BACK

CVE-2026-109433: How One Disabled Permission Disabled Them All in Camunda Cockpit

CVE-2026-109433 7.1



Hemanshu Baviskar

Oct 14, 2026



Research

- TL;DR
  - At a Glance
  - The Same Predicate, Written Twice
  - What The Caller Does With That Answer
  - Born Inverted, And We Got The Blame Wrong First
  - A Single-variable A/B
  - What This Does Not Do
  - Severity
  - Fix And Detection
  - Credit & Scope

BOOK A DEMO

predicate with the comparison reversed. It returns the right answer for an empty list and the wrong answer for every other input. The empty list is the shipped default, which is why it survived seven years.

Severity	High, 7.1 (CVSS 4.0) / Medium, 6.5 (CVSS 3.1)
Confirmed on	`camunda/camunda-bpm-platform:run-7.24.0`, source commit `ee4826e5e7`
Status	CVE-2026-109433, reported through VulnCheck as CNA

TL;DR

- **Defect:** `AbstractCockpitPluginResource.isPermissionDisabled()` returns `true` when the configured list contains any permission *other* than the one being tested. Its caller then skips adding the permission check, so the plugin query runs with no authorization filter.
- **Impact:** With one unrelated entry in `disabledPermissions`, a user holding only `APPLICATION cockpit ACCESS` and no data grants received every process instance, definition and incident in the engine from the Cockpit plugin endpoints.
- **Precondition:** `disabledPermissions` non-empty. It defaults to `Collections.emptyList()`, so a stock deployment is unaffected. One entry, any entry, is enough.
- **Status:** CVE-2026-109433, reported through VulnCheck as CNA. Camunda 7 Community Edition is End of Life, so a local patch is the realistic remedy.

At a Glance

Product	Camunda 7 (`camunda-bpm-platform`), Cockpit webapp
Issue	High: Cockpit plugin endpoints skip authorization filtering when `disabledPermissions` is non-empty
CVE	CVE-2026-109433 (an earlier identifier, CVE-2026-77228, was assigned to the same defect)

	`onDisabled()` returns `true` when the list contains any permission other than the one tested, so the caller never adds the permission check
Impact	A user with only `APPLICATION cockpit ACCESS` receives metadata for every process instance, definition and incident: IDs, keys, names, counts and start times. No variables, no cross-tenant access
Requires	Not the default. `disabledPermissions` must contain at least one entry (any entry), and the attacker needs an authenticated Cockpit session
Introduced	Commit `1d53b8de3a` (March 18, 2019)
Tested on	`camunda/camunda-bpm-platform:run-7.24.0`, source commit `ee4826e5e7`
Severity	7.1, High CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N 6.5, Medium CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N
Fix status	Camunda 7 Community Edition is End of Life; a local patch is the realistic remedy
Disclosure	Reported through VulnCheck as CNA
Found by	**CodeAnt AI Security Research Team**

The Same Predicate, Written Twice

`disabledPermissions` is a documented process-engine option. Its javadoc describes it as the permissions that will not be taken into account in authorization checks, and the intended behaviour is narrow: drop the named permission, keep enforcing everything else. The engine's test suite pins exactly that, in `DisabledPermissionsAuthorizationTest.testDeploymentStatisticsIgnoreReadInstance()`, which disables `READ_INSTANCE`, asserts the instance count becomes visible, and asserts in the same test that the unrelated incident check still filters.

The engine implements it the way you would expect:

```
    List<String> disabledPermissions = ... .get
    if (disabledPermissions != null) {
      for (String disabledPermission : disabled
        if (permission.getName().equals(disable
          return true;
        }
      }
    }
    return false;
  }
```

Null-guarded, and true only when the permission under test is the one named. Cockpit has its own copy, in its own class, in a different module:

```
  // webapps/assembly/.../cockpit/plugin/resou
  protected boolean isPermissionDisabled(Permis
    List<String> disabledPermissions = ... .get
    for (String disabledPerm : disabledPermissi
      if (!disabledPerm.equals(permission.getNa
        return true;
      }
    }
    return false;
  }
```

One character of difference, and the null guard is gone too. Put the two side by side on real inputs and the copy is right in precisely one column:

`disabledPermissions`	Testing `READ`	Engine says	Cockpit says
`[]` (default)	is READ disabled?	false	false
`["READ"]`	is READ disabled?	true	false

`,"READ"]`	disabled?		

The third row is the vulnerability. An operator disables one permission for an entirely unrelated reason and Cockpit begins reporting that every permission is disabled.

What The Caller Does With That Answer

The predicate would be harmless if its result were advisory. It is not. The caller uses it to decide whether to attach the authorization filter to the query at all:

```
// AbstractCockpitPluginResource.java:156
protected void addPermissionCheck(QueryParame
                                  String query
  if (!isPermissionDisabled(permission)) {
    ...
    query.getAuthCheck().addAtomicPermissionC
  }
}
```

No filter is added, so the SQL runs unrestricted. The endpoint does not fail, it over-answers.

The base plugin's process-instance resource adds exactly two checks this way, and with one unrelated permission in the list both are dropped in the same request:

```
configureAuthorizationCheck(query);
addPermissionCheck(query, PROCESS_INSTANCE,
addPermissionCheck(query, PROCESS_DEFINITION,
```

Born Inverted, And We Got The Blame Wrong First

We originally attributed this to commit `7655ff35c7` (2019-03-26), *"chore(webapp): adjust disabledPermissions property type"*, and told the story as a mechanical type refactor that silently inverted a security check. That attribution went into our advisory and into writing with the CNA. It is wrong, and the correction is worth more than the original.

`7655ff35c7` changed only the type:

```
- for (Permission disabledPerm : disabledPerm
-   if (!disabledPerm.getName().equals(permis
+ for (String disabledPerm : disabledPermissi
+   if (!disabledPerm.equals(permission.getNa
```

The negation is on both sides of the diff. The refactor inherited the bug, it did not cause it.

The real origin is `1d53b8de3a` (2019-03-18), *"fix(webapp): move permission check to webapp"*, eight days earlier. That commit **added** `isPermissionDisabled` to the Cockpit class, with the comparison already reversed. There was never a correct version of this method in the webapp.

Which makes it a better story, not a worse one. Nobody's refactor broke a working control. Somebody moved a permission check out of the engine and into the webapp, hand-wrote the predicate from scratch in the new location, reversed the comparison on the first attempt, and no test existed to notice. Nothing under `webapps/assembly/src/test/java` references `isPermissionDisabled` or `disabledPermissions` to this day.

> Duplicating a security predicate into a second module is the defect. The reversed comparison is just the form it took.

Two instances, same image, same configuration, same user whose only grant is `APPLICATION cockpit ACCESS`, differing by one property. The control instance is checked for cleanliness first, so the difference cannot be misattributed:

```
GET /camunda/api/cockpit/plugin/base/default/

disabledPermissions UNSET  ->  lowpriv sees
disabledPermissions SET    ->  lowpriv sees

# the single variable
disabledPermissions = ["RETRY_JOB"]   <- unre
```

One entry, naming a permission neither dropped check tests, removes both dropped checks.

What This Does Not Do

Our own report overstated this in two places, and both corrections cut the impact down. They are here rather than buried, because a maintainer who finds them first is entitled to discount everything else we wrote.

- **No process variables, no incident messages, no stack traces.** The report claimed all three. The exposure is metadata: `ProcessInstanceDto` carries `id`, `businessKey`, `startTime`, `incidents` and `suspensionState`, and `ProcessDefinitionDto` carries `id`, `name`, `key`, `version`, `calledFromActivityIds` and `failedJobs`. We pivoted a leaked instance id to `engine-rest/process-instance/<id>/variables` and got `AuthorizationException`. Variables appear in this plugin layer only as query criteria, never in the response.
- **Not across every tenant.** The report claimed tenant-wide reach. Tenant isolation is a separate mechanism, `configureTenantCheck`, and it still works. We verified it does.
- **Invisible in the UI, which cuts both ways.** Cockpit still renders "No process definitions deployed" for an affected user. The leak is in the HTTP response body, reachable by curl, devtools or `fetch()` with an ordinary Cockpit session cookie. A tester who only clicks through the interface will wrongly conclude the instance is unaffected.

Severity

Version	Vector	Score
CVSS 3.1	`AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N`	**6.5 Medium**
CVSS 4.0	`AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N`	**7.1 High**

This one crosses the severity word between versions, which is a property of the two formulas rather than a disagreement about the facts. The same vector content is Medium under 3.1 and High under 4.0, and we are publishing both rather than picking the flattering one.

We tried to justify 7.0 or above under 3.1 and could not. Two routes exist and both fail on the evidence. `PR:N` would give 7.5, but the exploit needs an authenticated session holding `APPLICATION cockpit ACCESS`, so `PR:L` is simply what the precondition is. `S:C` would give 7.7, and we argued it adversarially on the grounds that leaking across an authorization boundary crosses a security authority; the argument does not hold, because the leak stays inside the engine's own authorization scope and the separate tenant mechanism remains intact.

`I:N` and `A:N` are deliberate. The affected code path is query filtering, and no write path was shown to be affected.

Fix And Detection

Delete the duplicate. `PermissionCheckBuilder:110-112` already reaches the engine's implementation, which is the non-duplicating route the codebase itself demonstrates. If the copy has to stay, it is one line:

```
protected boolean isPermissionDisabled(Permis
  List<String> disabled = ... .getDisabledPe
```

That restores the null guard the copy dropped, and the comparison the copy never had.

- **Check whether the property is set, and if it is, that is the whole answer.** Any non-empty `disabledPermissions` on a 7.x deployment means Cockpit plugin queries are unfiltered today.
- **Add the test as an A/B.** The engine has `DisabledPermissionsAuthorizationTest`; the webapp has nothing. The control arm matters as much as the test arm, since the bug is the difference between them.
- **Detection is silent by design.** Nothing is logged when the property is set, Cockpit renders normally, and an operator who enabled the option for performance has no signal that they also disabled authorization filtering.

Credit & Scope

Finding	CodeAnt AI Security Research Team. The divergence between the two implementations, the single-variable A/B and the browser confirmation that the UI hides it are ours.
Corrected here	Our advisory attributed the inversion to `7655ff35c7` and described it as a type refactor. The true origin is `1d53b8de3a`, which introduced the method already inverted. Our advisory also claimed variables, incident messages, stack traces and tenant-wide reach; section 5 withdraws all four.
Coordination	Reported through VulnCheck as CNA. CVE-2026-109433. An earlier identifier, CVE-2026-77228, was assigned to the same defect.
Testing	Locally-provisioned containers only. Read-only; nothing destroyed, nothing exfiltrated, no third-party system touched.

[FAQ]

Frequently Asked Questions

Does this affect every Camunda 7 installation? +

How do I know if I'm affected? +

Does an attacker need an account? +

What data is exposed? +

Does this cross tenant boundaries? +

Why can't I see the leak in the Cockpit UI? +

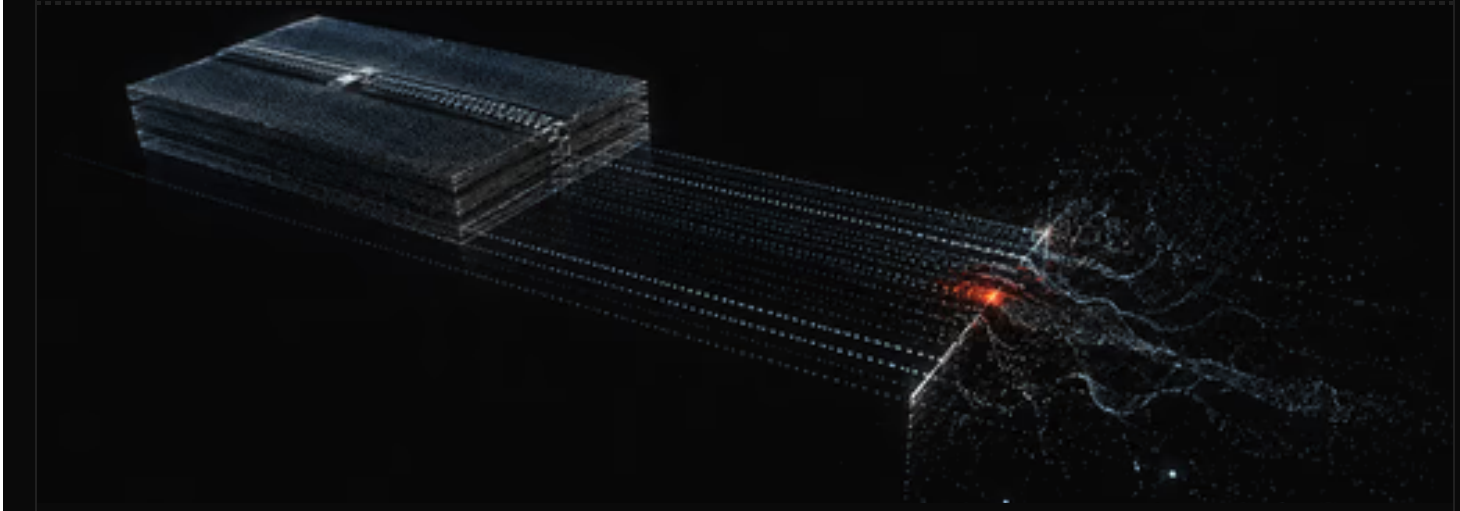
More from CodeAnt

VIEW ALL

Mar 3, 2026

CVE-2026-29000: pac4j-jwt Auth Bypass PoC With a Public Key

CVE-2026-29000 CVSS 10



Mar 9, 2026

CVE-2026-28292: simple-git RCE via Case-Sensitivity Bypass

CVE-2026-28292 CVSS 9.8

CodeAnt AI Product Resources Security Reseach Customer Enterprise

Mar 11, 2026

CVE-2026-31988: jszip DoS via Malformed ZIP File

CVE-2026-31988 CVSS 6.9

[GET STARTED]

Find out what's already exploitable in your codebase.

START PENTEST BOOK A DEMO

CodeAnt AI

PRODUCT
Blackbox Pentesting
Whitebox Pentesting
Graybox Pentesting

COMPANY
About
Pricing
Customer Story

DEVELOPERS
Documentati
Integrations
Security Res

355 Bryant St, San
Francisco, CA 94107

Enterprise

Contact Sales

Copyright © 2026 CodeAnt AI. All rights reserved.