

main

...

[vulnerabilities](#) / CVE-2022-3383 && CVE-2022-3384.md

H4de5-7 Create CVE-2022-3383 &amp;&amp; CVE-2022-3384.md

[History](#)

1 contributor

84 lines (34 sloc) | 3.09 KB

...

## CVE-2022-3383 && CVE-2022-3384

Ultimate Member – User Profile, User Registration, Login & Membership Plugin <= 2.5.0 - Authenticated (Admin+) Remote Code Execution via Multi-Select

The sink function is `call_user_func()` in several PHP files.

I found several vulnerabilities, some allows attacker to execute any command he want remotely, however some only allow attacker to execute specific PHP functions, just like `phpinfo()`. The detail will be showed below.

### CVE-2022-3383

RCE vulnerabilities are caused when two parameters of `call_user_func()` are all controlled by attacker in `class-fields.php` and `class-form.php`

These callback functions are used to help users defined their own functions. However, the parameters are not filtered.

There are several sink functions :

- `get_option_value_from_callback()`, `get_options_from_callback()`, `edit_field()` in `class-fields.php`
- `ajax_select_options()` in `class-form.php`

Take `get_option_value_from_callback()` as an example:

```
function get_option_value_from_callback( $value, $data, $type ) {

    if ( in_array( $type, array( 'select', 'multiselect' ) ) && ! empty( $data['custom_dropdown_options_source'] ) ) {

        $has_custom_source = apply_filters( "um_has_dropdown_options_source_{$data['metakey']}", false );

        if ( $has_custom_source ) {

            $opts = apply_filters( "um_get_field_{$data['metakey']}", array() );
            $arr_options = $opts['options'];

        } elseif ( function_exists( $data['custom_dropdown_options_source'] ) ) {
            if ( isset( $data['parent_dropdown_relationship'] ) ) {
                $_POST['parent_option_name'] = $data['parent_dropdown_relationship'];
                $_POST['parent_option'] = um_user( $data['parent_dropdown_relationship'] );

                $arr_options = call_user_func( $data['custom_dropdown_options_source'], $data['parent_dropdown_relationship'] );
            } else {
                $arr_options = call_user_func( $data['custom_dropdown_options_source'] );
            }
        }
    }
}
```

`$data['custom_dropdown_options_source']` and `$data['parent_dropdown_relationship']` are not filtered in this function.

This function is called by `um_option_match_callback_view_field()` in `um-filters-fields.php`

This function does not filter `$data['custom_dropdown_options_source']` too

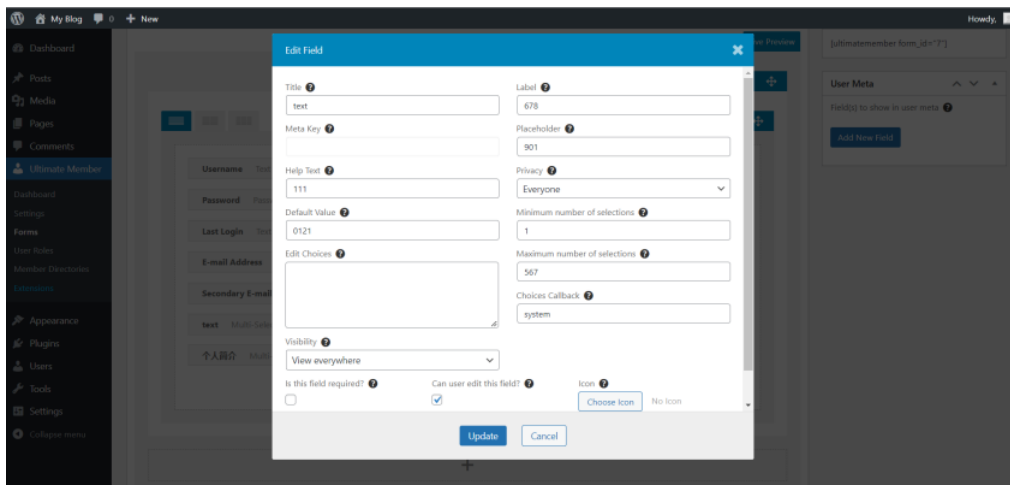
```
function um_option_match_callback_view_field( $value, $data ) {
    if ( ! empty( $data['custom_dropdown_options_source'] ) ) {
        return UM()->fields()->get_option_value_from_callback( $value, $data, $data['type'] );
    }

    return $value;
}

add_filter( 'um_profile_field_filter_hook__select', 'um_option_match_callback_view_field', 10, 2 );
add_filter( 'um_profile_field_filter_hook__multiselect', 'um_option_match_callback_view_field', 10, 2 );
add_filter( 'um_field_select_default_value', 'um_option_match_callback_view_field', 10, 2 );
add_filter( 'um_field_multiselect_default_value', 'um_option_match_callback_view_field', 10, 2 );
```

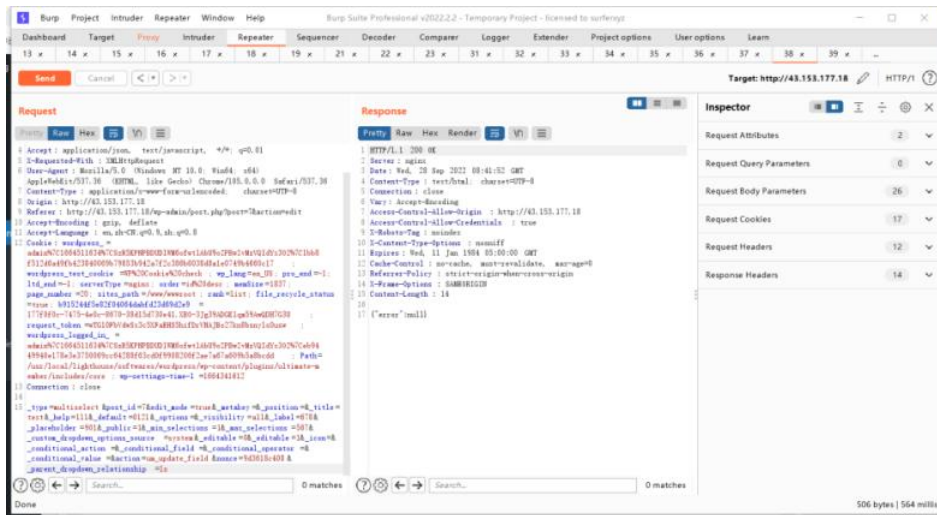
This function is called when processing select/multiselect field

Fields can be added here

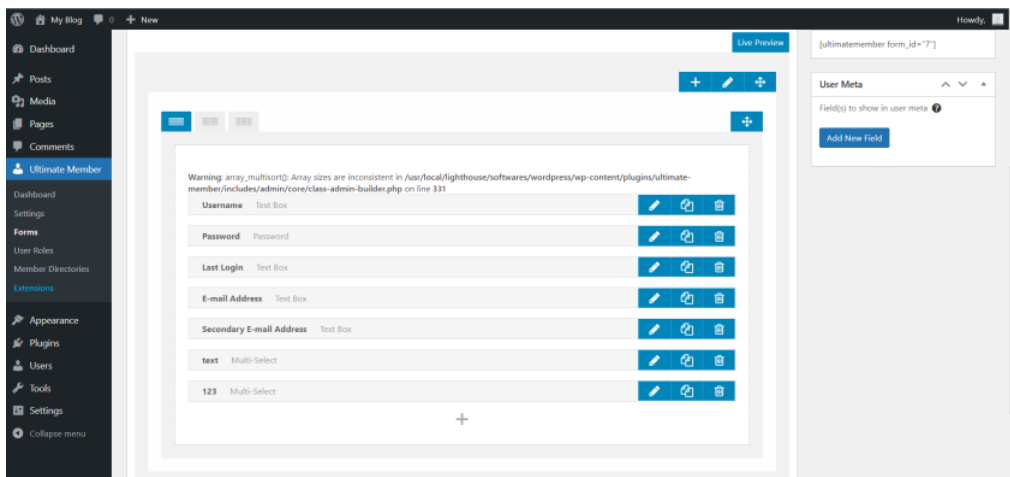


The core part is Choices Callback, which corresponds to `$data['custom_dropdown_options_source']`

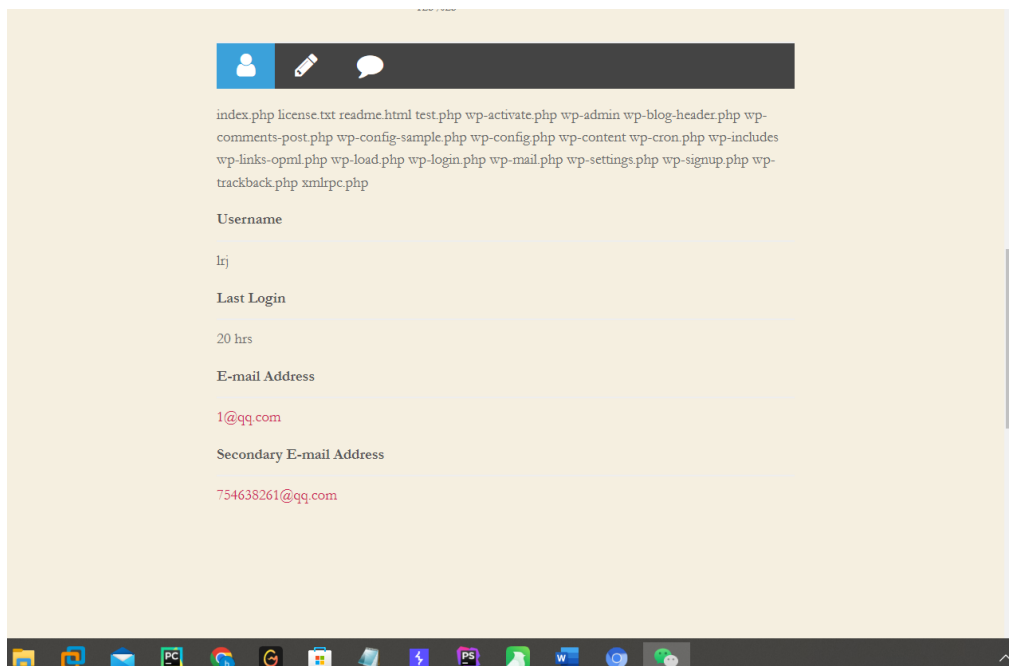
Then I capture the packet and add parameter `_parent_dropdown_relationship=ls`



Then I reload this page



And the result can be found in every users' profile



Other functions may be called in some other methods. I have not tested them. Thus, I just show how to use `get_option_value_from_callback()` to RCE here.

## CVE-2022-3384

Some `call_user_func()` in ultimate member only has one parameter, just like `populate_dropdown_options()` in `class-admin-builder.php`

```
function populate_dropdown_options() {
    UM()->admin()->check_ajax_nonce();

    if ( ! is_user_logged_in() || ! current_user_can( 'manage_options' ) ) {
        wp_send_json_error( __( 'This is not possible for security reasons.', 'ultimate-member' ) );
    }

    $arr_options = array();

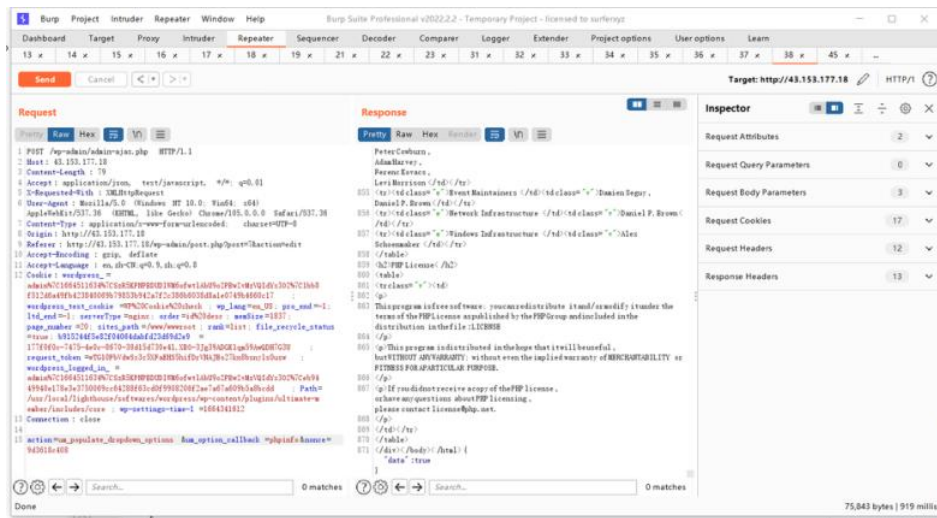
    // we can not use 'sanitize_key()' because it removes backslash needed for namespace and uppercase symbols
    $um_callback_func = sanitize_text_field( $_POST['um_option_callback'] );
    // removed added by sanitize slashes for the namespaces
    $um_callback_func = wp_unslash( $um_callback_func );

    if ( empty( $um_callback_func ) ) {
        $arr_options['status'] = 'empty';
        $arr_options['function_name'] = $um_callback_func;
        $arr_options['function_exists'] = function_exists( $um_callback_func );
    }

    $arr_options['data'] = array();
    if ( function_exists( $um_callback_func ) ) {
        $arr_options['data'] = call_user_func( $um_callback_func );
    }

    wp_send_json( $arr_options );
}
```

This function can only execute non-parameter PHP function, just like `phpinfo()`



And if attacker not add `_parent_dropdown_relationship` parameter in `get_option_value_from_callback()` he can only execute non-parameter PHP functions

I think some filters can be added to filter parameter to avoid the execution of unexpected functions