PrefectHQ / **prefect** Public

Code Issues 804 Pull requests 41 Discussions Actions Security an

Commit e216171

bunchesofdonald and **claude** authored on Mar 10 · ✕ 70 / 77 · Verified

```
Fix auth bypass via endswith() health check path exemption (#21063)

Co-authored-by: Claude Opus 4.6 <noreply@anthropic.com>
```

main (#21063) · prefect-redis-0.2.11 ••• 3.6.22.dev7

1 parent d7a24b6 commit e216171

2 files changed +68 -5

Top

Filter files…

- src/prefect/server/api
  - server.py
- tests/server
  - test_app.py

Search within code

src/prefect/server/api/server.py

```
@@ -417,16 +417,20 @@ async def server_version() -> str: # type:
ignore[reportUnusedFunction]
417 417         auth_string = prefect.settings.PREFECT_SERVER_API_AUTH_STRING.value()
418 418
419 419         if auth_string is not None:
    420 +             health_check_paths = {health_check_path, "/ready"}
420 421
421 422             @api_app.middleware("http")
```

```
422  423          async def token_validation(request: Request, call_next: Any):  # type:
     ignore[reportUnusedFunction]
423  424              header_token = request.headers.get("Authorization")
424  425
425       -          # used for probes in k8s and such
426       -          if (
427       -              request.url.path.endswith(("health", "ready"))
428       -              and request.method.upper() == "GET"
429       -          ):
     426  +          # Allow unauthenticated health/ready probes (e.g. k8s).
     427  +          # Use scope["path"] (not request.url.path) because url.path
     428  +          # can be spoofed via Host header manipulation. Use exact path
     429  +          # matching (not suffix matching) to prevent auth bypass via
     430  +          # crafted paths like /variables/name/system-health.
     431  +          scope = request.scope
     432  +          app_path = scope["path"].removeprefix(scope.get("root_path", ""))
     433  +          if app_path in health_check_paths and request.method.upper() ==
     "GET":
430  434                  return await call_next(request)
431  435              try:
432  436                  if header_token is None:
```

tests/server/test_app.py

```
     •••  @@ -1,3 +1,6 @@
     1    + import base64
     2    + from collections.abc import Generator
     3    +
1    4    import pytest
2    5    from fastapi.testclient import TestClient
3    6
     ⇣ ⇡  @@ -48,3 +51,59 @@ def test_app_add_csrf_middleware_when_enabled(enabled:
          bool):
48   51              if "CsrfMiddleware" in str(middleware)
49   52          ]
50   53          assert len(matching) == (1 if enabled else 0)
     54   +
     55   +
     56   + class TestAuthMiddleware:
     57   +     AUTH_STRING = "admin:test"
```

```
58  +     VALID_AUTH_HEADER = "Basic " + base64.b64encode(b"admin:test").decode()
59  +
60  +     @pytest.fixture()
61  +     def anonymous_client(self) -> Generator[TestClient, None, None]:
62  +         with temporary_settings({PREFECT_SERVER_API_AUTH_STRING:
      self.AUTH_STRING}):
63  +             app = create_app(ignore_cache=True)
64  +             yield TestClient(app)
65  +
66  +     def test_health_bypasses_auth(self, anonymous_client: TestClient):
67  +         response = anonymous_client.get("/api/health")
68  +         assert response.status_code == 200
69  +
70  +     def test_ready_bypasses_auth(self, anonymous_client: TestClient):
71  +         response = anonymous_client.get("/api/ready")
72  +         assert response.status_code == 200
73  +
74  +     def test_other_routes_require_auth(self, anonymous_client: TestClient):
75  +         response = anonymous_client.get("/api/version")
76  +         assert response.status_code == 401
77  +
78  +     def test_valid_auth_allows_access(self, anonymous_client: TestClient):
79  +         response = anonymous_client.get(
80  +             "/api/version",
81  +             headers={"Authorization": self.VALID_AUTH_HEADER},
82  +         )
83  +         assert response.status_code == 200
84  +
85  +     def test_path_ending_in_health_does_not_bypass_auth(
86  +         self, anonymous_client: TestClient
87  +     ):
88  +         """Regression: suffix matching allowed bypass via resource names like
89  +         /api/variables/name/system-health"""
90  +         response = anonymous_client.get("/api/variables/name/system-health")
91  +         assert response.status_code == 401
92  +
93  +     def test_path_ending_in_ready_does_not_bypass_auth(
94  +         self, anonymous_client: TestClient
95  +     ):
96  +         response = anonymous_client.get("/api/variables/name/system-ready")
```

```
97  +         assert response.status_code == 401
98  +
99  +     def test_host_header_manipulation_does_not_bypass_auth(
100 +         self, anonymous_client: TestClient
101 +     ):
102 +         """Regression: Host header like 'localhost/health?' causes
103 +         request.url.path to evaluate to '/health' while the real route
104 +         is still served."""
105 +         response = anonymous_client.get(
106 +             "/api/version",
107 +             headers={"Host": "localhost/health?"},
108 +         )
109 +         assert response.status_code == 401
```

Comments 0

Please sign in to comment.