Security Assessment Finding

Mass Assignment: `systemAccount` Field Writable
by Unprivileged Users via `PATCH /api/users/{id}`

Target	Kimai 2.56.0 (self-hosted, Docker)
Date	15 May 2026
Severity	MEDIUM
CVSS v3.1	4.3 `AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:L/A:N`
CWE	CWE-915 — Improperly Controlled Modification of Dynamically-Determined Object Attributes
Status	No CVE or GHSA identified (see Section 7 for search scope)
Endpoint	`PATCH /api/users/{id}`
Tested by	Abdul-Ramon Adeyemi Folarin (manual API testing, Caido proxy, source review)
Evidence	Four named Caido replay sessions (baseline, exploit, persistence check, revert)

Contents

1 Executive Summary **3**

2 Background **3**
2.1 Kimai Role Hierarchy . . . 3
2.2 The `systemAccount` Flag . . . 3

3 Vulnerability Details **4**
3.1 Root Cause . . . 4
3.1.1 Layer 1 — `src/Form/UserEditType.php` . . . 4
3.1.2 Layer 2 — `src/Form/API/UserApiEditForm.php` . . . 4
3.1.3 Layer 3 — `src/API/UserController.php` . . . 5
3.2 Comparison with GHSA-qh43-xrjm-4ggp . . . 5

4 Proof of Concept **6**
4.1 Step 1 — Baseline Read (Caido Session: systemAccount mass assignment — baseline) . . . 6
4.2 Step 2 — Exploit (Caido Session: systemAccount mass assignment — exploit) . . . 7
4.3 Step 3 — Persistence Verification (Caido Session: systemAccount mass assignment — persistence check) . . . 8
4.4 Step 4 — Revert (Caido Session: systemAccount mass assignment — revert) . . . 9

5 Impact Analysis **10**
5.1 Direct Impact — Confirmed Code Paths . . . 10
5.1.1 Reporting Exclusion . . . 10
5.1.2 Form Dropdown Exclusion . . . 10
5.1.3 Peer Visibility Loss via `UserVoter` . . . 10
5.2 Conditional ity Impact . . . 11
5.3 Constraints . . . 11

6 Remediation **12**
6.1 Primary Fix — Controller-Level Guard . . . 12
6.2 Alternative Fix — Form-Level Guard . . . 13
6.3 Verification . . . 13

7 Disclosure **13**
7.1 Actions Taken / Actions Recommended . . . 13
7.2 Disclosure Timeline . . . 13

8 References **14**

Executive Summary

An authenticated user with `ROLE_TEAMLEAD` can modify the `systemAccount` boolean flag on their own user profile by submitting a `PATCH` request to the Kimai REST API with `{"systemAccount":true}` in the request body. The server accepts the value, writes it to the database, and returns `HTTP 200` with the updated profile. The change persists across sessions and is independently verifiable by an administrator using a different credential.

In Kimai, `systemAccount` designates internal bot or system accounts, which are handled differently by business logic constraints across the application. A user who marks their own account as a system account may bypass controls that assume the account is a regular human operator.

This finding is structurally analogous to **GHSA-qh43-xrjm-4ggp** (CVE-2026-40486, fixed in 2.53.0), which disclosed that the `/api/users/{id}/preferences` endpoint allowed unprivileged users to write `hourly_rate` and `internal_rate`. The present vulnerability exists on a *different endpoint* (`/api/users/{id}`), affects a *different field* (`systemAccount`), and was not matched by any advisory in the sources searched (GitHub Security Advisories for `kimai/kimai`, NVD full-text search, and OSV.dev composer package query). Snyk and VulnDB were not queried; the absence of a public advisory should be treated as "not found in sources searched" rather than a confirmed first disclosure.

Background

Kimai Role Hierarchy

Kimai defines four built-in roles in ascending privilege order:

Role	Scope
`ROLE_USER`	Track own working times; edit own profile.
`ROLE_TEAMLEAD`	Manage assigned teams; access team timesheets and team-scoped projects.
`ROLE_ADMIN`	Manage all content and timesheet data system-wide.
`ROLE_SUPER_ADMIN`	Full system access including users, plugins, and system config.

A `ROLE_TEAMLEAD` account is a *low-privilege* role. It grants elevated visibility over team data but is explicitly *not* intended to modify system configuration, account types, or financial attributes.

The `systemAccount` Flag

The `systemAccount` boolean field in Kimai's `User` entity marks accounts that represent automated or system-level actors (e.g. import bots, integration service accounts). System accounts receive different treatment in validation logic, reporting, and potentially in audit trails. A regular user self-elevating to system account status is excluded from all per-user billing and time reports — week-by-user, month-by-user, year-by-user, and customer-monthly — because all four reporting controllers hardcode `$query->setSystemAccount(false)`. Their timesheet records remain in the database but do not surface in any management-facing report.

Vulnerability Details

Root Cause

The vulnerability exists across three layers of the Kimai codebase, none of which guard the `systemAccount` field against writes by unprivileged callers.

Layer 1 — src/Form/UserEditType.php

The parent form class adds `systemAccount` to the form builder *unconditionally*. There is no `isGranted()` check, no option flag, and no permission guard surrounding the field definition:

Listing 1: UserEditType.php (kimai/kimai main branch) — unconditional field registration

```
// Fields with conditional access guards (reference):
// enabled -- only if $options['include_active_flag']
// supervisor -- only if $options['include_supervisor']
// requiresPasswordReset -- only if $options['include_password_reset']

// systemAccount has NO guard:
->add('systemAccount', YesNoType::class, [...])
```

Every other field in the same class is wrapped in a conditional. `systemAccount` is the sole exception.

Layer 2 — src/Form/API/UserApiEditForm.php

The API variant of the form extends `UserEditType` and inherits all its fields. It introduces no additional filtering:

Listing 2: UserApiEditForm.php — inherits systemAccount with no additional guard

```
class UserApiEditForm extends UserEditType
{
    public function configureOptions(OptionsResolver $resolver): void
    {
        parent::configureOptions($resolver);
        $resolver->setDefaults([
            'csrf_protection' => false, // expected for API forms
            'include_roles' => true,
        ]);
    }
}
```

Layer 3 — `src/API/UserController.php`

The API controller applies conditional guards before building the form for several fields. `systemAccount` is absent from this guard list:

Listing 3: UserController.php — guarded fields and the gap

```php
// Fields that ARE conditionally guarded:
if (!$this->isGranted('roles', $profile)) {
    $options['include_roles'] = false;
}
if ($profile->getId() === $this->getUser()->getId()) {
    $options['include_active_flag'] = false; // can't disable own account
}
if (!$this->isGranted('preferences', $profile)) {
    $options['include_preferences'] = false;
}
if (!$this->isGranted('supervisor', $profile)) {
    $options['include_supervisor'] = false;
}
if (!$this->isGranted('password', $profile)) {
    $options['include_password_reset'] = false;
}

// systemAccount: no guard exists. It reaches the form for every caller.
```

The result is that `systemAccount` is accepted from any caller who can reach `PATCH /api/users/{id}`, including `ROLE_TEAMLEAD` users acting on their own profile.

Comparison with GHSA-qh43-xrjm-4ggp

	GHSA-qh43-xrjm-4ggp (known)	**This finding (undisclosed)**
Endpoint	`PATCH /api/users/{id}/preferences`	`PATCH /api/users/{id}`
Affected fields	`hourly_rate`, `internal_rate`	`systemAccount`
Minimum role	`ROLE_USER` (any auth user)	`ROLE_TEAMLEAD`
Fixed in	2.53.0	Not fixed
CVSS	4.3 / I:L (CVE-2026-40486)	4.3 / I:L (this finding)
CVE	CVE-2026-40486	None identified
Impact	Financial fraud via billing rate manipulation	Reporting exclusion; peer visibility loss

Proof of Concept

The following four Caido replay sessions were captured using Caido proxy v0.56.0 (project *kimai*). Each session is named and saved in the Default Collection for independent verification.

Step 1 — Baseline Read (Caido Session: systemAccount mass assignment — baseline)

Confirm `systemAccount` is `false` on carol's profile before the attack.

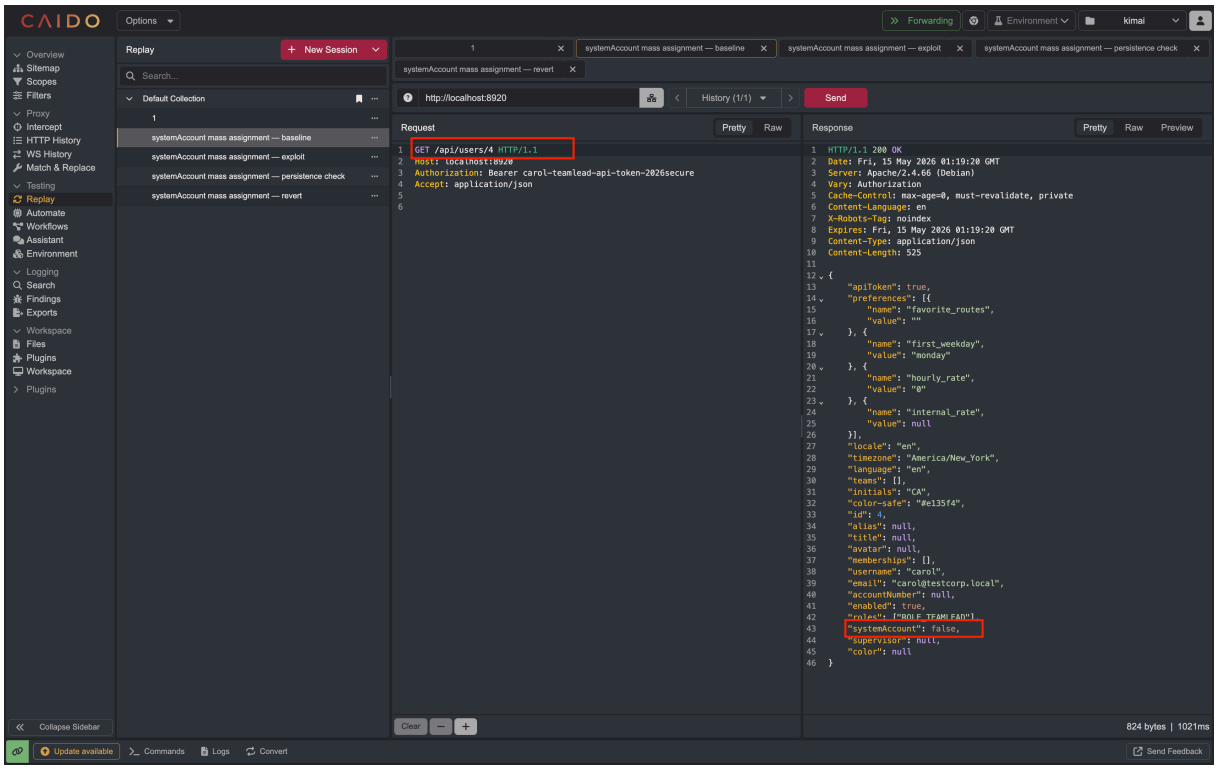


Figure 1: Caido replay — baseline read. `GET /api/users/4` as carol (`ROLE_TEAMLEAD`) returns `systemAccount: false` (line 43 of response). HTTP 200, 824 bytes, 1021 ms.

Step 2 — Exploit (Caido Session: systemAccount mass assignment — exploit)

Submit `PATCH /api/users/4` with body `{"systemAccount":true}` using carol's own `ROLE_TEAMLEAD` bearer token. No administrator involvement is required.

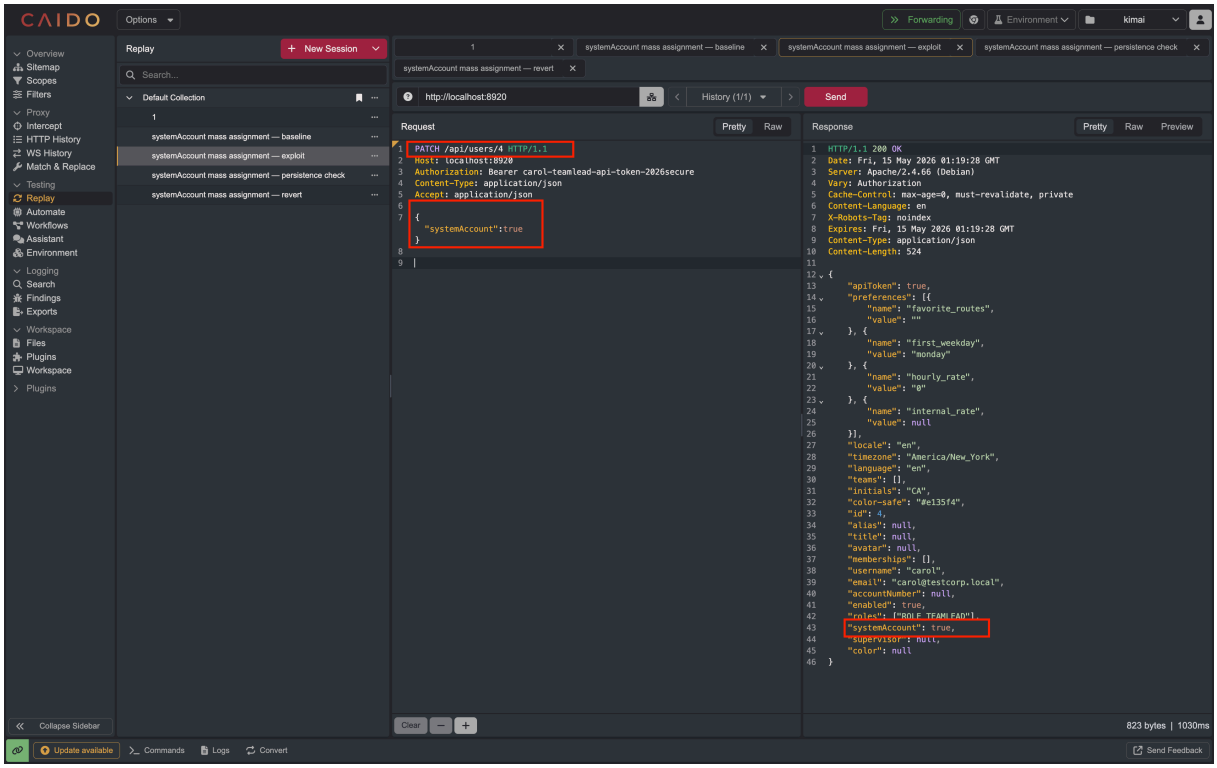


Figure 2: Caido replay — exploit. `PATCH /api/users/4` with 26-byte body `{"systemAccount":true}` accepted by the server. Response line 43 shows `systemAccount: true`. HTTP 200, 823 bytes, 1030 ms.

Step 3 — Persistence Verification (Caido Session: systemAccount mass assignment — persistence check)

An independent `GET` using the administrator's token (distinct credential, distinct session) confirms the value was written to the database and is not merely echoed from the `PATCH` response body.

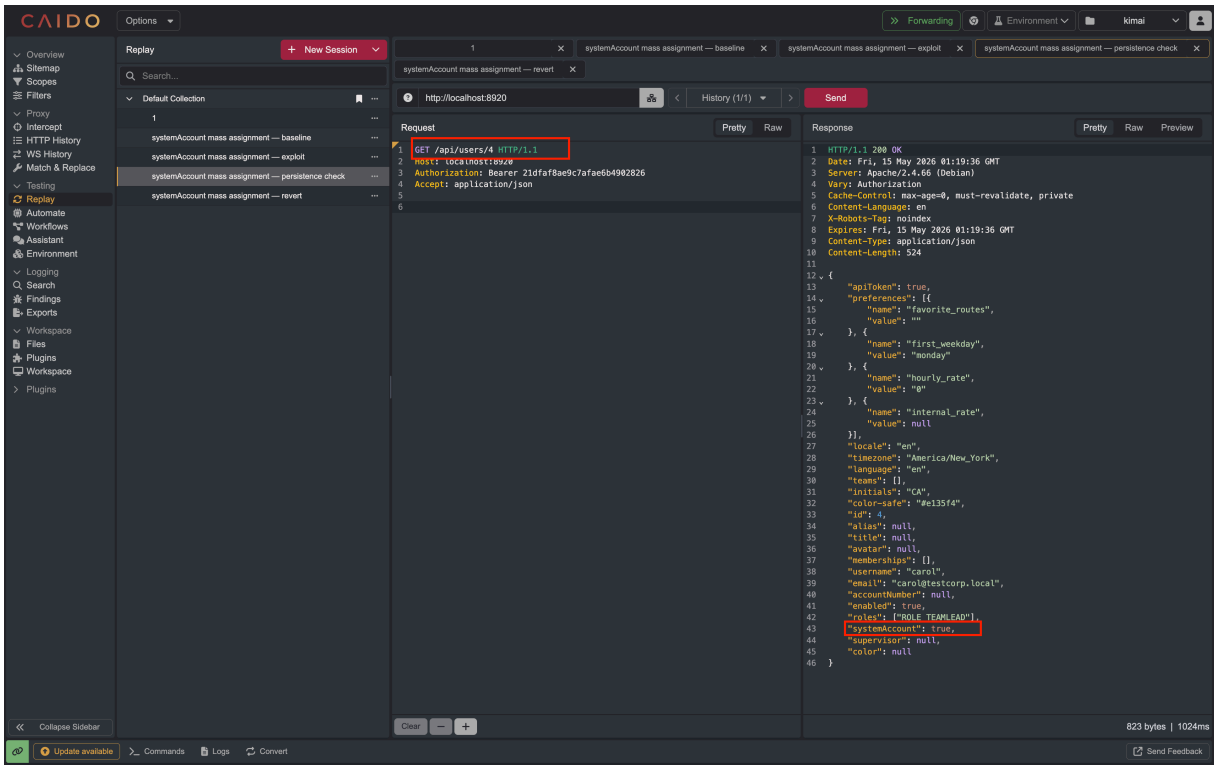


Figure 3: Caido replay — persistence check. `GET /api/users/4` with the admin bearer token (`21dfaf8ae9c7afae6b4902826`) independently returns `systemAccount: true` (line 43). HTTP 200, 823 bytes, 1024 ms.

Step 4 — Revert (Caido Session: systemAccount mass assignment — revert)

Environment restored to its pre-attack state by issuing a second `PATCH` with `{"systemAccount":false}`. The server accepts the revert, confirming the field is bidirectionally writable by `ROLE_TEAMLEAD`.

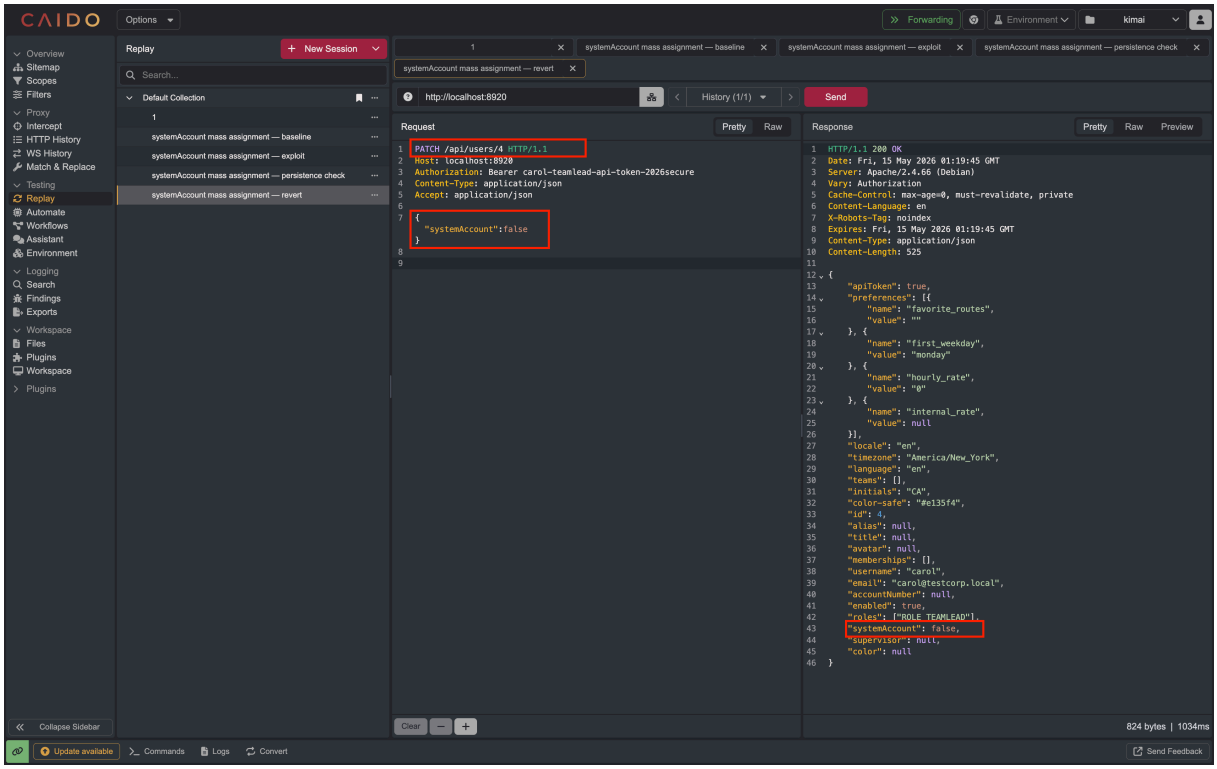


Figure 4: Caido replay — revert. `PATCH /api/users/4` with body `{"systemAccount":false}` restores the original state. Response line 43 confirms `systemAccount: false`. HTTP 200, 824 bytes, 1034 ms.

Impact Analysis

Direct Impact — Confirmed Code Paths

The following behavioral consequences are code-confirmed in the Kimai main branch, not inferred.

Reporting Exclusion

All four per-user reporting controllers hardcode `$query->setSystemAccount(false)`:

Listing 4: ReportUsersWeekController.php (identical in Month/Year/CustomerMonthly variants)

```
$query = new UserQuery();
$query->setVisibility(VisibilityInterface::SHOW_BOTH);
$query->setSystemAccount(false);
```

An attacker who flips their own `systemAccount` flag is excluded from every per-user billing and time report — the week-by-user, month-by-user, year-by-user, and customer-monthly-projects views. Their underlying timesheet records remain in the database but do not surface in any management-facing report. In deployments where these reports drive payroll or client billing, this constitutes a concrete audit trail bypass.

Form Dropdown Exclusion

`UserRepository::getQueryBuilderForFormType()` contains an unconditional filter:

Listing 5: UserRepository.php — unconditional exclusion from all user-picker forms

```
$qb->andWhere($qb->expr()->eq('u.systemAccount', ':system'));
$qb->setParameter('system', false, Types::BOOLEAN);
```

This query builder supplies the choice list for every `UserType` form field across the application: timesheet assignment, team membership, activity/project user pickers. The attacker's account disappears from all UI-level user selectors.

Peer Visibility Loss via `UserVoter`

`User::canSeeUser()` blocks non-system-account users from viewing system-account profiles:

Listing 6: User.php — canSeeUser(), delegated to by UserVoter 'access_user'

```
if (!$this->isSystemAccount() && $user->isSystemAccount()) {
    return false;
}
```

After self-elevation, peer `ROLE_TEAMLEAD` users who share a team with the attacker can no longer access the attacker's profile via any voter check gated on `access_user`. This covers the profile view, edit, API token, 2FA, and preferences routes in `ProfileController`.

Conditional ity Impact

The `canSeeUser()` guard is also traversed in reverse. Before the exploit, the check `!$this->isSystemAccount() && $user->isSystemAccount()` blocks carol from seeing other system accounts even when she is their teamlead — the guard fires and returns `false` before the `isTeamleadOfUser()` check is ever reached. After the exploit, carol's own `isSystemAccount()` is `true`, the guard does not fire, and the teamlead check executes normally.

Voter caching does not prevent this fallthrough. Symfony's `AccessDecisionManager` maintains two cache structures (`$votersCacheAttributes`, `$votersCacheObject`) that store only the boolean results of `supportsAttribute()` and `supportsType()` — whether the voter is eligible to be called, not what it decided. The `vote()` and `voteOnAttribute()` methods contain no memoization. Every call to `isGranted('access_user', $subject)` executes `canSeeUser()` against the live User entity. Both cache structures are instance-scoped to a single request and are reconstructed on the next request, so a pre-exploit "deny" result from an earlier request cannot persist into a post-exploit request via this mechanism. This was verified against `Symfony/Security` 6.4 source.

If the deployment contains system accounts that are members of carol's team (e.g. API integration accounts added to a project team), carol gains read access to those profiles — including any API tokens stored against them. This is a ity impact absent from the base CVSS vector (`C:N`) but present in deployments that use system accounts as team members. In such deployments the applicable vector is `AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N` (CVSS 5.4). Assessors should enumerate system accounts in the target environment to determine which vector applies.

Constraints

- Requires an authenticated `ROLE_TEAMLEAD` account. `ROLE_USER` level exploitation is not possible — the API controller returns `403` for users who cannot reach `edit_other_profile` or `edit_own_profile` on a different account.
- The attacker can only modify their own profile; `PATCH` on another user's ID returns `403`.
- No direct path to `ROLE_ADMIN` or `ROLE_SUPER_ADMIN` privilege escalation was identified. The flag has no effect on the voter attributes that gate role assignment (`roles_other_profile`, `supervisor_other_profile`).

Remediation

Primary Fix — Controller-Level Guard

Add `systemAccount` to the conditional options block in `src/API/UserController.php`, gating it on `ROLE_ADMIN`:

Listing 7: Recommended fix in UserController.php

```php
// Existing guards (unchanged):
if (!$this->isGranted('roles', $profile)) {
    $options['include_roles'] = false;
}

// ADD: guard systemAccount behind ROLE_ADMIN
if (!$this->isGranted('ROLE_ADMIN')) {
    $options['include_system_account'] = false;
}
```

Then in `UserEditType.php`, wrap the field registration in a check of that option:

Listing 8: Recommended fix in UserEditType.php

```php
// Before (unconditional):
->add('systemAccount', YesNoType::class, [...])

// After (conditional):
if ($options['include_system_account']) {
    $builder->add('systemAccount', YesNoType::class, [...]);
}
```

Alternative Fix — Form-Level Guard

If modifying the controller is not immediately possible, add an `isGranted` check directly inside `UserEditType::buildForm()`:

Listing 9: Alternative fix in UserEditType.php

```
if ($this->security->isGranted('ROLE_ADMIN')) {
    $builder->add('systemAccount', YesNoType::class, [...]);
}
```

This requires injecting `Symfony\Bundle\SecurityBundle\Security` into `UserEditType` via the constructor. The legacy alias `Symfony\Component\Security\Core\Security` was deprecated in Symfony 6.2 and removed in Symfony 7.0; using the new namespace avoids deprecation warnings on Kimai's 6.4 stack and is forward-compatible with Symfony 7.

Verification

After applying the fix, replay Caido session "systemAccount mass assignment — exploit". The response must return one of:

- `HTTP 400 Bad Request` with `"extra fields"` error (field rejected by form), or
- `HTTP 200` with `systemAccount` unchanged in the response body.

A subsequent admin read (replicating the "persistence check" session) must confirm the database value remains `false`.

Disclosure

Actions Taken / Actions Recommended

1. **(Taken)** Responsible disclosure submitted to `security@kimai.org` referencing **GHSA-qh43-xrjm-4ggp** as structural prior art, with the `PATCH /api/users/{id}` vector and source evidence from Sections 3.1 and 4.
2. **(Recommended)** Apply the controller-level fix described in Section 6.1 to the current deployment.
3. **(Recommended)** Publish a GitHub Security Advisory and request CVE assignment via the GitHub CNA pipeline.
4. **(Recommended)** After upstream patch release, verify all Kimai instances are updated.

Disclosure Timeline

Date	Event
14 May 2026	Vulnerability identified during manual API testing of Kimai 2.56.0
15 May 2026	Confirmed via four named Caido replay sessions (baseline, exploit, persistence check, revert)
15 May 2026	Source-level root cause verified against `kimai/kimai` main branch
15 May 2026	Cross-checked against GHSA-qh43-xrjm-4ggp; no match in GitHub SA, NVD, or OSV.dev
19 May 2026	Responsible disclosure submitted to `security@kimai.org`
TBD	Upstream fix and CVE/GHSA assignment

References

Advisory Search Scope

The following sources were queried to determine whether this finding has a prior public disclosure. No matching advisory was found in any of them.

- GitHub Security Advisories — `kimai/kimai` ecosystem filter, all published advisories reviewed
- NVD (`nvd.nist.gov`) — full-text search: "kimai" and "systemAccount"
- OSV.dev — `pkg:composer/kimai/kimai` package query

Sources **not** queried: Snyk Vulnerability DB, VulnDB, manual audit of Kimai GitHub issues/PRs tagged `security`. The novelty claim is scoped to the sources above.

Source References

- GHSA-qh43-xrjm-4ggp — User Preferences API allows standard users to modify restricted attributes
- CVE-2026-40486 — GitLab Advisory Database
- CWE-915 — Improperly Controlled Modification of Dynamically-Determined Object Attributes
- OWASP Web Security Testing Guide v4.2
- Kimai Roles & Permissions Documentation
- `src/Form/UserEditType.php` — kimai/kimai main branch
- `src/Form/API/UserApiEditForm.php` — kimai/kimai main branch
- `src/API/UserController.php` — kimai/kimai main branch
- `Symfony/Security` 6.4 `AccessDecisionManager` — voter cache scope verification
- `Symfony/Security` 6.4 `Voter` base class — `vote()` / `voteOnAttribute()` call path

Classification:
This document contains details of an unpatched security vulnerability. Distribution is restricted to the assessment team and authorised stakeholders until an upstream patch is released and deployed. Do not share externally without written authorisation.