Stored XSS in "Update Status" section under "OrangeBuzz" via the GET/POST parameters `createPost[linkTitle]` and `createPost[linkAddress]` #1217

Closed   **cooliscool** opened this issue on 7 Apr · 9 comments

cooliscool commented on 7 Apr • edited

Environment details
OrangeHRM version: 4.10.1
OrangeHRM source: Release build from Sourceforge or Git clone
Platform: Ubuntu
PHP version: 7.3.33
Database and version: MariaDB 10.3
Web server: Apache 2.4.52

If applicable:
Browser: Firefox

Describe the bug
Insufficient input validation in Buzz - `addNewPost` API results in Stored Cross Site Scripting attack. An attacker - who is an authenticated user - can craft a malicious request causing malicious Javascript to execute in the browser of any other user. The malicious Javascript can trigger when a victim user visits the Buzz page.

To Reproduce

1. Authenticate to the user dashboard.
2. Visit 'Buzz' page
3. Collect the CSRF token from the HTML response. It can be found in an 'input' field with the id `createPost__csrf_token`. Example :

```
<input type="hidden" name="createPost[_csrf_token]" value="8d7f3ee80979a1360fb445a6ffa1ffec" id="createPost__csrf_token">
```

4. Collect the logged in user cookie `_orangehrm`
5. Fire the following POST request including the CSRF token and cookie obtained. Replace the Host header too :

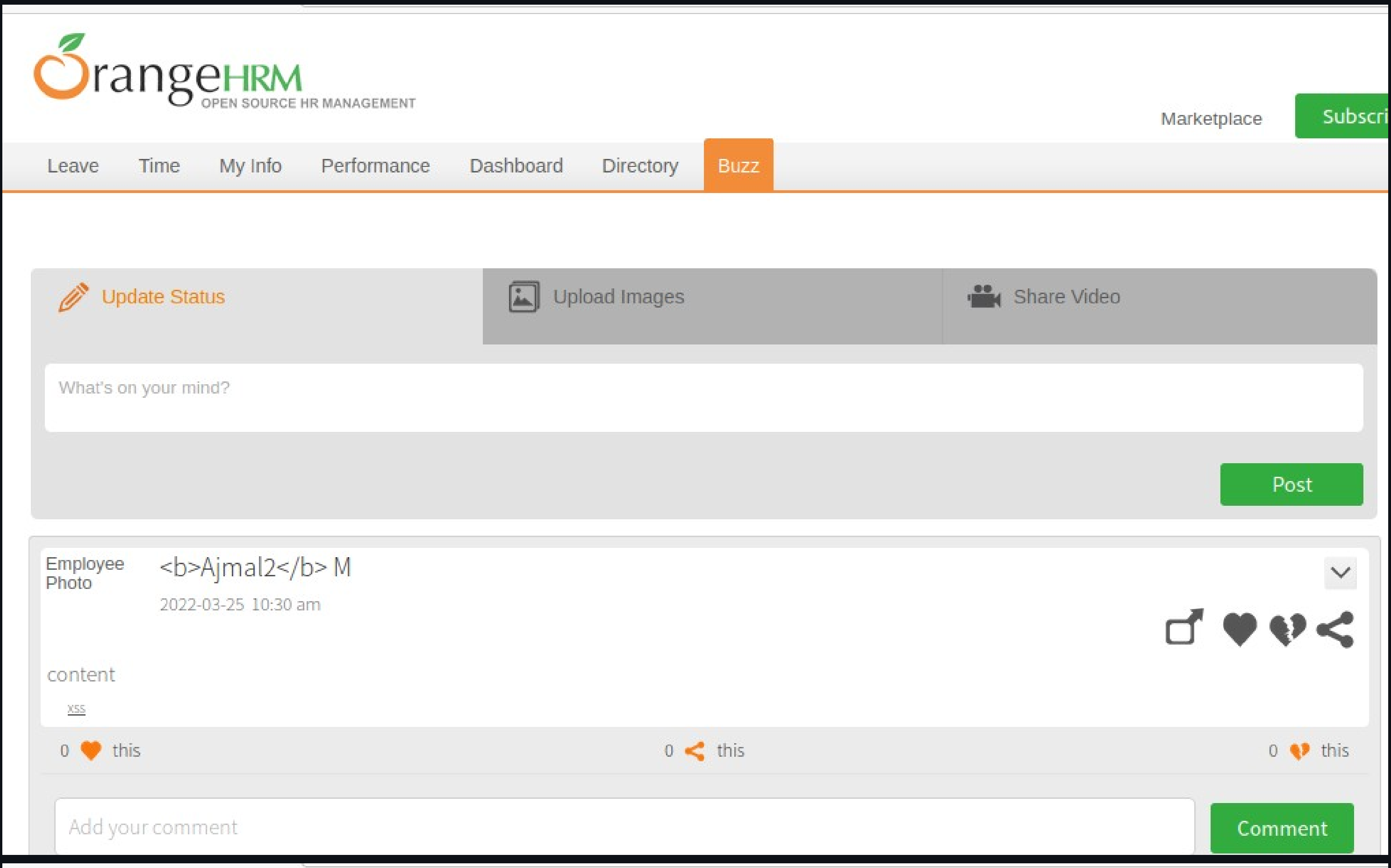
```
POST /symfony/web/index.php/buzz/addNewPost HTTP/1.1
Host: 1.2.3.44
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 173
Origin: http://54.165.4.147
Connection: close
Cookie: Loggedin=True; _orangehrm=qod7ejr1fqjtvtnm27i62ltcrb

createPost%5Bcontent%5D=content&createPost%5BlinkAddress%5D=javascript:alert(1221)&createPost%5BlinkTitle%5D=xss&createPost%5Bli
```

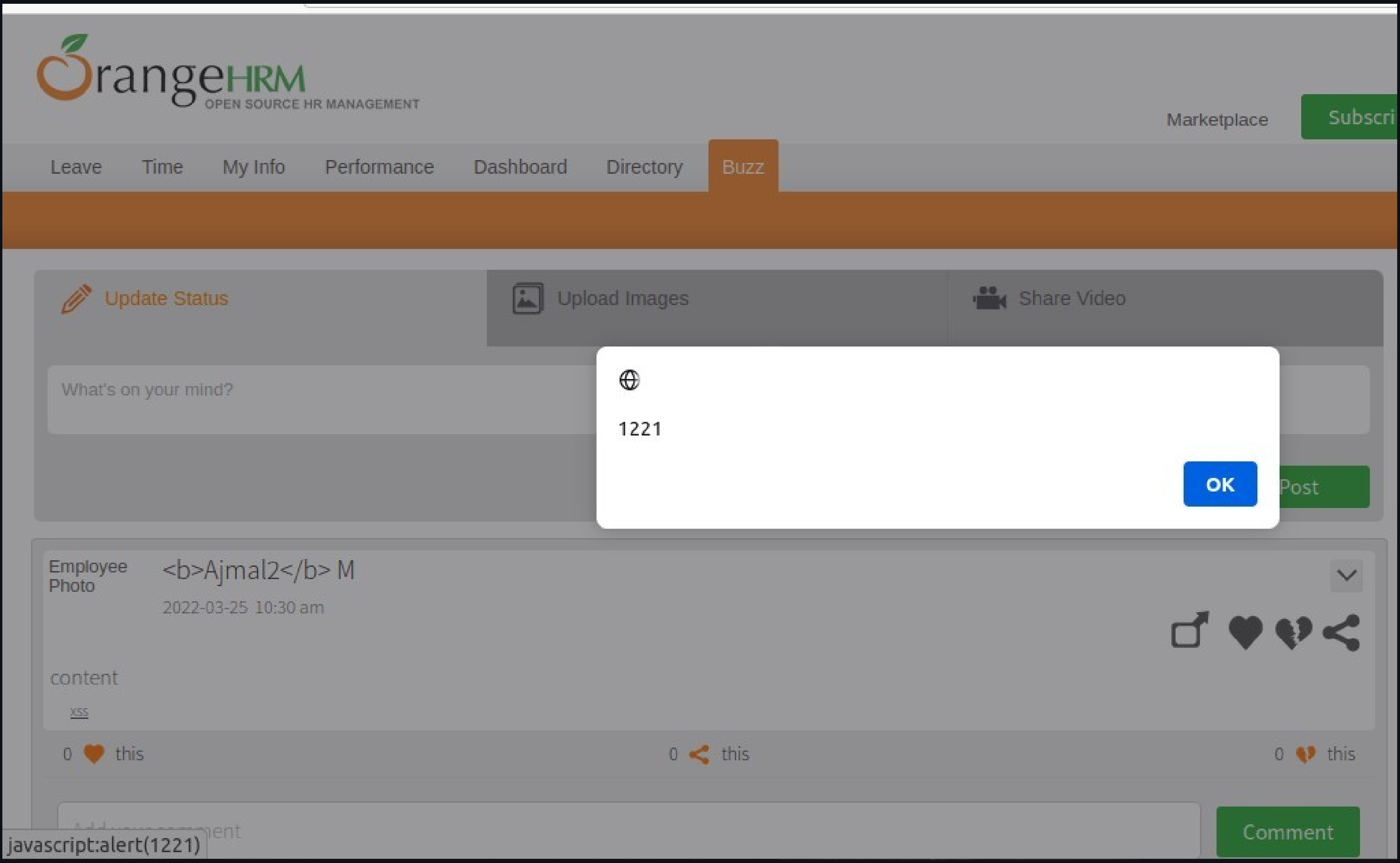
6. Click on the link 'xss' in the most recent post.

Expected behavior
The value of parameters `createPost[linkTitle]` and `createPost[linkAddress]` should be validated by API and an error should be thrown.

What do you see instead:
The malicious payload gets submitted successfully and get's stored in the posting made by the user.

Screenshots





Technical Details

A logged in user can post status updates to their buzz feed. From the front-end application a user will be able to post a text within a single field which says "What's on your mind" to the buzz feed. This happens via a POST request to the URL `/symfony/web/index.php/buzz/addNewPost` through the `createPost[content]` request body parameter. While investigating this API, we found that there are extra parameter fields in the body of this API which is not directly exposed through the frontend application.

The following request body parameters found in the API results certain profound effects in the HTML response sent by server:

1. `createPost[linkAddress]`

Causes the addition of an `<a>` anchor tag in response with id `linkTitle` & with attribute `src` with the value set for `createPost[linkAddress]` parameter.

2. `createPost[linkTitle]`

   Causes an `<a>` anchor tag in response with id `linkTitle` which is click able and displayed with the text content sent in the above request parameter.

Combining the above 2 parameters, it's possible to get an anchor HTML tag with a visible clickable text and a desired URL as src which is clickable.

The URL payload could be javascript as `javascript:alert(121)`. This can result in execution of arbitrary malicious javascript code on the client side if the victim clicks on this link.
The impact of this can be severe since this particular code gets stored in the database and gets delivered to the feed of every logged-in user in orangeHRM. Every user will have this delivered through their 'Buzz' feed.

In terms of impact, this vulnerability enables an attacker to stealing CSRF token and perform arbitrary actions on the website on behalf of the victim user.

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

4 participants