not allow comments or users, but if needed, handled by the WordPress admin page.

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 to the theme.

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?

Uses gravatar, Google AJAX API. Google AJAX API is not HTTPS (see notes). No third party resources control user data or interact with the blog in a sensitive nature. Uses a video from mozilla, http://videos-cdn.mozilla.net/serv/interns/Interns-It%20can%20be%20you-720p-MPEG-4.webm Should be HTTPS also (see notes)

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?

Blog data, posts, events, etc. The data is sent from the blog to the user and does not accept data from the user. No user-submitted data is used on the blog (except for data posted by blog admins).

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 to the theme - it doesn't accept data or handle it if it did.

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?

No sensitive data is collected.

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?

There is an admin console, which is handled by WordPress.

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?

N/A to the blog. All GET input is escaped properly by the blog.

Testing:
List all tests performed on the application

1. Checked for XSS entry points
2. Checked for HTTPS in third party data
3. Manual code review of pages
4. Any areas users can enter input?

Results:
List the results of the above tests

1. None, all data escaped properly
2. Not used for Google AJAX API.
3. Looks good
4. None

Meetings and Notes:

HTTPS for Google AJAX API is needed to prevent mixed content warning, talked to basta, will be implemented.

HTTPS for http://videos-cdn.mozilla.net/serv/interns/Interns-It%20can%20be%20you-720p-MPEG-4.webm should be implemented also.