ANALYSIS OF A WORDPRESS PLUGIN EXPLOIT

I was reading ArsTechnica like I do every morning, and saw an article about how yet another popular WordPress plugin was found to have a remote execution vulnerability. The comments on the article were predictably bad and misinformed, so I decided to look into the security fix and see what caused the original issue (and how the exploit worked).

The plugin is Custom Contacts Form, which has over 670,000 downloads.

This bug is awful, catastrophic to sites that enable registration by untrusted users.

First, this bug has been in the plugin for at least 3 years, I didn’t feel like figuring out exactly when it cropped up though. 3 years is a long time.

This bug allows any visitor on your blog (they don’t even need to be logged in) to download an export file of your contact form. That alone could be very catastrophic depending on your site.

More importantly, this bug allows any authenticated user on your blog (of any privilege level) to execute arbitrary SQL commands. Let that sink in for a moment.

So, how did this bug come to be?

Looks like gross incompetence combined with a possible misunderstanding of the `is_admin` function. See, `is_admin` doesn't check to see if the current user is an admin, it checks if the current user is in the admin area (/wp-admin), which as most WordPress users know, any user can access (it's where the profile settings are). Even subscriber level users can access the admin area.

So let's take a look at the code that caused the issue:

```
if (!is_admin()) { /* is front */
    // ...
} else { /* is admin */
    // ...
    add_action('init', array(&$custom_contact_admin, 'adminInit'), 1);
    // ...
}
```

So seen above is a shortened code snipped from the main plugin file, that adds a hook to execute `adminInit` if the user is in the WordPress admin. Now lets look at that hook:

```
function adminInit() {
    $this->downloadExportFile();
    $this->downloadCSVExportFile();
    $this->runImport();
}
```

The above function executes a few other functions. This is already worrying based on function names. I'd expect adminInit to check if the current user had some specific capability or role first, but it doesn't. Maybe it still does in those functions?

```
function runImport() {
    if (isset($_POST['ccf_clear_import']) || isset($_POST['ccf_merge_import'])) {
        //chmod('modules/export/', 0777);
        ccf_utils::load_module('export/custom-contact-forms-export.php');
        $transit = new CustomContactFormsExport(parent::getAdminOptionsName());
        $settings['import_general_settings'] = ($_POST['ccf_import_overwrite_settings'] == 1) ? true : false;
        $settings['import_forms'] = ($_POST['ccf_import_forms'] == 1) ? true : false;
        $settings['import_fields'] = ($_POST['ccf_import_fields'] == 1) ? true : false;
        $settings['import_field_options'] = ($_POST['ccf_import_field_options'] == 1) ? true : false;
        $settings['import_styles'] = ($_POST['ccf_import_styles'] == 1) ? true : false;
        $settings['import_saved_submissions'] = ($_POST['ccf_import_saved_submissions'] == 1) ? true : false;
        $settings['mode'] = ($_POST['ccf_clear_import']) ? 'clear_import' : 'merge_import';
        $transit->importFromFile($_FILES['import_file'], $settings);
        ccf_utils::redirect('options-general.php?page=custom-contact-forms');
    }
}
```

The two download functions also don't check permissions, which is how an attacker can dump your contact entries.

Now in this `runImport` function, the important call is to `$transit->importFromFile`. It takes an uploaded file, and does something with it. Let’s take a look:

```php
function importFromFile($file, $settings = array('mode' => 'clear_import', 'import_general_settings' => false, 'import_forms' => true,'import_fields' => true, 'import_field_options' => true, 'import_styles' => true, 'import_saved_submissions' => false)) {
    $path = CCF_BASE_PATH. 'import/';
    $file_name = basename(time() . $file['name']);
    $file_extension = pathinfo($file['name'], PATHINFO_EXTENSION);
    if ( stripos( $file_extension, 'sql' ) ) {
        unlink( $file['tmp_name'] );
        wp_die( 'You can only import .sql files.' );
    }
    // ...
}
```

I’ve left out the bulk of the function, as you can probably see what it’s going to do. It takes a SQL file, and runs it. Since this function isn’t behind an authentication/capability/role check, that means anyone can upload any SQL file and run it….

So how would this have been avoided? A simple capability check is normally sufficient:

```php
if ( current_user_can( 'manage_options' ) ) {
    // Is a real admin
}
```

Source: http://brandonwamboldt.ca/analysis-of-a-wordpress-plugin-exploit-1538/