Missing CSRF Protection on Registration Approval Actions

Moderate **Fasse** published **GHSA-ph84-r98x-2j22** 4 days ago

Package

php **admidio/admidio** (Composer)

Affected versions	Patched versions
<= 5.0.6	5.0.8

Description

Summary

The create_user, assign_member, and assign_user action modes in modules/registration.php approve pending user registrations via GET request without validating a CSRF token. Unlike the delete_user mode in the same file (which correctly validates the token), these three approval actions read their parameters from $_GET and perform irreversible state changes without any protection. An attacker who has submitted a pending registration can extract their own user UUID from the registration confirmation email URL, then trick any user with the rol_approve_users right into visiting a crafted URL that automatically approves the registration. This bypasses the manual registration approval workflow entirely.

Details

CSRF Protection Is Present for delete_user but Absent for Approval Modes

File: modules/registration.php, lines 90-128

The delete_user mode validates the CSRF token (line 99), but the three approval modes do not:

```
// assign_member and assign_user: no CSRF check
} elseif (in_array($getMode, array('assign_member', 'assign_user'))) {
    $registrationService = new RegistrationService($gDb, $getUserUUID);
    $message = $registrationService->assignRegistration($getUserUUIDAssigned, $getMode =
```

```
    $gMessage->setForwardUrl($message['forwardUrl']);
    $gMessage->show($message['message']);

// create_user: no CSRF check
} elseif ($getMode === 'create_user') {
    $registrationUser->acceptRegistration();
    if ($gCurrentUser->isAdministratorRoles()) {
        admRedirect(SecurityUtils::encodeUrl(ADMIDIO_URL . FOLDER_MODULES.'/profile/role
            array('accept_registration' => true, 'user_uuid' => $getUserUUID)));
    }

// delete_user: CSRF IS validated
} elseif ($getMode === 'delete_user') {
    SecurityUtils::validateCsrfToken($_POST['adm_csrf_token']); // <-- protected
    $registrationUser->delete();
}
```

The three approval modes read both UUIDs exclusively from $_GET (lines 41-43):

The approve action modes accept $_GET parameters `user_uuid` and `user_uuid_assigned` without any POST body or CSRF token. Both parameters pass through `admFuncVariableIsValid()` with `uuid` type validation, which prevents SQL injection but provides no CSRF protection.

User UUID Is Known to the Attacker from Registration Email

File: `D:/bugcrowd/admidio/repo/src/Infrastructure/Service/RegistrationService.php`, lines 154-157

When a user submits a registration, Admidio sends a confirmation email containing a URL of the form:

```
https://TARGET/adm_program/modules/registration.php?
id=VALIDATION_ID&user_uuid=REGISTRANT_UUID
```

The `user_uuid` in this URL is the registrant's own UUID. The attacker has this UUID because they received the confirmation email for their own registration.

isAdministratorRegistration() Is a Delegated Right

File: `D:/bugcrowd/admidio/repo/src/Users/Entity/User.php`, lines 1603-1606

```
public function isAdministratorRegistration(): bool
{
    return $this->checkRolesRight('rol_approve_users');
}
```

The `rol_approve_users` right is a delegated organizational privilege, not full system administrator access. Any member designated to review registrations -- for example, a membership secretary or club administrator -- is a valid CSRF victim.

PoC

Scenario: Attacker bypasses manual registration approval

Prerequisites: (1) Manual registration approval is enabled. (2) The attacker submits a registration form and receives a confirmation email with their `user_uuid`. (3) After clicking the confirmation link, their registration enters the pending queue.

Step 1: Attacker extracts their own user_uuid from the registration email

The confirmation email contains a link of the form:

```
https://TARGET/adm_program/modules/registration.php?
id=VALIDATION_ID&user_uuid=ATTACKER_UUID
```

The `ATTACKER_UUID` is visible to the attacker from their own email.

Step 2: CSRF auto-approval via image tag

The attacker hosts a page that the victim (admin with `rol_approve_users` right) visits:

```
<img src="https://TARGET/adm_program/modules/registration.php?mode=create_user&use
```

When the victim loads this page, Admidio silently accepts the attacker registration and assigns default organization roles. No confirmation or token is required.

Step 3: Force-assign registration to an existing account (account takeover)

If the attacker knows the UUID of an existing member (obtainable from profile page URLs when the user list is visible) and has a pending registration:

```
<img src="https://TARGET/adm_program/modules/registration.php?mode=assign_user&use
```

This merges the pending registration into the existing account, replacing that account login credentials with the attacker credentials.

Impact

- **Manual Approval Bypass:** An attacker with a pending registration can force auto-approval without waiting for an administrator to manually review it. This grants them organization membership, including access to events, documents, mailing lists, and other role-restricted features.
- **Account Takeover via assign_user CSRF:** If the attacker knows any member UUID (visible in profile page URLs), the `assign_user` mode merges the attacker registration into that member

account, replacing the existing member login with the attacker credentials. This is a full account takeover requiring only that the victim admin visit a crafted URL.

- **Low Attack Complexity:** The attacker only needs their own registration email to get their UUID. The CSRF payload is a plain GET request via an image tag -- no JavaScript required.
- **Delegated Right:** The required victim right (`rol_approve_users`) is a common delegation target in organizations with membership approval workflows.

Recommended Fix

Add `SecurityUtils::validateCsrfToken($_POST["adm_csrf_token"])` at the beginning of each approval action, consistent with how `delete_user` is already protected in the same file.

```
// File: modules/registration.php

} elseif (in_array($getMode, array('assign_member', 'assign_user'))) {
    // ADD: validate CSRF token
    SecurityUtils::validateCsrfToken($_POST['adm_csrf_token']);
    $registrationService = new RegistrationService($gDb, $getUserUUID);
    $message = $registrationService->assignRegistration($getUserUUIDAssigned, $getMode =
    ...

} elseif ($getMode === 'create_user') {
    // ADD: validate CSRF token
    SecurityUtils::validateCsrfToken($_POST['adm_csrf_token']);
    $registrationUser->acceptRegistration();
    ...

} elseif ($getMode === 'delete_user') {
    SecurityUtils::validateCsrfToken($_POST['adm_csrf_token']); // already protected
    $registrationUser->delete();
}
```

Additionally, convert the approval action URLs from GET-based links to POST-form buttons (with the CSRF token in a hidden field). The existing `delete_user` button uses `callUrlHideElement()` which already sends the token in the POST body -- use the same pattern for approval buttons.

Severity

Moderate 4.5 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	High
User interaction	Required

Scope	Unchanged
Confidentiality	None
Integrity	High
Availability	None

Learn more about base metrics

CVSS:3.1/AV:N/AC:L/PR:H/UI:R/S:U/C:N/I:H/A:N

CVE ID

CVE-2026-34384

Weaknesses

▶ CWE-352

Credits

 **offset** Reporter