GOVERNMENT OF THE REPUBLIC OF VANUATU

PRIME MINISTER'S OFFICE

CERTVU
DEPARTMENT OF COMMUNICATIONS & DIGITAL TRANSFORMATION

PM B 9108 Port Vila, Vanuatu
Tel: (678) 33380

GOUVERNEMENT DE LA REPUBLIQUE DU VANUATU

BUREAU DU PREMIER MINISTRE

CERTVU

DEPARTMENT DE COMMUNICATION ET DE TRANSFORMATION NUMERIQUE

SPP 9108 Port Vila, Vanuatu
Tel: (678) 33380

4 September 2026

Advisory 251: python-jose HS256 Token Forgery via Incomplete HMAC Key Validation (CVE-2026-85394).

Release Date: 3rd September 2026
Impact: CRITICAL
TLP: CLEAR

The Department of Communications and Digital Transformation (DCDT) through CERT Vanuatu (CERTVU), provides the following advisory.

This alert is relevant to Organizations, System/Network administrators, and software development teams that build or operate Python applications using python-jose for JWT-based authentication or authorization, including via frameworks such as FastAPI, where python-jose is commonly used for OAuth2/JWT handling. This alert is intended to be understood by technical users and systems administrators.

What is it?

CVE-2026-85394 is a critical signature-verification bypass in python-jose, a widely-used Python library (approximately 1,800 GitHub stars and roughly 40 million PyPI downloads per month) for JSON Object Signing and Encryption (JOSE), commonly used to implement JWT-based authentication, including in FastAPI's own official OAuth2/JWT tutorial. The library's guard against "algorithm confusion" attacks - where an attacker tricks a service into verifying an HS256 (HMAC) token using an asymmetric public key as the shared secret - was already patched once, as CVE-2024-33663, by rejecting keys in PEM or OpenSSH format. token, bypassing the intended signature check on any application that does not explicitly restrict the algorithms it accepts.

What are the systems affected?

The following version(s) are affected:

- python-jose through 3.5.0, all currently published versions – (Affected)

No fixed version has been published at the time of writing – see "Mitigation process" below

What does this mean?

Typical attack flow:

1. **Obtain the service's public verification key**
    - An attacker obtains the public key an affected application uses to verify JWTs signed with an asymmetric algorithm such as RS256 or ES256 — for example, by requesting it from a public JWKS endpoint, which is by design not secret.
2. **Forge an HS256 token using the public key as an HMAC secret**
    - The attacker DER-encodes the public key and uses it as the shared secret to sign a token with the HS256 algorithm. Because python-jose's guard against this only recognizes PEM- and SSH-formatted keys, the DER-encoded key is accepted, and the forged token passes verification on any application that does not explicitly restrict which algorithms it accepts.

Attack vectors:

- A network-based attack against any application using python-jose to verify JWTs with an asymmetric algorithm, where the application's public key is obtainable (for example via a JWKS endpoint) and the verification code does not explicitly restrict accepted algorithms to the asymmetric algorithm in use.
- No user interaction, no privileges, and no special access conditions are required beyond knowledge of the public key (CVSS AV:N/AC:L/PR:N/UI:N). CERTVU is not aware of confirmed active exploitation at the time of writing, but proof-of-concept forgery details are already public and no fixed version currently exists.

Successful exploitation may allow attackers to:

- Forge validly-signed authentication or authorization tokens without ever holding a legitimate account or credential, on any affected application that does not explicitly restrict accepted algorithms.
- Impersonate any user — including privileged or administrative accounts, depending on how the application derives identity from the token — and gain unauthorized access to protected resources and functionality.

Mitigation process

CERTVU recommends the following:

1. Enforce an explicit algorithm allow-list in every JWT verification call — no vendor patch is currently available

- Wherever your application calls python-jose's jwt.decode() (or an equivalent verification function), explicitly pass the algorithms parameter naming only the specific asymmetric algorithm you use (for example, algorithms=["RS256"]), and never allow "HS256" to be silently accepted for tokens intended to be verified asymmetrically. This is a workaround, not a fix — the underlying key-validation gap remains present in the library.

2. Treat this as unpatched: python-jose's maintainers had not released a fixed version or formal advisory at the time of writing (GitHub issue #414 remains open)
3. Audit the estate for a python-jose dependency, direct or transitive
4. Where practical, evaluate migrating to an actively-maintained JWT library
5. Review authentication logs for anomalous tokens

Report suspected compromise to CERTVU at incident@cert.gov.vu or on telephone (678) 33380.

Reference

1. https://www.cve.org/CVERecord?id=CVE-2026-85394
2. https://github.com/mpdavis/python-jose/issues/414