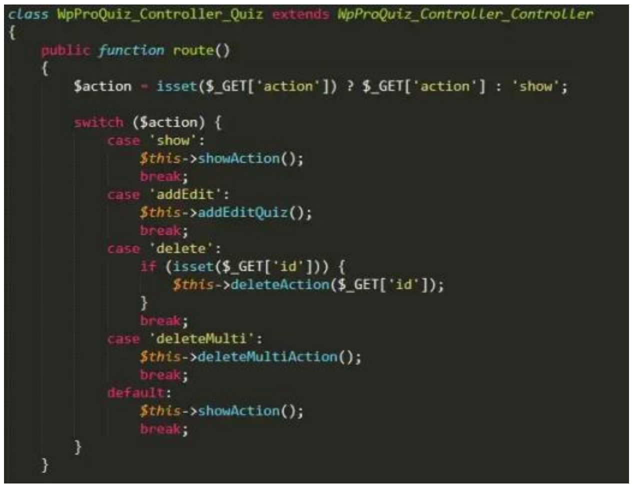
Hoan Hp Follow
Jun 8, 2020 · 2 min read · Listen

0-day story 1: wp-pro-quiz

Summary: I found a quite matter vulnerability in plugin wp-pro-quiz (version≤**0.37**). It can be downloaded here . Abusing this issue, an unauthenticated attacker can cheat the admin to delete any quiz on vulnerable website

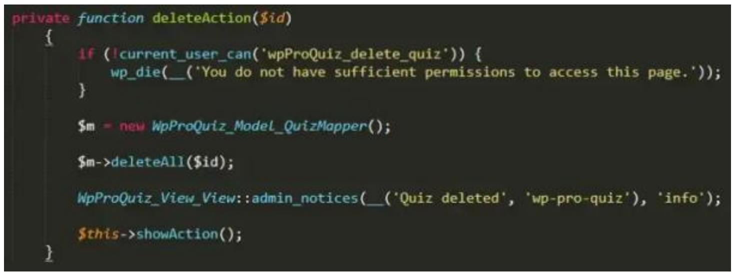
Analyze: CSRF in wp-pro-quiz

Look at this snippet of code at file wp-pro-quiz/lib/controller/WpProQuiz_Controller_Quiz.php



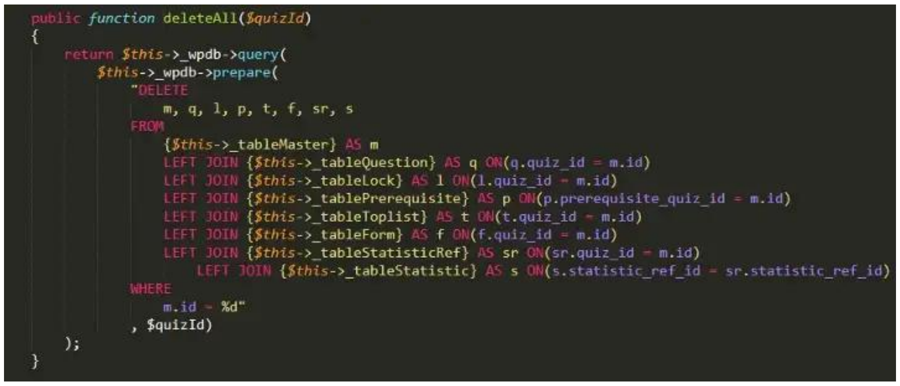
```
class WpProQuiz_Controller_Quiz extends WpProQuiz_Controller_Controller
{
    public function route()
    {
        $action = isset($_GET['action']) ? $_GET['action'] : 'show';

        switch ($action) {
            case 'show':
                $this->showAction();
                break;
            case 'addEdit':
                $this->addEditQuiz();
                break;
            case 'delete':
                if (isset($_GET['id'])) {
                    $this->deleteAction($_GET['id']);
                }
                break;
            case 'deleteMulti':
                $this->deleteMultiAction();
                break;
            default:
                $this->showAction();
                break;
        }
    }
```

We can see that $_GET['id'] is passed directly into deleteAction() which is implemented as below:



```
private function deleteAction($id)
    {
        if (!current_user_can('wpProQuiz_delete_quiz')) {
            wp_die(__('You do not have sufficient permissions to access this page.'));
        }

        $m = new WpProQuiz_Model_QuizMapper();

        $m->deleteAll($id);

        WpProQuiz_View_View::admin_notices(__('Quiz deleted', 'wp-pro-quiz'), 'info');

        $this->showAction();
    }
```

After check the user's permission, $id will be passed into deleteAll(). In here, application will call a SQL query to delete every quiz record having the ID.

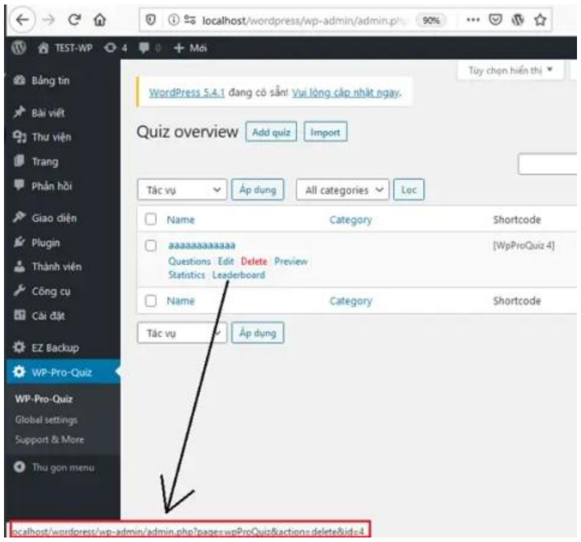


```
public function deleteAll($quizId)
{
    return $this->_wpdb->query(
        $this->_wpdb->prepare(
            "DELETE
                m, q, l, p, t, f, sr, s
            FROM
                {$this->_tableMaster} AS m
                LEFT JOIN {$this->_tableQuestion} AS q ON(q.quiz_id = m.id)
                LEFT JOIN {$this->_tableLock} AS l ON(l.quiz_id = m.id)
                LEFT JOIN {$this->_tablePrerequisite} AS p ON(p.prerequisite_quiz_id = m.id)
                LEFT JOIN {$this->_tableToplist} AS t ON(t.quiz_id = m.id)
                LEFT JOIN {$this->_tableForm} AS f ON(f.quiz_id = m.id)
                LEFT JOIN {$this->_tableStatisticRef} AS sr ON(sr.quiz_id = m.id)
                    LEFT JOIN {$this->_tableStatistic} AS s ON(s.statistic_ref_id = sr.statistic_ref_id)
            WHERE
                m.id = %d"
            , $quizId)
    );
}
```

So, it's easy to realize that there is no CSRF-token in entire the flow, a quiz 1 d easily just by one request. We had CSRF here!

Exploit:

- Attacker sends a link having format like: http://victim.com/wp-admin/admin.php?page=wpProQuiz&action=delete&id=4 to admin
- Admin somehow clicks the link, quiz with id 4 will be removed



WordPress 5.4.1 đang có sẵn! Vui lòng cập nhật ngay.

Name	Category	Shortcode
aaaaaaaaaaaa		[WpProQuiz 4]

localhost/wordpress/wp-admin/admin.php?page=wpProQuiz&action=delete&id=4

URI request to delete a quiz

Conclusion:

CSRF token is important, you should implement it for important tasks

Reference:

https://owasp.org/www-community/attacks/csrf