PyJWT accepts unknown `crit` header extensions

High **jpadilla** published **GHSA-752w-5fwx-jx9f** on Mar 12

Package

pyjwt (pip)

Affected versions

< 2.11.0

Patched versions

2.12.0

Description

Summary

PyJWT does not validate the `crit` (Critical) Header Parameter defined in RFC 7515 §4.1.11. When a JWS token contains a `crit` array listing extensions that PyJWT does not understand, the library accepts the token instead of rejecting it. This violates the **MUST** requirement in the RFC.

This is the same class of vulnerability as CVE-2025-59420 (Authlib), which received CVSS 7.5 (HIGH).

RFC Requirement

RFC 7515 §4.1.11:

> The "crit" (Critical) Header Parameter indicates that extensions to this specification and/or [JWA] are being used that **MUST** be understood and processed. [...] If any of the listed extension Header Parameters are **not understood and supported** by the recipient, then the **JWS is invalid**.

Proof of Concept

```python
import jwt  # PyJWT 2.8.0
import hmac, hashlib, base64, json

# Construct token with unknown critical extension
header = {"alg": "HS256", "crit": ["x-custom-policy"], "x-custom-policy": "require-mfa"}
payload = {"sub": "attacker", "role": "admin"}

def b64url(data):
    return base64.urlsafe_b64encode(data).rstrip(b"=").decode()

h = b64url(json.dumps(header, separators=(",", ":")).encode())
p = b64url(json.dumps(payload, separators=(",", ":")).encode())
sig = b64url(hmac.new(b"secret", f"{h}.{p}".encode(), hashlib.sha256).digest())
token = f"{h}.{p}.{sig}"

# Should REJECT — x-custom-policy is not understood by PyJWT
try:
    result = jwt.decode(token, "secret", algorithms=["HS256"])
    print(f"ACCEPTED: {result}")
    # Output: ACCEPTED: {'sub': 'attacker', 'role': 'admin'}
except Exception as e:
    print(f"REJECTED: {e}")
```

Expected: `jwt.exceptions.InvalidTokenError: Unsupported critical extension: x-custom-policy`
Actual: Token accepted, payload returned.

Comparison with RFC-compliant library

```python
# jwcrypto — correctly rejects
from jwcrypto import jwt as jw_jwt, jwk
key = jwk.JWK(kty="oct", k=b64url(b"secret"))
jw_jwt.JWT(jwt=token, key=key, algs=["HS256"])
# raises: InvalidJWSObject('Unknown critical header: "x-custom-policy"')
```

Impact

- **Split-brain verification** in mixed-library deployments (e.g., API gateway using jwcrypto rejects, backend using PyJWT accepts)
- **Security policy bypass** when `crit` carries enforcement semantics (MFA, token binding, scope restrictions)
- **Token binding bypass** — RFC 7800 `cnf` (Proof-of-Possession) can be silently ignored

- See CVE-2025-59420 for full impact analysis

Suggested Fix

In `jwt/api_jwt.py`, add validation in `_validate_headers()` or `decode()`:

```
_SUPPORTED_CRIT = {"b64"}  # Add extensions PyJWT actually supports

def _validate_crit(self, headers: dict) -> None:
    crit = headers.get("crit")
    if crit is None:
        return
    if not isinstance(crit, list) or len(crit) == 0:
        raise InvalidTokenError("crit must be a non-empty array")
    for ext in crit:
        if ext not in self._SUPPORTED_CRIT:
            raise InvalidTokenError(f"Unsupported critical extension: {ext}")
        if ext not in headers:
            raise InvalidTokenError(f"Critical extension {ext} not in header")
```

CWE

- CWE-345: Insufficient Verification of Data Authenticity
- CWE-863: Incorrect Authorization

References

- RFC 7515 §4.1.11
- CVE-2025-59420 — Authlib crit bypass (CVSS 7.5)
- RFC 7800 — Proof-of-Possession Key Semantics

Severity

High 7.5 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	None

User interaction	None
Scope	Unchanged
Confidentiality	None
Integrity	High
Availability	None

Learn more about base metrics

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

CVE ID

CVE-2026-32597

Weaknesses

- CWE-345
- CWE-863

Credits

dmbs335 Reporter