Mozilla Security Review

General Information

Project Name:	Event Specific Signup Page on Mozilla.org
Security Reviewer:	Matt Fuller (mfuller)
Date of Review:	7/19/12
Bugzilla Bug #:	773044

Background Information:

In order to allow Mozilla Reps to track their signups, additional functionality was added to the contribute form that allows for a "callback URL" to be used to ping a reps site. When the form is loaded, the callback URL is used to send an empty POST to the URL which then increases a counter.

Helpful URLs:

Blocked Bug: https://bugzilla.mozilla.org/show_bug.cgi?id=765783
Pull Request with Changes: https://github.com/mozilla/bedrock/pull/183
Reps Code that Accepts POST: https://github.com/mozilla/remo/blob/master/remo/events/views.py#L92

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 app is accessed through a form at mozilla.org/contribute.
It is a public app.
There is no login and no need for HTTPS.

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?

Python, Django, HTML, CSS
MySQL backend is used for the storage of form data, but this review focuses on the event specific "ping" to the reps site.

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?

No accounts are on the form page. The reps accounts are handled elsewhere.

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?

No sessions. It’s just a simple form that saves the email address, comment, and what the user is interested in working on.

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?

No third-party resources besides a captcha box which loads with the form. It is not tied to the user data (of which email is the only piece) but used to prevent spam submissions.

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 an email address, a comment, and a “what I’d like to work on” description. It can not be trusted, although methods are in place to prevent submissions of invalid data. It can be manipulated in transit since SSL is not in use, although server side validations are done to ensure data is still valid.
Once received, the data is saved in a database for later use in email lists.

The app also takes a parameter via GET - “callback” which is a URL to ping when the form is submitted. This POST does not include any personal data (it’s empty) and is only used to increment a counter. It can not be trusted either, but a JS function only allows POSTs to one of a list of approved domains.

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?

No uploads are permitted.

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?

Emails and comments. Does not need to be encrypted.
No passwords or other 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?

No admin console related to this app.

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?

User input is sanitized. Never displayed back to user (XSS).
URL variable is only used in JS and carefully checked to ensure it’s a domain on a whitelist.
CSRF tokens are not used, but a captcha is to prevent spam attacks.
Input is validated on client and server side.
No output.
X-frame is not needed (okay to embed this)
Debug mode is disabled.
No commands to OS.

Testing:
List all tests performed on the application

1. Tested for XSS
2. Check for CSRF
3. X-Frame options?
4. Inject malformed javascript

Results:
List the results of the above tests

1. Input sanitized, never echoed back to user
2. Vulnerable, but captcha prevents
3. Not needed (not sensitive form)
4. Domain whitelist prevents

Meetings and Notes:

This application is very simple - it simply retrieves the URL parameter, and, if it matches a whitelist, sends an empty POST to the URL. That URL is a reps site which increments a counter when it receives a POST. The only concerns are that anyone can POST to that reps page and increment the counter, but this is not a security issue.