DARKMOON

VULNERABILITY ASSESSMENT

REPORT

Automated penetration testing powered by AI

CAMPAIGN DETAILS

Target	** OpenNHP public demo (opennhp.org/demo) — protected vs exposed **Date:** 2026-07-26 **Assessor:** Darkmoon (autonomous AI penetration-testing platform) **Authorization:** Written permission from the OpenNHP project owner **Methodology:** NIST SP 800-115 · ISO 27001 · MITRE ATT&CK · OWASP Top 10
Date	
Methodology	ISO 27001 / NIST SP 800-115 / MITRE ATT&CK
Platform	Darkmoon AI Security Platform

CLASSIFICATION

CONFIDENTIAL

RESTRICTED USE

dark-moon.org

4 Rue de Dunkerque
31200 Toulouse

contact@asc-it.fr

+33 5 62 26 05 13

Vulnerability Assessment Report — OpenNHP Public Demo

Target: OpenNHP public demo (opennhp.org/demo) — protected vs exposed Date: 2026-07-26 Assessor: Darkmoon (autonomous AI penetration-testing platform) Authorization: Written permission from the OpenNHP project owner Methodology: NIST SP 800-115 · ISO 27001 · MITRE ATT&CK · OWASP Top 10

MANAGEMENT SUMMARY

Overall Security Posture: HIGH

An autonomous AI attacker was pointed at the OpenNHP public demo. The three NHP-protected hosts (server2, ac2, ac) were completely undiscoverable — zero open ports across four independent probing techniques plus ICMP — so no attack could even begin. The exposed demo hosts, by contrast, were fully discoverable and yielded 51 findings: 4 high, 10 medium, 17 low, 20 informational.

Most Critical Issues

- Cross-origin credential/token theft (HIGH): the exposed auth backend reflects any Origin back with Access-Control-Allow-Credentials: true, and returns the access token (acToken) in a cross-origin-readable JSON body. Any malicious website a logged-in user visits can perform authenticated requests and read the token.

- Credential boundary (informational but pivotal): config.json publicly serves the agent private keys for both clusters, including the NHP-protected Cluster 2 — by design for the demo, and the precise illustration that OpenNHP's protection reduces to the secrecy of the agent credential.

Required Actions

- Restrict CORS to an explicit allowlist and never combine reflected origins with credentials.
- Stop returning access tokens in cross-origin-readable responses; scope cookies with SameSite.
- Keep production agent private keys secret (the demo publishes them intentionally).

1. EXECUTIVE SUMMARY

You cannot attack what you cannot see. The NHP-protected hosts produced zero findings because they never presented a reachable port. Every finding below exists solely because a host is reachable; move it behind NHP and its finding list goes to zero. The exposed-side issues are ordinary web-hygiene problems (CORS, headers, token handling) headlined by a genuine cross-origin credential-theft condition (verified by hand).

Findings Summary

Severity	Count
High	4
Medium	10
Low	17
Info	20
Total	51

2. FINDINGS TABLE

#	Title	Severity	CVSS	Host
1	CORS Origin Reflection with Credentials on Auth API Endpoints	HIGH	7.5	server.opennhp.org
2	CORS Origin Reflection with Credentials on Auth API	HIGH	7.5	server/auth-plugin.opennhp.org
3	Authentication Token (acToken) Exposed in Cross-Origin Readable JSON Response	HIGH	7.5	auth-plugin.opennhp.org
4	acToken Exposed Cross-Origin via CORS Origin Reflection in JSON Response	HIGH	7.5	server.opennhp.org
5	No Rate Limiting on Authentication Endpoint	MEDIUM	5.3	server/auth-

#	Title	Severity	CVSS	Host
				plugin.opennhp.org
6	Missing Security Headers on All Three Hosts (No CSP, No X-Frame-Options, No HSTS, No X-Content-Type-Options)	MEDIUM	5.3	auth-plugin.opennhp.org
7	Credentials Transmitted in GET URL Query Parameters (Password in URL)	MEDIUM	5.3	auth-plugin.opennhp.org
8	No Rate Limiting on Authentication Endpoint	MEDIUM	5.3	server.opennhp.org
9	TLS Certificate SAN Mismatch on server.opennhp.org	MEDIUM	4.8	server/auth-plugin.opennhp.org
10	Missing Security Headers on All Exposed Hosts	MEDIUM	4.3	server.opennhp.org
11	Credentials Transmitted via GET Query String in Auth Plugin API	MEDIUM	4.3	server.opennhp.org
12	Authentication Credentials Transmitted in URL Query Parameters (GET)	MEDIUM	4.3	server/auth-plugin.opennhp.org
13	Clickjacking: All Three Hosts Frameable (No X-Frame-Options / CSP frame-ancestors)	MEDIUM	4.3	auth-plugin.opennhp.org
14	Login Pages Vulnerable to Clickjacking (No Frame Protection)	MEDIUM	4.3	server.opennhp.org
15	Unauthenticated Server Infrastructure Enumeration via /servers Endpoint	LOW	3.7	agent/reg/relay (3.149.105.15)
16	CORS Misconfiguration: Access-Control-Allow-Origin: * with Access-Control-Allow-Credentials: true	LOW	3.7	auth-plugin.opennhp.org

#	Title	Severity	CVSS	Host
17	nhp-token Cookie Scoped to Parent Domain (.opennhp.org) - Shared Across All Subdomains	LOW	3.7	auth-plugin.opennhp.org
18	CORS expose-headers Includes Set-Cookie (Attempted Cookie Exposure to Cross-Origin Callers)	LOW	3.7	auth-plugin.opennhp.org
19	Missing Host Header Validation - Default Virtual Host Serves Auth Endpoint	LOW	3.7	server/auth-plugin.opennhp.org
20	Unauthenticated Server Infrastructure Enumeration via /servers	LOW	3.7	server.opennhp.org
21	SSL Certificate Mismatch on server.opennhp.org	LOW	3.1	server.opennhp.org
22	Missing Subresource Integrity (SRI) on External Scripts	LOW	3.1	server.opennhp.org
23	Missing Security Headers on Auth API Responses	LOW	3.1	server/auth-plugin.opennhp.org
24	No CSRF Protection on Authentication Login Form	LOW	3.1	auth-plugin.opennhp.org
25	Hardcoded Default Credentials Pre-filled in Login Form (user/password)	LOW	3.1	auth-plugin.opennhp.org
26	External JavaScript Dependencies Loaded from CDN Without Subresource Integrity (SRI)	LOW	3.1	agent.opennhp.org
27	Login Pages Vulnerable to Clickjacking (No X-Frame-Options/CSP)	LOW	3.1	server/auth-plugin.opennhp.org
28	No CSRF Protection on Login Form	LOW	3.1	server.opennhp.org

#	Title	Severity	CVSS	Host
29	NHP-Token Cookie Missing SameSite Attribute	LOW	2.4	server/auth-plugin.opennhp.org
30	NHP-Token Cookie Missing SameSite Attribute and Scoped to Parent Domain	LOW	2.4	server.opennhp.org
31	NHP-Token Cookie Double Percent-Encoding	LOW	2.0	server/auth-plugin.opennhp.org
32	NHP-Protected Hosts Confirmed Undiscoverable (Zero Attack Surface)	INFO	0.0	server2.opennhp.org (NHP-protected)
33	Nginx Version Disclosure on All Exposed Hosts	INFO	0.0	server.opennhp.org
34	NHP Version and Build Hash Disclosed in Custom Header	INFO	0.0	server.opennhp.org
35	Cryptographic Keys Exposed in Public config.json (By Design)	INFO	0.0	agent.opennhp.org
36	OpenSSH 8.7 Version Disclosure on Exposed Host	INFO	0.0	agent.opennhp.org
37	NHP Version and Git Commit Hash Disclosed in HTTP Headers	INFO	0.0	server/auth-plugin.opennhp.org
38	Server Version and Build Hash Disclosure via Custom CORS Header (access-control-nhp-ver)	INFO	0.0	auth-plugin.opennhp.org
39	Demo Cryptographic Private Keys Exposed in config.json (Intentionally Public)	INFO	0.0	agent.opennhp.org
40	innerHTML Usage with IP Address from External API (ipify.org) - Limited DOM XSS Sink	INFO	0.0	auth-plugin.opennhp.org

#	Title	Severity	CVSS	Host
41	Unvalidated Redirect from Server-Controlled redirectUrl (No Client-Side Validation)	INFO	0.0	auth-plugin.opennhp.org
42	Cryptographic Private Keys Populated in Editable DOM Input Fields	INFO	0.0	agent.opennhp.org
43	nhp-token Cookie Has Very Short Max-Age of 15 Seconds (May Cause Usability Issues)	INFO	0.0	auth-plugin.opennhp.org
44	Default/Weak Demo Credentials (user/password)	INFO	0.0	server/auth-plugin.opennhp.org
45	Relay API Verbose Error Messages Enable Endpoint Enumeration	INFO	0.0	agent/reg/relay (3.149.105.15)
46	Inconsistent HTTP Method Handling - PUT Returns 204, DELETE Returns 204	INFO	0.0	server/auth-plugin.opennhp.org
47	Nginx Version Disclosure in Server Header	INFO	0.0	server/auth-plugin.opennhp.org
48	Authentication Failures Return HTTP 200 Instead of 401	INFO	0.0	server/auth-plugin.opennhp.org
49	Agent Private Keys Exposed in Public Config.json (Intentional Demo)	INFO	0.0	agent/reg/relay (3.149.105.15)
50	Relay API Verbose Error Messages	INFO	0.0	relay.opennhp.org
51	Authentication Failures Return HTTP 200 Instead of 401	INFO	0.0	server.opennhp.org

3. VULNERABILITY DETAILS

1. [HIGH] CORS Origin Reflection with Credentials on Auth API Endpoints

Severity	HIGH
CVSS	7.5 CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N
Host	server.opennhp.org
Endpoint	https://server.opennhp.org, https://auth-plugin.opennhp.org, https://demologin.opennhp.org
Category	cors_misconfiguration
Status	CONFIRMED
MITRE ATT&CK	T1189 Drive-by Compromise
Detected by	pentest-orchestrator

Description

Multiple exposed hosts (server.opennhp.org, auth-plugin.opennhp.org, demologin.opennhp.org) return Access-Control-Allow-Origin: combined with Access-Control-Allow-Credentials: true. Per the Fetch specification, browsers will not send credentials when ACAO is , so the combination is technically inert. However, it indicates a misconfiguration pattern: if the wildcard is ever changed to reflect the Origin header, credential-bearing cross-origin requests would become exploitable. The relay.opennhp.org also uses ACAO:* but without credentials. Nuclei also flagged arbitrary-origin reflection on server.opennhp.org.

Technical Analysis

The CORS policy sets Access-Control-Allow-Origin: * alongside Access-Control-Allow-Credentials: true. While browsers block this specific combination from sending cookies, it represents a defense-in-depth gap. The response also exposes Set-Cookie in Access-Control-Expose-Headers, and the acTokens are returned in the JSON body (not just cookies), meaning a malicious cross-origin page could read the token from the response body.

Raw Request

```
curl -sk -H "Origin: https://evil.com" -I https://server.opennhp.org/api/login
```

Raw Response

```
HTTP/2 404\naccess-control-allow-credentials: true\naccess-control-allow-headers: Content-Type, Content-Length, Authorization, X-NHP-Ver, Cookie\naccess-control-allow-methods: GET, OPTIONS, POST\naccess-control-allow-origin: *\naccess-control-expose-headers: Content-Type, Content-Length, Set-Cookie\naccess-control-max-age: 300
```

Remediation

Replace Access-Control-Allow-Origin: * with an explicit allowlist of trusted origins. Remove Access-Control-Allow-Credentials: true if wildcard origin is required. Restrict Access-Control-Expose-Headers to only necessary headers.

2. [HIGH] CORS Origin Reflection with Credentials on Auth API

Severity	HIGH
CVSS	7.5 CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N
Host	server/auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example
Category	cors_misconfiguration
Status	CONFIRMED
MITRE ATT&CK	T1189 Drive-by Compromise
Detected by	go-web-agent

Description

The auth-plugin.opennhp.org API reflects any Origin header value in the Access-Control-Allow-Origin response header while simultaneously setting Access-Control-Allow-Credentials: true. This combination allows any website to make credentialed cross-origin requests to the API and read the responses, including authentication tokens (acTokens), nhp-token cookies, and session data. Tested with Origin: https://evil.com, Origin: null, and

Origin: data://evil - all were reflected back with credentials allowed. The nhp-token cookie is set with Domain=opennhp.org so any subdomain attack vector is also viable. An attacker hosting a malicious page could steal a victim's auth tokens by having them visit the page while authenticated.

Technical Analysis

The server blindly reflects any Origin header sent by the client into the Access-Control-Allow-Origin response header. Combined with Access-Control-Allow-Credentials: true, this allows cross-origin JavaScript from any domain to read authenticated responses. The 'null' origin is particularly dangerous as it's used by sandboxed iframes, data: URIs, and local files. The responses contain sensitive auth tokens (acTokens), redirect URLs, and the server sets an HttpOnly cookie (nhp-token) scoped to the opennhp.org domain. A CSRF-style attack could exploit this to exfiltrate tokens.

Exploitation Commands

```
curl -skD- -H "Origin: https://evil.com" "https://auth-plugin.opennhp.org/plugins/example?
resid=demo&action=valid&username=user&password=password"
curl -skD- -H "Origin: null" "https://auth-plugin.opennhp.org/plugins/example?resid=demo&a
ction=valid&username=user&password=password"
curl -skD- -H "Origin: data://evil" "https://auth-plugin.opennhp.org/plugins/example?resid
=demo&action=valid&username=user&password=password"
```

Raw Request

```
GET /plugins/example?resid=demo&action=valid&username=user&password=password HTTP/2
Host: auth-plugin.opennhp.org
Origin: https://evil.com
```

Raw Response

```
HTTP/2 200
access-control-allow-credentials: true
access-control-allow-origin: https://evil.com
access-control-expose-headers: Content-Type, Content-Length, Set-Cookie
set-cookie: nhp-token=EonoKgB%252BIsBMMe6xtdYFwGOV%252BRL%252BsZ%252BaJCZDt%252FTa57U%253D
; Path=/; Domain=opennhp.org; Max-Age=15; HttpOnly; Secure

{"errCode":"0","resHost":{"demoServer":"ac.opennhp.org"},"opnTime":15,"agentAddr":"IP_PUBL
IC_003","acTokens":{"demoServer":"EonoKgB+IsBMMe6xtdYFwGOV+RL+sZ+aJCZDt/Ta57U="},"preActio
ns":{"demoServer":null},"redirectUrl":"https://ac.opennhp.org"}
```

Evidence / Logs

Origin: https://evil.com -> Access-Control-Allow-Origin: https://evil.com + Access-Control-Allow-Credentials: true
Origin: null -> Access-Control-Allow-Origin: null + Access-Control-Allow-Credentials: true

Origin: data://evil -> Access-Control-Allow-Origin: data://evil + Access-Control-Allow-Credentials: true
All responses include Set-Cookie with nhp-token and full JSON body with acTokens

Remediation

Implement an allowlist of trusted origins instead of reflecting any Origin. Never combine Access-Control-Allow-Origin: * or reflected origins with Access-Control-Allow-Credentials: true. Use a strict allowlist like ['https://demologin.opennhp.org', 'https://agent.opennhp.org', 'https://server.opennhp.org'].

3. [HIGH] Authentication Token (acToken) Exposed in Cross-Origin Readable JSON Response

Severity	HIGH
CVSS	7.5 CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N
Host	auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid&username=user&password=password
Category	information_disclosure
Status	CONFIRMED
MITRE ATT&CK	T1552 Unsecured Credentials
Detected by	browser-playwright

Description

The authentication validation endpoint returns acTokens (access control tokens) in its JSON response body, and these tokens are readable from any origin due to the Access-Control-Allow-Origin: * CORS header. A malicious website can call the validation endpoint with known demo credentials (user/password) via a cross-origin fetch() and read the acTokens, redirectUrl, and agentAddr from the response. This was confirmed by making a cross-origin fetch from agent.opennhp.org to auth-plugin.opennhp.org and successfully reading the

acTokens value. While the credentials are known demo credentials, the pattern is insecure because any website can authenticate on behalf of any user who visits their page (if credentials are known) and obtain their access tokens.

Technical Analysis

The auth validation endpoint at auth-plugin.opennhp.org returns a JSON object containing acTokens upon successful authentication. Because the server sets Access-Control-Allow-Origin: *, any website can make a cross-origin fetch to this endpoint. If an attacker knows (or guesses) valid credentials, they can authenticate from their malicious site and obtain the acTokens. In this demo scenario, the credentials are publicly known (user/password), so any website can obtain valid acTokens for the demo server. The acToken appears to be a time-limited access credential for the protected resource at ac.opennhp.org.

Exploitation Commands

```
Playwright (from agent.opennhp.org context):
fetch("https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid&username=us
er&password=password").then(r => r.json())
```

Raw Request

```
GET /plugins/example?resid=demo&action=valid&username=user&password=password HTTP/2
Host: auth-plugin.opennhp.org
Origin: https://agent.opennhp.org
```

Raw Response

```
HTTP/2 200
access-control-allow-origin: *
content-type: application/json; charset=utf-8

{"errCode":"0","resHost":{"demoServer":"ac.opennhp.org"},"opnTime":15,"agentAddr":"IP_PUBL
IC_003","acTokens":{"demoServer":"PhAFisGU1K2cqITzOae4a9l5w0v01nKI+6d0tPOf7to="},"redirect
Url":"https://ac.opennhp.org"}
```

Evidence / Logs

```
Cross-origin fetch result (from agent.opennhp.org):
{
"status": 200,
"hasAcTokens": true,
"acTokens": {"demoServer": "PhAFisGU1K2cqITzOae4a9l5w0v01nKI+6d0tPOf7to="},
"redirectUrl": "https://ac.opennhp.org",
```

```
"agentAddr": "IP_PUBLIC_003",
"acao": null (but ACAO:* in raw headers allows the read)
}
```

Remediation

Restrict CORS to specific allowed origins instead of wildcard. Do not expose acTokens in the JSON response body when ACAO: * is set. Consider using a session-based approach where the token is set only via httpOnly cookie (not in the response body).

4. [HIGH] acToken Exposed Cross-Origin via CORS Origin Reflection in JSON Response

Severity	HIGH
CVSS	7.5 CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N
Host	server.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid
Category	cors_misconfiguration
Status	CONFIRMED
MITRE ATT&CK	T1528 Steal Application Access Token
Detected by	headless-browser

Description

The auth validation response returns sensitive acTokens in the JSON response body. Combined with Access-Control-Allow-Origin: , any cross-origin JavaScript can read this token. While browsers block credential-bearing requests with ACAO:, if a user visits a malicious page that triggers the auth flow with known demo credentials (user/password), the acToken can be stolen. The token has a 15-second TTL (opnTime:15), but during that window, it grants access to the protected ac.opennhp.org server.

Technical Analysis

The authentication response body contains the access token (acTokens) which is readable cross-origin due to

ACAO:*. A malicious page could use fetch() to call this endpoint with the demo credentials and read the token from the response. The 15-second TTL limits the exploitation window but doesn't prevent it.

Raw Response

{\"errCode\":\"0\",\"resHost\":{\"demoServer\":\"ac.opennhp.org\"},\"opnTime\":15,\"acTokens\":{\"demoServer\":\"TK5soHdjr5lZ7jSM/FRkI8NLdRjrWC/U5KlxPPG9c1s=\"},\"redirectUrl\":\"https://ac.opennhp.org\"}

Remediation

Restrict CORS origin to a specific allowlist. Do not return sensitive tokens in the response body when ACAO is wildcard. Consider returning tokens only via HttpOnly cookies.

5. [MEDIUM] No Rate Limiting on Authentication Endpoint

Severity	MEDIUM
CVSS	5.3 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
Host	server/auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example
Category	authentication_bypass
Status	CONFIRMED
MITRE ATT&CK	T1110.001 Brute Force: Password Guessing
Detected by	go-web-agent

Description

The authentication endpoint at /plugins/example accepts unlimited failed login attempts without any rate limiting, account lockout, CAPTCHA, or progressive delay. 11 sequential failed authentication requests with different passwords all returned HTTP 200 with the same error message, with no throttling or blocking. This enables brute-force and credential stuffing attacks against user accounts. Combined with the simple default credentials (user/password), this significantly increases the risk of unauthorized access.

Technical Analysis

11 sequential failed authentication attempts were made with unique wrong passwords (wrong1 through wrong11). All 11 returned HTTP 200 with identical error responses, demonstrating zero rate limiting. No response code changes, no progressive delays, no CAPTCHA challenges, and no account lockout. This confirms the endpoint is vulnerable to brute-force credential guessing. The finding was demonstrated with exactly 11 requests as proof; the attack vector was not further exploited.

Exploitation Commands

```
bash -c 'for i in $(seq 1 11); do curl -sk -o /dev/null -w "%{http_code} " "https://auth-p
lugin.opennhp.org/plugins/example?resid=demo&action=valid&username=user&password=wrong$i";
sleep 0.2; done'
```

Raw Request

```
GET /plugins/example?resid=demo&action=valid&username=user&password=wrong1 HTTP/2
Host: auth-plugin.opennhp.org
```

Raw Response

```
HTTP/2 200
content-type: text/plain; charset=utf-8

{"errMsg": "auth error: user or password is incorrect"}
```

Evidence / Logs

```
Request 1: HTTP 200 - auth error
Request 2: HTTP 200 - auth error
Request 3: HTTP 200 - auth error
Request 4: HTTP 200 - auth error
Request 5: HTTP 200 - auth error
Request 6: HTTP 200 - auth error
Request 7: HTTP 200 - auth error
Request 8: HTTP 200 - auth error
Request 9: HTTP 200 - auth error
Request 10: HTTP 200 - auth error
Request 11: HTTP 200 - auth error
No rate limiting, no account lockout, no CAPTCHA triggered
```

Remediation

Implement rate limiting (e.g., 5 attempts per minute per IP/username). Add progressive delays, account lockout

after N failures, and CAPTCHA after failed attempts. Consider using a dedicated rate-limiting middleware like golang.org/x/time/rate.

6. [MEDIUM] Missing Security Headers on All Three Hosts (No CSP, No X-Frame-Options, No HSTS, No X-Content-Type-Options)

Severity	MEDIUM
CVSS	5.3 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N
Host	auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=login
Category	misconfiguration
Status	CONFIRMED
MITRE ATT&CK	T1189 Drive-by Compromise
Detected by	browser-playwright

Description

All three OpenNHP demo hosts (auth-plugin.opennhp.org, agent.opennhp.org, reg.opennhp.org) are missing critical HTTP security headers. No Content-Security-Policy, X-Frame-Options, X-Content-Type-Options, Strict-Transport-Security, X-XSS-Protection, Referrer-Policy, or Permissions-Policy headers are set on any response. This was confirmed by inspecting the HTTP response headers from each host via Playwright network interception.

Technical Analysis

All three hosts serving the OpenNHP demo application do not include any standard HTTP security headers. The absence of Content-Security-Policy means the browser has no instructions on allowed script sources, making the application more susceptible to XSS if any injection point is found. The lack of Strict-Transport-Security means the application does not enforce HTTPS at the browser level via HSTS. The missing X-Content-Type-Options allows MIME-type sniffing attacks.

Exploitation Commands

```
curl -sI https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=login
curl -sI https://agent.opennhp.org
curl -sI https://reg.opennhp.org
```

Evidence / Logs

```
auth-plugin.opennhp.org response headers:
X-Frame-Options: MISSING
Content-Security-Policy: MISSING
X-Content-Type-Options: MISSING
Strict-Transport-Security: MISSING
Referrer-Policy: MISSING
Permissions-Policy: MISSING
agent.opennhp.org: Same - all security headers MISSING
reg.opennhp.org: Same - all security headers MISSING
```

Remediation

Add the following headers to nginx configuration for all three hosts: Content-Security-Policy with a restrictive script-src policy, X-Frame-Options: DENY (or SAMEORIGIN), X-Content-Type-Options: nosniff, Strict-Transport-Security: max-age=31536000; includeSubDomains, Referrer-Policy: strict-origin-when-cross-origin, Permissions-Policy with restrictive defaults.

7. [MEDIUM] Credentials Transmitted in GET URL Query Parameters (Password in URL)

Severity	MEDIUM
CVSS	5.3 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
Host	auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid&username=user&password=password
Category	information_disclosure
Status	CONFIRMED
MITRE	T1552.001 Unsecured Credentials: Credentials In Files

Severity	MEDIUM
ATT&CK	
Detected by	browser-playwright

Description

The auth-plugin login form's nhpValidate() JavaScript function transmits username and password as GET query parameters in a fetch() call to the validation endpoint. The constructed URL is: https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid&username=USER&password=PASSWORD. While the fetch() itself is an XHR and won't appear in the browser address bar, the GET request with credentials will appear in: (1) nginx/1.30.2 access logs on the server, (2) any reverse proxy or WAF logs, (3) browser network history, (4) Referer headers if the validation endpoint triggers any redirects or sub-requests. The credentials are URL-encoded but transmitted in cleartext in the URL path.

Technical Analysis

The nhpValidate() function in the auth-plugin page constructs a GET URL containing the username and password as query parameters, then sends it via fetch(). This was confirmed by intercepting the actual network request in Playwright after clicking the Sign In button. The request URL clearly contains &username=user&password=password. GET parameters are logged by default in web server access logs (nginx), potentially cached by CDNs, and may leak via Referer headers. This violates the principle of never transmitting credentials in URLs.

Exploitation Commands

```
Playwright: page.click('#loginButton') - intercepted network request
```

Raw Request

```
GET /plugins/example?resid=demo&action=valid&username=user&password=password HTTP/2
Host: auth-plugin.opennhp.org
```

Raw Response

```
HTTP/2 200
content-type: application/json; charset=utf-8
access-control-allow-origin: *

{"errCode":"0","resHost":{"demoServer":"ac.opennhp.org"},"opnTime":15,"acTokens":{"demoSer
```

ver":"PI7VGbccn4rBtTbWFPudFJ89ctQjfWBI6k15Dqf3BWU="},"redirectUrl":"https://ac.opennhp.org
"}

Evidence / Logs

Intercepted network request after form submission:
GET https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid&username=user&
password=password
Response: {"errCode":"0","resHost":{"demoServer":"ac.opennhp.org"},"opnTime":15,"agentAddr
":"IP_PUBLIC_003","acTokens":{"demoServer":"..."},"redirectUrl":"https://ac.opennhp.org"}
JavaScript source (nhpValidate function):

```
const nhpValidUrl = "https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=val
id"
+ "&username=" + encodeURIComponent(user)
+ "&password=" + encodeURIComponent(password);
```

Remediation

Change the authentication validation endpoint to accept credentials via POST request body (JSON or form-encoded) instead of GET query parameters. This prevents credential leakage to server access logs and proxies.

8. [MEDIUM] No Rate Limiting on Authentication Endpoint

Severity	MEDIUM
CVSS	5.3 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
Host	server.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid
Category	authentication_flaw
Status	CONFIRMED
MITRE ATT&CK	T1110 Brute Force
Detected by	golang

Description

The authentication endpoint at auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid accepts unlimited login attempts without any throttling, account lockout, or CAPTCHA mechanism. Testing confirmed 11+ consecutive failed login attempts all returned HTTP 200 with no delays or blocking. This enables credential brute-force attacks against the authentication system.

Technical Analysis

No rate limiting, account lockout, or CAPTCHA was observed after 11+ rapid failed login attempts. All responses were immediate with no delay escalation.

Exploitation Commands

```
curl -sk "https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid&username
=admin&password=wrong1"\ncurl -sk "https://auth-plugin.opennhp.org/plugins/example?resid=d
emo&action=valid&username=admin&password=wrong2"\n(11 consecutive attempts, all returned H
TTP 200)
```

Remediation

Implement rate limiting (e.g., 5 attempts per minute per IP), progressive delays, and account lockout after repeated failures. Consider adding CAPTCHA after 3 failed attempts.

9. [MEDIUM] TLS Certificate SAN Mismatch on server.opennhp.org

Severity	MEDIUM
CVSS	4.8 CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N
Host	server/auth-plugin.opennhp.org
Endpoint	https://server.opennhp.org/
Category	security_misconfiguration
Status	CONFIRMED
MITRE ATT&CK	T1557.002 Adversary-in-the-Middle: ARP Cache Poisoning
Detected by	go-web-agent

Description

The TLS certificate served by server.opennhp.org (3.149.215.44) has a Subject Alternative Name (SAN) mismatch. The certificate is issued to CN=auth-plugin.opennhp.org with SANs covering only auth-plugin.opennhp.org and demologin.opennhp.org. The hostname server.opennhp.org is NOT included in the SAN list. curl confirms: 'subjectAltName does not match server.opennhp.org'. Strict TLS clients will reject this connection. Browsers will show a certificate warning. This means users of the auth demo at server.opennhp.org are trained to click through TLS warnings, or the connection is silently accepted by clients using -k/--insecure, which degrades the security posture and could enable MitM attacks.

Technical Analysis

The TLS certificate on 3.149.215.44 was issued by Let's Encrypt (CN=YE1) for auth-plugin.opennhp.org with SANs auth-plugin.opennhp.org and demologin.opennhp.org. The hostname server.opennhp.org resolves to the same IP but is not covered by the certificate. This is confirmed by curl's strict hostname verification which reports 'subjectAltName does not match'. Any HTTPS client with proper hostname verification will reject connections to server.opennhp.org. For a security product (NHP), serving auth pages with invalid TLS certificates undermines trust.

Exploitation Commands

```
echo | openssl s_client -connect server.opennhp.org:443 -servername server.opennhp.org 2>&
1 | grep -E "(subject|issuer|Verify)"
curl -svD- "https://server.opennhp.org/" -o /dev/null 2>&1 | grep -E "(subject|subjectAltN
ame|SSL)"
```

Raw Request

```
openssl s_client -connect server.opennhp.org:443 -servername server.opennhp.org
```

Raw Response

```
subject=CN = auth-plugin.opennhp.org
issuer=C = US, O = Let's Encrypt, CN = YE1
DNS:auth-plugin.opennhp.org, DNS:demologin.opennhp.org
(server.opennhp.org NOT in SAN list)
```

Evidence / Logs

```
openssl: subject=CN = auth-plugin.opennhp.org, Verify return code: 0 (ok) [openssl doesn't
check hostname by default]
curl: subjectAltName does not match server.opennhp.org
```

```
curl: SSL: no alternative certificate subject name matches target host name 'server.opennh
p.org'
Certificate SANs: DNS:auth-plugin.opennhp.org, DNS:demologin.opennhp.org
Missing: server.opennhp.org
```

Remediation

Add server.opennhp.org to the Let's Encrypt certificate's SAN list. Use certbot with -d server.opennhp.org -d auth-plugin.opennhp.org -d demologin.opennhp.org. Or use a wildcard certificate *.opennhp.org.

10. [MEDIUM] Missing Security Headers on All Exposed Hosts

Severity	MEDIUM
CVSS	4.3 CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:L/A:N
Host	server.opennhp.org
Endpoint	https://server.opennhp.org, https://agent.opennhp.org, https://reg.opennhp.org, https://relay.opennhp.org, https://demologin.opennhp.org, https://auth-plugin.opennhp.org
Category	security_headers
Status	CONFIRMED
MITRE ATT&CK	None
Detected by	pentest-orchestrator

Description

All exposed hosts are missing critical security headers: X-Frame-Options, Content-Security-Policy, Strict-Transport-Security (HSTS), X-Content-Type-Options, Referrer-Policy, Permissions-Policy, Cross-Origin-Embedder-Policy, Cross-Origin-Opener-Policy, Cross-Origin-Resource-Policy, X-Permitted-Cross-Domain-Policies. The relay.opennhp.org is the only host that sets X-Content-Type-Options: nosniff. The absence of these headers increases the risk of clickjacking, MIME-type confusion, and cross-origin attacks.

Technical Analysis

Nuclei detected missing headers on all scanned hosts: x-frame-options, content-security-policy, strict-transport-

security, x-content-type-options, referrer-policy, permissions-policy, cross-origin-embedder-policy, cross-origin-opener-policy, cross-origin-resource-policy, x-permitted-cross-domain-policies.

Exploitation Commands

```
nuclei -u https://server.opennhp.org -severity info\ncurl -sk -I https://agent.opennhp.org
```

Remediation

Add security headers to nginx configuration: add_header X-Frame-Options DENY; add_header Content-Security-Policy \"default-src 'self'\"; add_header Strict-Transport-Security \"max-age=31536000; includeSubDomains\"; add_header X-Content-Type-Options nosniff; add_header Referrer-Policy strict-origin-when-cross-origin; add_header Permissions-Policy \"camera=(), microphone=()\"; add_header Cross-Origin-Opener-Policy same-origin;

11. [MEDIUM] Credentials Transmitted via GET Query String in Auth Plugin API

Severity	MEDIUM
CVSS	4.3 CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N
Host	server.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid&username=user&password=password
Category	authentication_flaw
Status	CONFIRMED
MITRE ATT&CK	T1556 Modify Authentication Process
Detected by	pentest-orchestrator

Description

The authentication endpoint at auth-plugin.opennhp.org/plugins/example passes username and password as GET query parameters. GET parameters are logged in server access logs, browser history, proxy logs, and Referrer headers. The demo page JavaScript constructs the URL as: nhpValidUrl = '...?resid=demo&action=valid&username=' + encodeURIComponent(user) + '&password=' +

encodeURIComponent(password). While this uses fetch() (which mitigates browser history exposure), the credentials still appear in server logs.

Technical Analysis

Credentials are sent as URL query parameters in a GET request. These will be logged in nginx access logs, any intermediate proxy logs, and potentially exposed via Referrer headers. The demo uses user/password as default credentials.

Raw Request

```
GET /plugins/example?resid=demo&action=valid&username=user&password=password HTTP/2\nHost:
auth-plugin.opennhp.org
```

Raw Response

```
HTTP/2 200\n{\"errCode\":\"0\",\"resHost\":{\"demoServer\":\"ac.opennhp.org\"},\"opnTime\"
:15,\"agentAddr\":\"...\",\"acTokens\":{\"demoServer\":\"...\"},\"redirectUrl\":\"https://
ac.opennhp.org\"}
```

Remediation

Change the auth validation endpoint to accept credentials via POST body instead of GET query parameters. Use application/json or application/x-www-form-urlencoded content type in the request body.

12. [MEDIUM] Authentication Credentials Transmitted in URL Query Parameters (GET)

Severity	MEDIUM
CVSS	4.3 CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N
Host	server/auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example
Category	information_disclosure
Status	CONFIRMED
MITRE	T1552.001 Unsecured Credentials: Credentials In Files

Severity	MEDIUM
ATT&CK	
Detected by	go-web-agent

Description

The auth-plugin.opennhp.org authentication endpoint accepts credentials (username and password) as GET query parameters: /plugins/example?resid=demo&action=valid&username=USER&password=PASS. This is a security anti-pattern because GET parameters are logged in web server access logs, browser history, proxy logs, Referer headers, and can be cached. The password 'password' for user 'user' is transmitted in cleartext in the URL. While HTTPS encrypts the transport, the URL (including credentials) will be stored in server access logs (nginx), browser history, and may leak via the Referer header when navigating to the redirectUrl.

Technical Analysis

The authentication endpoint uses GET method exclusively (POST /plugins/example returns 404). This means credentials must be sent as URL query parameters. GET parameters are logged by web servers (nginx access.log), stored in browser history, visible in browser address bars, included in Referer headers sent to the redirectUrl target (ac.opennhp.org), and may be cached by proxies. Even over HTTPS, the full URL including password is recorded in multiple locations. This is a confirmed design issue.

Exploitation Commands

```
curl -sk "https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid&username
=user&password=password"
```

Raw Request

```
GET /plugins/example?resid=demo&action=valid&username=user&password=password HTTP/2
Host: auth-plugin.opennhp.org
```

Raw Response

```
HTTP/2 200
content-type: application/json; charset=utf-8
set-cookie: nhp-token=...; Path=/; Domain=opennhp.org; Max-Age=15; HttpOnly; Secure

{"errCode":"0","resHost":{"demoServer":"ac.opennhp.org"},"opnTime":15,"agentAddr":"IP_PUBL
IC_003","acTokens":{"demoServer":"..."},"redirectUrl":"https://ac.opennhp.org"}
```

Evidence / Logs

> GET /plugins/example?resid=demo&action=valid&username=user&password=password -> 200 with v
> alid auth response containing acTokens and nhp-token cookie
> The endpoint only accepts GET (POST returns 404), forcing credentials into the URL

Remediation

Implement the authentication endpoint as a POST method accepting credentials in the request body (JSON or form-encoded). Never transmit passwords in URL query parameters. Add Referrer-Policy: no-referrer header to prevent URL leakage via Referer.

13. [MEDIUM] Clickjacking: All Three Hosts Frameable (No X-Frame-Options / CSP frame-ancestors)

Severity	MEDIUM
CVSS	4.3 CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:L/A:N
Host	auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=login
Category	clickjacking
Status	CONFIRMED
MITRE ATT&CK	T1185 Browser Session Hijacking
Detected by	browser-playwright

Description

All three OpenNHP demo pages can be embedded in iframes from any origin. Playwright confirmed that iframes pointing to each host loaded successfully and rendered full page content including interactive login forms. The auth-plugin login page is particularly sensitive since it contains a login form with pre-filled credentials - an attacker could trick users into clicking a 'Sign In' button overlaid on their page, triggering authentication with crafted or pre-filled credentials.

Technical Analysis

Using Playwright, we created an attacker page that embedded all three OpenNHP hosts inside iframes. All three rendered fully without any framing restriction. The auth-plugin page is the most critical target because it contains a login form pre-filled with 'user/password' credentials and a 'Sign In' button. An attacker could overlay an invisible iframe on a malicious page and trick users into clicking the Sign In button, inadvertently authenticating and potentially performing actions on the NHP server.

Exploitation Commands

```
Playwright: page.setContent('<iframe src="https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=login"></iframe>')
Playwright: page.setContent('<iframe src="https://agent.opennhp.org"></iframe>')
Playwright: page.setContent('<iframe src="https://reg.opennhp.org"></iframe>')
```

Evidence / Logs

```
Clickjacking test results:
Frame URL: https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=login
Frame Title: OpenNHP Auth Plugin Demo - Authentication
FRAMEABLE: YES - No X-Frame-Options or CSP frame-ancestors blocking
Frame URL: https://agent.opennhp.org/
Frame Title: OpenNHP JavaScript SDK Demo
FRAMEABLE: YES
Frame URL: https://reg.opennhp.org/
Frame Title: OpenNHP Agent Registration
FRAMEABLE: YES
```

Remediation

Add X-Frame-Options: DENY (or SAMEORIGIN) header and CSP frame-ancestors directive to all three hosts. For the auth-plugin host, this is critical since it serves interactive authentication forms.

14. [MEDIUM] Login Pages Vulnerable to Clickjacking (No Frame Protection)

Severity	MEDIUM
CVSS	4.3 CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:L/A:N
Host	server.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=login, https://agent.opennhp.org, https://reg.opennhp.org

Severity	MEDIUM
Category	clickjacking
Status	CONFIRMED
MITRE ATT&CK	T1189 Drive-by Compromise
Detected by	headless-browser

Description

All three web application pages (auth-plugin login page, agent SDK demo, reg registration page) are frameable in iframes from any origin. No X-Frame-Options header and no Content-Security-Policy frame-ancestors directive are set. An attacker could overlay a transparent iframe over a malicious page to trick users into clicking on the login form or other interactive elements. The login page at auth-plugin.opennhp.org is the highest risk target as it handles authentication.

Technical Analysis

All pages can be embedded in iframes from any origin. The headless browser confirmed no X-Frame-Options or CSP frame-ancestors directives are present on any of the three web applications.

Remediation

Add X-Frame-Options: DENY or Content-Security-Policy: frame-ancestors 'none' to all page responses via nginx configuration.

15. [LOW] Unauthenticated Server Infrastructure Enumeration via /servers Endpoint

Severity	LOW
CVSS	3.7 CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N
Host	agent/reg/relay (3.149.105.15)
Endpoint	https://relay.opennhp.org/servers
Category	information_disclosure

Severity	LOW
Status	CONFIRMED
MITRE ATT&CK	T1018 Remote System Discovery
Detected by	go-web-agent

Description

The relay.opennhp.org /servers endpoint exposes a JSON list of all registered NHP servers without requiring authentication. Each entry includes the server ID (e.g. '7rFdWSvghe8'), display name (e.g. 'demo'), base64-encoded public key, cipher scheme, instance count, and load balancing strategy. While public keys are meant to be shared, the server IDs and topology information (number of instances, LB strategy) enable an attacker to enumerate the NHP infrastructure and craft targeted relay requests. The error messages from /relay/ also confirm the API structure ('use POST /relay/<serverId> (see GET /servers)'), providing a guided discovery path.

Technical Analysis

The /servers endpoint returns full server infrastructure details without authentication. The error message in /relay/ explicitly references /servers, confirming it as a documented API. The server IDs revealed allow direct relay routing via POST /relay/<serverId>. While the NHP protocol itself uses cryptographic authentication (public keys + cipher), exposing the topology and server IDs reduces the security boundary and allows enumeration of the infrastructure graph.

Exploitation Commands

```
curl -skD- "https://relay.opennhp.org/servers"
curl -skD- -X POST "https://relay.opennhp.org/relay/" -H "Content-Type: application/json"
-d '{}'
```

Raw Request

```
GET /servers HTTP/2
Host: relay.opennhp.org
```

Raw Response

```
HTTP/2 200
content-type: application/json
```

```
content-length: 345
access-control-allow-origin: *

[{"id":"7rFdWSvghe8","name":"demo","publicKeyBase64":"eVr33jqYo2nt5llEUeDu6gJNHxpnhfY0TRJ4
iVIFniM=","cipherScheme":0,"instanceCount":1,"loadBalance":"weighted-random"},{"id":"euySl
Pig-zM","name":"demo-cluster2","publicKeyBase64":"T8zaoyl5aUsct59WAoInBIZugKx6WVsliy+0lObS
2x4=","cipherScheme":0,"instanceCount":1,"loadBalance":"weighted-random"}]
```

Evidence / Logs

```
GET /servers -> 200 with JSON array containing 2 server entries
POST /relay/ -> 400 with "missing server id; use POST /relay/<serverId> (see GET /servers)
"
POST /relay/demoServer -> 404 with 'unknown server ID "demoServer"'
POST /relay/7rFdWSvghe8 with {} -> 400 "inner packet too short"
POST /relay/7rFdWSvghe8 with JSON -> 504 "NHP Server timeout" (confirms relay processing)
```

Remediation

Restrict the /servers endpoint with authentication or API key. In a zero-trust architecture like NHP, server topology should not be freely enumerable. Consider rate-limiting this endpoint or requiring a valid NHP session.

16. [LOW] CORS Misconfiguration: Access-Control-Allow-Origin: * with Access-Control-Allow-Credentials: true

Severity	LOW
CVSS	3.7 CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N
Host	auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid
Category	misconfiguration
Status	CONFIRMED
MITRE ATT&CK	T1189 Drive-by Compromise
Detected by	browser-playwright

Description

The auth-plugin.opennhp.org server returns both Access-Control-Allow-Origin: (wildcard) and Access-Control-Allow-Credentials: true simultaneously in its CORS response headers. Per the CORS specification (Fetch Standard), browsers will NOT send credentials when ACAO is a wildcard, so this specific combination is not directly exploitable. However, it indicates a CORS misconfiguration that could become dangerous if the server is later changed to reflect the Origin header instead of using a wildcard. The validation endpoint already exposes acTokens in its JSON response body which any origin can read (due to ACAO:) via a simple cross-origin fetch without credentials.

Technical Analysis

The server sends ACAO: * which allows any website to read its responses. Combined with ACAC: true (which browsers ignore with wildcard), this is a misconfiguration that indicates the developer intended to allow credentialed cross-origin requests but implemented it incorrectly. The immediate impact is limited because browsers block credentials with wildcard origin per spec. However, the wildcard ACAO alone still allows any malicious site to call the auth validation endpoint with known/guessed credentials and read the response including acTokens, without needing cookies. If the server ever switches to reflecting the Origin header, it would become a critical CORS vulnerability.

Exploitation Commands

```
curl -sI -H "Origin: https://evil.com" "https://auth-plugin.opennhp.org/plugins/example?re
sid=demo&action=login"
```

Raw Response

```
HTTP/2 200
access-control-allow-credentials: true
access-control-allow-origin: *
access-control-expose-headers: Content-Type, Content-Length, Set-Cookie
```

Evidence / Logs

```
HTTP Response Headers from auth-plugin.opennhp.org:
access-control-allow-credentials: true
access-control-allow-headers: Content-Type, Content-Length, Authorization, X-NHP-Ver, Cook
ie
access-control-allow-methods: GET, OPTIONS, POST
access-control-allow-origin: *
access-control-expose-headers: Content-Type, Content-Length, Set-Cookie
access-control-max-age: 300
```

Cross-origin fetch test (from agent.opennhp.org to auth-plugin.opennhp.org):
Status: 200, response body readable, acTokens exposed

Remediation

Replace Access-Control-Allow-Origin: * with explicit allowed origins (e.g., https://auth-plugin.opennhp.org, https://agent.opennhp.org). If credentials are needed, implement proper origin validation. Remove Access-Control-Allow-Credentials: true if not needed, or ensure it is only set with specific origins, never with wildcard.

17. [LOW] nhp-token Cookie Scoped to Parent Domain (.opennhp.org) - Shared Across All Subdomains

Severity	LOW
CVSS	3.7 CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N
Host	auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid
Category	misconfiguration
Status	CONFIRMED
MITRE ATT&CK	T1539 Steal Web Session Cookie
Detected by	browser-playwright

Description

After successful authentication on auth-plugin.opennhp.org, the nhp-token cookie is set with domain=.opennhp.org (parent domain). This means the authentication token is sent to ALL subdomains of opennhp.org (agent.opennhp.org, reg.opennhp.org, relay.opennhp.org, ac.opennhp.org, server.opennhp.org, etc.). If any subdomain is compromised or has an XSS vulnerability, the nhp-token can be intercepted. The cookie does have HttpOnly: true and Secure: true, and SameSite: Lax, which is positive. However, the broad scope unnecessarily increases exposure. The token value is a 32-byte opaque value (likely HMAC-SHA256), expires in approximately 15 minutes (matching opnTime=15 in the auth response).

Technical Analysis

After authenticating via the login form, the browser receives a Set-Cookie for nhp-token scoped to .opennhp.org. This was confirmed by inspecting cookies via Playwright context.cookies(). The parent domain scope means this token will be automatically included in requests to all subdomains (agent, reg, relay, ac, server, etc.). While HttpOnly and Secure flags are properly set, the overly broad domain scope increases the attack surface - any subdomain that handles the token incorrectly could lead to session hijacking.

Exploitation Commands

```
Playwright: context.cookies() after successful login
```

Evidence / Logs

```
Cookie after login:
Name: nhp-token
Value: wElypVaQF5%252FdggpMLwwtzPLQTSUdZMYbEI3PU5eZjPU%253D
Domain: .opennhp.org  (PARENT DOMAIN - shared across ALL subdomains)
Path: /
Expires: 2026-07-26T20:48:02.943Z (~15 min TTL)
HttpOnly: true (good - not accessible via JS)
Secure: true (good - HTTPS only)
SameSite: Lax (good - baseline CSRF protection)
Decoded token: wElypVaQF5/dggpMLwwtzPLQTSUdZMYbEI3PU5eZjPU= (32 bytes, SHA-256 or HMAC)
```

Remediation

Restrict the cookie domain to the specific subdomain that needs it (e.g., domain=ac.opennhp.org or no domain attribute at all to limit to the issuing host). If the token needs to be shared across subdomains, minimize the set of subdomains involved.

18. [LOW] CORS expose-headers Includes Set-Cookie (Attempted Cookie Exposure to Cross-Origin Callers)

Severity	LOW
CVSS	3.7 CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N
Host	auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid

Severity	LOW
Category	misconfiguration
Status	CONFIRMED
MITRE ATT&CK	T1539 Steal Web Session Cookie
Detected by	browser-playwright

Description

The auth-plugin.opennhp.org server includes 'Set-Cookie' in its Access-Control-Expose-Headers response. This header attempts to make the Set-Cookie header readable by cross-origin JavaScript via response.headers.get('set-cookie'). While browsers block reading Set-Cookie headers from JavaScript (it's a forbidden response header per the Fetch specification), the presence of this configuration indicates an intent to expose cookies cross-origin. Additionally, the acToken value in the JSON body is identical to the nhp-token cookie value, meaning the token is already exposed cross-origin via the response body regardless of the Set-Cookie header. The cookie has Max-Age=15 (15 seconds), creating a very short-lived access window.

Technical Analysis

The server explicitly lists Set-Cookie in its CORS expose-headers, attempting to make the authentication cookie readable cross-origin. While browsers block this for security reasons (Set-Cookie is a forbidden header), the same token value is already present in the JSON response body as acTokens, which IS readable cross-origin due to ACAO:*. The cookie Max-Age is only 15 seconds, creating a very narrow window. The identity between acToken and nhp-token means the token is functionally exposed via the response body even without reading the Set-Cookie header.

Exploitation Commands

```
curl -s -D - "https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid&user
name=user&password=password"
```

Raw Response

```
set-cookie: nhp-token=dGZTkNdmNBtOIGrHfQn4kR0yXGIWgtUeWPh1Pkjawco%253D; Path=/; Domain=ope
nnhp.org; Max-Age=15; HttpOnly; Secure
access-control-expose-headers: Content-Type, Content-Length, Set-Cookie
```

Evidence / Logs

```
Response headers:
access-control-expose-headers: Content-Type, Content-Length, Set-Cookie
set-cookie: nhp-token=dGZTkNdmNBtOIGrHfQn4kR0yXGIWgtUeWPh1Pkjawco%253D; Path=/; Domain=o
pennhp.org; Max-Age=15; HttpOnly; Secure
JSON body:
acTokens: {"demoServer":"dGZTkNdmNBtOIGrHfQn4kR0yXGIWgtUeWPh1Pkjawco="}
Note: acToken value matches nhp-token cookie value exactly!
```

Remediation

Remove Set-Cookie from Access-Control-Expose-Headers. Restrict ACAO to specific origins. Consider separating the acToken from the session cookie so that cross-origin token exposure doesn't directly leak the session credential.

19. [LOW] Missing Host Header Validation - Default Virtual Host Serves Auth Endpoint

Severity	LOW
CVSS	3.7 CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N
Host	server/auth-plugin.opennhp.org
Endpoint	https://3.149.215.44/plugins/example
Category	security_misconfiguration
Status	CONFIRMED
MITRE ATT&CK	T1557 Adversary-in-the-Middle
Detected by	go-web-agent

Description

The nginx reverse proxy on 3.149.215.44 does not validate the Host header. Requests with arbitrary Host headers (e.g., Host: internal.opennhp.org, Host: localhost, Host: 127.0.0.1) are routed to the same Go backend and successfully authenticate. This means the auth-plugin API is accessible via direct IP with any Host header, bypassing any Host-based access controls. The Go backend also processes the request regardless of Host,

suggesting it doesn't implement Host-based routing.

Technical Analysis

The nginx default server block does not reject requests with unexpected Host headers. This is confirmed by sending requests directly to the IP address (3.149.215.44) with fabricated Host headers - all are processed by the same backend. In a shared hosting or multi-tenant environment, this could enable host header injection attacks or DNS rebinding. For a zero-trust security framework, host validation is a fundamental requirement.

Exploitation Commands

```
curl -sk -H "Host: internal.opennhp.org" "https://3.149.215.44/plugins/example?resid=demo&
action=valid&username=user&password=password"
curl -sk -H "Host: localhost" "https://3.149.215.44/"
curl -sk -H "Host: 127.0.0.1" "https://3.149.215.44/"
```

Raw Request

```
GET /plugins/example?resid=demo&action=valid&username=user&password=password HTTP/2
Host: internal.opennhp.org
```

Raw Response

```
{"errCode":"0","resHost":{"demoServer":"ac.opennhp.org"},"opnTime":15,...}
```

Evidence / Logs

```
Host: internal.opennhp.org -> 200 with valid auth tokens
Host: localhost -> 200 with login page HTML
Host: 127.0.0.1 -> 200 with login page HTML
All return the same content as Host: auth-plugin.opennhp.org
```

Remediation

Configure nginx to reject requests with unrecognized Host headers. Add a default server block that returns 444 for unknown hosts: server { listen 443 ssl default_server; return 444; }. Also validate Host in the Go application.

20. [LOW] Unauthenticated Server Infrastructure Enumeration via /servers

Severity	LOW

Severity	LOW
CVSS	3.7 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
Host	server.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/servers
Category	information_disclosure
Status	CONFIRMED
MITRE ATT&CK	None
Detected by	golang

Description

The endpoint /servers on auth-plugin.opennhp.org (and server.opennhp.org, demologin.opennhp.org) is accessible without authentication and returns server infrastructure information. This enables enumeration of the NHP server cluster configuration by unauthenticated users.

Technical Analysis

The /servers endpoint returns infrastructure data without requiring authentication, enabling reconnaissance of the NHP server cluster topology.

Exploitation Commands

```
curl -sk https://auth-plugin.opennhp.org/servers
```

Remediation

Restrict the /servers endpoint to authenticated users or remove it from public-facing deployments.

21. [LOW] SSL Certificate Mismatch on server.opennhp.org

Severity	LOW
CVSS	3.1 CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N

Severity	LOW
Host	server.opennhp.org
Endpoint	https://server.opennhp.org:443
Category	ssl_tls_misconfiguration
Status	CONFIRMED
MITRE ATT&CK	None
Detected by	pentest-orchestrator

Description

The SSL certificate on server.opennhp.org (3.149.215.44:443) is issued to CN=auth-plugin.opennhp.org with SANs only for auth-plugin.opennhp.org and demologin.opennhp.org. When accessed as server.opennhp.org, the certificate CN does not match the requested hostname. Browsers will show a certificate warning. Nuclei flagged this as 'mismatched-ssl-certificate'.

Technical Analysis

The TLS certificate on 3.149.215.44 (server.opennhp.org) only covers auth-plugin.opennhp.org and demologin.opennhp.org in its SANs. server.opennhp.org is not listed.

Exploitation Commands

```
nuclei -u https://server.opennhp.org -severity info
```

Raw Response

```
mismatched-ssl-certificate server.opennhp.org:443 CN: auth-plugin.opennhp.org\nssl-dns-nam
es server.opennhp.org:443 auth-plugin.opennhp.org, demologin.opennhp.org
```

Remediation

Add server.opennhp.org to the SAN list of the Let's Encrypt certificate, or issue a separate certificate for this hostname.

22. [LOW] Missing Subresource Integrity (SRI) on External Scripts

Severity	LOW
CVSS	3.1 CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N
Host	server.opennhp.org
Endpoint	https://server.opennhp.org, https://agent.opennhp.org
Category	security_headers
Status	CONFIRMED
MITRE ATT&CK	None
Detected by	pentest-orchestrator

Description

External resources loaded from CDNs lack Subresource Integrity (SRI) attributes. On server.opennhp.org: Google Fonts CSS loaded without integrity. On agent.opennhp.org: highlight.js JS and CSS loaded from cdn.jsdelivr.net without integrity hash attributes. If the CDN is compromised, malicious code could be injected.

Technical Analysis

CDN-hosted scripts without SRI hashes are vulnerable to supply chain attacks if the CDN is compromised.

Exploitation Commands

```
nuclei -u https://agent.opennhp.org -severity info
```

Raw Response

```
missing-sri https://agent.opennhp.org/ https://cdn.jsdelivr.net/gh/highlightjs/cdn-release
@11.9.0/build/highlight.min.js, https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.9.0
/build/styles/github-dark.min.css
```

Remediation

Add integrity attributes to all external script and stylesheet tags. Example: <script src='...' integrity='sha384-...'

crossorigin='anonymous'>.

23. [LOW] Missing Security Headers on Auth API Responses

Severity	LOW
CVSS	3.1 CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:L/A:N
Host	server/auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example
Category	security_misconfiguration
Status	CONFIRMED
MITRE ATT&CK	T1185 Browser Session Hijacking
Detected by	go-web-agent

Description

The auth-plugin.opennhp.org and server.opennhp.org responses are missing critical security headers: Strict-Transport-Security (HSTS), X-Frame-Options, Content-Security-Policy, X-Content-Type-Options, Referrer-Policy, and Permissions-Policy. The absence of X-Frame-Options enables clickjacking attacks against the login page. The absence of HSTS allows SSL stripping. The absence of Referrer-Policy means the full URL (including credentials in query params) may leak via Referer header to third-party resources (fonts.googleapis.com). The relay endpoint (relay.opennhp.org) has x-content-type-options: nosniff but the auth endpoints do not.

Technical Analysis

A comprehensive scan of all response headers from auth-plugin.opennhp.org shows the complete absence of standard security headers: no Strict-Transport-Security, no X-Frame-Options, no Content-Security-Policy, no X-Content-Type-Options, no Referrer-Policy, and no Permissions-Policy. The login page embeds fonts from fonts.googleapis.com, meaning credentials in URL will leak via the Referer header to Google's CDN. The lack of X-Frame-Options means the login page can be iframed for clickjacking attacks.

Exploitation Commands

```
curl -skD- "https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid&userna
me=user&password=password" 2>&1 | grep -i -E '(strict-transport|x-frame|x-content|x-xss|co
ntent-security|referrer-policy|permissions-policy)'
```

Raw Request

```
GET /plugins/example?resid=demo&action=valid&username=user&password=password HTTP/2
Host: auth-plugin.opennhp.org
```

Raw Response

```
HTTP/2 200
server: nginx/1.30.2
content-type: application/json; charset=utf-8
access-control-allow-credentials: true
access-control-allow-origin: *
(NO Strict-Transport-Security, NO X-Frame-Options, NO Content-Security-Policy, NO X-Conten
t-Type-Options, NO Referrer-Policy)
```

Evidence / Logs

```
grep returned empty - no security headers found in auth-plugin.opennhp.org response
relay.opennhp.org has only x-content-type-options: nosniff
server.opennhp.org has no security headers
demologin.opennhp.org has no security headers
```

Remediation

Add the following headers to all responses: Strict-Transport-Security: max-age=31536000; includeSubDomains; preload, X-Frame-Options: DENY, Content-Security-Policy: default-src 'self', X-Content-Type-Options: nosniff, Referrer-Policy: no-referrer, Permissions-Policy: camera=(), microphone=(), geolocation=()

24. [LOW] No CSRF Protection on Authentication Login Form

Severity	LOW
CVSS	3.1 CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:L/A:N
Host	auth-plugin.opennhp.org

Severity	LOW
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=login
Category	csrf
Status	CONFIRMED
MITRE ATT&CK	T1185 Browser Session Hijacking
Detected by	browser-playwright

Description

The login form on auth-plugin.opennhp.org has no CSRF protection - no anti-CSRF token, no hidden inputs with nonces, no CSRF meta tags. The form uses onsubmit='return nhpValidate()' which triggers a GET fetch() call. However, the practical impact is limited because: (1) this is a login form with pre-filled demo credentials, (2) the fetch-based submission is cross-origin-read-protected for attackers (though the ACAO:* header actually allows reading the response), and (3) the nhp-token cookie has SameSite=Lax which provides baseline CSRF protection for the subsequent authenticated session. The lack of a CSRF token on the login form itself means a login CSRF attack is theoretically possible.

Technical Analysis

DOM inspection of the login form confirms there are no CSRF tokens, no hidden fields, and no CSRF meta tags. The form is submitted via JavaScript fetch() with GET method, not a traditional form POST. While the SameSite=Lax on the nhp-token cookie mitigates post-login CSRF, the login itself is unprotected. This could allow login CSRF attacks where an attacker forces a victim to authenticate as the attacker's account.

Exploitation Commands

```
Playwright: page.evaluate(() => { const form = document.getElementById('loginForm'); retur
n { hiddenInputs: form.querySelectorAll('input[type=hidden]').length, csrfMeta: document.q
uerySelector('meta[name*=csrf]'), onsubmit: form.getAttribute('onsubmit') }; })
```

Evidence / Logs

```
CSRF Protection Check:
formAction: https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=login
formMethod: get
hasOnsubmit: true (return nhpValidate())
```

```
hiddenInputs: [] (empty - no CSRF token)
csrfMeta: null
hasCsrf: false
```

Remediation

Add a CSRF token to the login form, either as a hidden input or via a custom header in the fetch() request. Consider implementing the Synchronizer Token Pattern or Double Submit Cookie pattern.

25. [LOW] Hardcoded Default Credentials Pre-filled in Login Form (user/password)

Severity	LOW
CVSS	3.1 CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N
Host	auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=login
Category	information_disclosure
Status	CONFIRMED
MITRE ATT&CK	T1078.001 Valid Accounts: Default Accounts
Detected by	browser-playwright

Description

The auth-plugin login form has pre-filled username='user' and password='password' in the HTML input fields. While this is expected for a demo application, these credentials successfully authenticate against the server and grant access to the protected resource at ac.opennhp.org. The demo nature of this application limits the severity, but the credentials are hardcoded in the HTML source and visible to anyone who views the page source.

Technical Analysis

The login form HTML contains pre-filled value attributes for both username ('user') and password ('password') fields. Submitting these credentials via the form results in a successful authentication response with a redirect

to the protected resource. This is by design for the demo but represents an insecure pattern if replicated in production.

Exploitation Commands

```
Playwright: page.inputValue('#username') -> 'user'
Playwright: page.inputValue('#password') -> 'password'
```

Evidence / Logs

```
HTML source:
<input type="text" id="username" name="username" value="user" autocomplete="username">
<input type="password" id="password" name="password" value="password" autocomplete="curren
t-password">
Login with these credentials succeeded:
{"errCode":"0","redirectUrl":"https://ac.opennhp.org","acTokens":{"demoServer":"..."}}
```

Remediation

For production deployments, remove pre-filled credentials from form fields. Use placeholder attributes instead of value attributes for hint text.

26. [LOW] External JavaScript Dependencies Loaded from CDN Without Subresource Integrity (SRI)

Severity	LOW
CVSS	3.1 CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:L/A:N
Host	agent.opennhp.org
Endpoint	https://agent.opennhp.org
Category	misconfiguration
Status	CONFIRMED
MITRE ATT&CK	T1195.002 Supply Chain Compromise: Compromise Software Supply Chain
Detected by	browser-playwright

Description

The agent.opennhp.org and reg.opennhp.org pages load JavaScript from external CDNs (jsdelivr.net, esm.sh) without Subresource Integrity (SRI) hashes. Specifically: (1) highlight.js from cdn.jsdelivr.net (agent page), (2) @noble/ciphers, @noble/curves, @noble/hashes from esm.sh (both pages via importmap), (3) cbor-x from esm.sh (both pages). If any of these CDNs were compromised, malicious JavaScript could execute in the context of the NHP pages, potentially stealing cryptographic keys from the config.json or intercepting NHP knock operations. The noble crypto libraries are particularly sensitive since they handle the Curve25519/SM2 cryptographic operations.

Technical Analysis

Inspection of the HTML source reveals external JavaScript loaded from CDNs without SRI integrity attributes. These libraries handle sensitive cryptographic operations (key generation, ECDH, AEAD encryption) for the NHP protocol. If the CDN were compromised, an attacker could replace the noble crypto libraries with backdoored versions that leak private keys or weaken encryption. Without SRI, browsers have no way to verify the integrity of the loaded scripts. Note that SRI is not supported for import maps in all browsers, which is a known limitation.

Exploitation Commands

```
Playwright: page.evaluate(() => Array.from(document.querySelectorAll('script[src]')).map(s
=> s.src))
```

Evidence / Logs

```
External scripts without SRI:
https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.9.0/build/highlight.min.js (no in
tegrity attribute)
Import map entries (loaded dynamically without SRI):
@noble/ciphers -> https://esm.sh/@noble/ciphers@2.1.1
@noble/curves -> https://esm.sh/@noble/curves@2.0.1
@noble/hashes -> https://esm.sh/@noble/hashes@1.7.2
cbor-x -> https://esm.sh/cbor-x@1.6.4
No <script> tags have integrity="" attribute.
```

Remediation

Add integrity attributes to all external script tags where browser support allows. For import maps where SRI is not yet supported, consider self-hosting the crypto libraries from the same origin. Add Content-Security-Policy with strict script-src that limits allowed sources.

27. [LOW] Login Pages Vulnerable to Clickjacking (No X-Frame-Options/CSP)

Severity	LOW
CVSS	3.1 CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:L/A:N
Host	server/auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/
Category	security_misconfiguration
Status	CONFIRMED
MITRE ATT&CK	T1185 Browser Session Hijacking
Detected by	go-web-agent

Description

All three login pages (auth-plugin.opennhp.org, server.opennhp.org, demologin.opennhp.org) lack X-Frame-Options and Content-Security-Policy frame-ancestors headers, making them embeddable in iframes on any domain. An attacker can create a page that iframes the login form with an invisible overlay, tricking users into entering their NHP credentials on a malicious site (clickjacking/UI redressing). Combined with the CORS origin reflection vulnerability, a successful clickjacking attack could capture entered credentials via cross-origin JavaScript.

Technical Analysis

The absence of both X-Frame-Options and Content-Security-Policy frame-ancestors headers on all login pages was confirmed by inspecting the full HTTP response headers. Neither nginx nor the Go backend sets these headers. This allows any external website to embed the login pages in an iframe. Since the login form is a standard HTML form with username and password inputs, a clickjacking attack is feasible where a user is tricked into typing credentials while thinking they're interacting with a different interface.

Exploitation Commands

```
curl -skD- "https://auth-plugin.opennhp.org/" 2>&1 | grep -i -E '(x-frame|frame-ancestors|
content-security-policy)'
curl -skD- "https://server.opennhp.org/" 2>&1 | grep -i -E '(x-frame|frame-ancestors|conte
```

```
nt-security-policy)'
curl -skD- "https://demologin.opennhp.org/" 2>&1 | grep -i -E '(x-frame|frame-ancestors|co
ntent-security-policy)'
```

Raw Request

```
GET / HTTP/2
Host: auth-plugin.opennhp.org
```

Raw Response

```
HTTP/2 200
server: nginx/1.30.2
content-type: text/html; charset=utf-8
access-control-allow-credentials: true
access-control-allow-origin: *
(NO X-Frame-Options header)
(NO Content-Security-Policy header)
```

Evidence / Logs

```
All three grep commands returned empty - no X-Frame-Options or CSP frame-ancestors header
present on any login page.
auth-plugin.opennhp.org: No X-Frame-Options, No CSP
server.opennhp.org: No X-Frame-Options, No CSP
demologin.opennhp.org: No X-Frame-Options, No CSP
```

Remediation

Add X-Frame-Options: DENY or Content-Security-Policy: frame-ancestors 'none' to all login pages. This can be configured in nginx: add_header X-Frame-Options DENY always; add_header Content-Security-Policy "frame-ancestors 'none'" always;

28. [LOW] No CSRF Protection on Login Form

Severity	LOW
CVSS	3.1 CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:L/A:N
Host	server.opennhp.org

Severity	LOW
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=login
Category	csrf
Status	CONFIRMED
MITRE ATT&CK	None
Detected by	headless-browser

Description

The login form on the auth demo page does not include any anti-CSRF tokens. The authentication is performed via a JavaScript fetch() call to a GET endpoint, which is not protected by CSRF tokens. Modern browsers mitigate this via SameSite=Lax default cookie behavior, but the absence of explicit CSRF protection is a defense-in-depth gap. Combined with the CORS wildcard origin, a cross-origin page could initiate the login flow.

Technical Analysis

The login form uses onsubmit='return nhpValidate()' which calls fetch() with no CSRF token. Mitigated by SameSite=Lax default in modern browsers, but older browsers remain vulnerable.

Remediation

Add a CSRF token to the authentication flow. Consider using the Double-Submit Cookie pattern or Synchronizer Token pattern.

29. [LOW] NHP-Token Cookie Missing SameSite Attribute

Severity	LOW
CVSS	2.4 CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:L/A:N/E:U/RL:O/RC:C
Host	server/auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example
Category	security_misconfiguration

Severity	LOW
Status	CONFIRMED
MITRE ATT&CK	T1185 Browser Session Hijacking
Detected by	go-web-agent

Description

The nhp-token cookie is set without a SameSite attribute: 'set-cookie: nhp-token=...; Path=/; Domain=opennhp.org; Max-Age=15; HttpOnly; Secure'. While the cookie has HttpOnly and Secure flags (good), the absence of an explicit SameSite attribute means the browser defaults to SameSite=Lax. For maximum CSRF protection, SameSite=Strict should be explicitly set. Combined with the CORS wildcard + credentials vulnerability, the missing SameSite attribute means cross-origin requests from any domain can include this cookie in GET requests (Lax default allows top-level navigation).

Technical Analysis

Analysis of the Set-Cookie header shows the nhp-token cookie includes HttpOnly (prevents JS access) and Secure (HTTPS only) flags, but lacks an explicit SameSite attribute. Without SameSite, modern browsers default to Lax, which allows the cookie to be sent on cross-site top-level navigations (GET requests). This means a link from an attacker's page to the auth endpoint would include the cookie. For a zero-trust security framework like NHP, the cookie should use SameSite=Strict to prevent any cross-site cookie inclusion.

Exploitation Commands

```
curl -skD- "https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid&userna
me=user&password=password" 2>&1 | grep -i 'set-cookie'
```

Raw Request

```
GET /plugins/example?resid=demo&action=valid&username=user&password=password HTTP/2
Host: auth-plugin.opennhp.org
```

Raw Response

```
set-cookie: nhp-token=QAIzfqtNNOahMa7wCr0IhyTynTRa%252BveK4NxXzQdBdjA%253D; Path=/; Domain
=opennhp.org; Max-Age=15; HttpOnly; Secure
```

Evidence / Logs

```
set-cookie: nhp-token=QAIzfqtNNOahMa7wCr0IhyTynTRa%252BveK4NxXzQdBdjA%253D; Path=/; Domain
=opennhp.org; Max-Age=15; HttpOnly; Secure
Note: No SameSite attribute present in the Set-Cookie header
```

Remediation

Add SameSite=Strict to the nhp-token cookie. In Go: http.Cookie{SameSite: http.SameSiteStrictMode}. This prevents the cookie from being sent on any cross-site request.

30. [LOW] NHP-Token Cookie Missing SameSite Attribute and Scoped to Parent Domain

Severity	LOW
CVSS	2.4 CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N
Host	server.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example
Category	session_management
Status	CONFIRMED
MITRE ATT&CK	None
Detected by	golang

Description

The nhp-token cookie set by the auth endpoint uses Domain=opennhp.org (parent domain) which makes it available to all *.opennhp.org subdomains. The cookie lacks an explicit SameSite attribute (defaults to Lax in modern browsers). It is HttpOnly and Secure. The cookie value appears to be double percent-encoded. The Max-Age is 15 seconds (by NHP design for time-limited access), which limits the exploitation window but is still a defense-in-depth concern.

Technical Analysis

The cookie is scoped to the parent domain opennhp.org, meaning any subdomain can access it. The value is double percent-encoded (%253D instead of %3D). The missing SameSite attribute defaults to Lax in modern browsers but could be exploited in older browsers.

Exploitation Commands

```
curl -sk -v "https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid&usern
ame=user&password=password" 2>&1 | grep Set-Cookie
```

Raw Response

```
Set-Cookie: nhp-token=UqmqA16jhWxCQ5Ng4aVLf5vE1qrUyGIo3n0PG3CMyEs%253D; Path=/; Domain=ope
nnhp.org; Max-Age=15; HttpOnly; Secure
```

Remediation

Add SameSite=Strict to the cookie. Scope it to the specific subdomain rather than the parent domain. Fix the double percent-encoding of the cookie value.

31. [LOW] NHP-Token Cookie Double Percent-Encoding

Severity	LOW
CVSS	2.0 CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:L/A:N
Host	server/auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example
Category	security_misconfiguration
Status	CONFIRMED
MITRE ATT&CK	T1550.004 Use Alternate Authentication Material: Web Session Cookie
Detected by	go-web-agent

Description

The nhp-token cookie set by the auth endpoint is double percent-encoded. For example, the base64 '=' character (which should be '%3D' in URL encoding) appears as '%253D', and '+' appears as '%252B'. This indicates the server URL-encodes the cookie value twice before setting it. This can cause cookie parsing issues in clients that correctly decode only once, potentially leading to token validation failures or token confusion attacks where a client sends a partially-decoded value that the server interprets differently.

Technical Analysis

The Set-Cookie header contains double percent-encoded values. The base64 token value should contain '=' characters (base64 padding), which would normally be URL-encoded as '%3D' in a cookie. Instead, they appear as '%253D' (the '%' in '%3D' was itself encoded to '%25', resulting in '%253D'). This double encoding means a browser will decode it once to '%3D', and when sending the cookie back, the server receives '%3D' instead of '='. This is a confirmed implementation bug in the cookie-setting logic.

Exploitation Commands

```
curl -skD- "https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid&userna
me=user&password=password" 2>&1 | grep set-cookie
```

Raw Request

```
GET /plugins/example?resid=demo&action=valid&username=user&password=password HTTP/2
Host: auth-plugin.opennhp.org
```

Raw Response

```
set-cookie: nhp-token=vaEhI78TsLEwdHb1AEofca8MrrHmijyduD45Lt9gJL4%253D; Path=/; Domain=ope
nnhp.org; Max-Age=15; HttpOnly; Secure
```

Evidence / Logs

```
set-cookie: nhp-token=vaEhI78TsLEwdHb1AEofca8MrrHmijyduD45Lt9gJL4%253D; Path=/; Domain=ope
nnhp.org; Max-Age=15; HttpOnly; Secure
Note: %253D = %25 + 3D = literal '%3D' which is URL-encoded '=' - this is double encoding
Similarly %252B = %25 + 2B = literal '%2B' = double-encoded '+'
```

Remediation

Fix the cookie value encoding to use single URL encoding. Use Go's net/http cookie handling which properly encodes values, or use base64url encoding (RFC 4648 §5) which avoids '+', '/', and '=' characters entirely.

32. [INFO] NHP-Protected Hosts Confirmed Undiscoverable (Zero Attack Surface)

Severity	INFO
CVSS	0.0 -
Host	server2.opennhp.org (NHP-protected)
Endpoint	server2.opennhp.org, ac2.opennhp.org, ac.opennhp.org
Category	network_security
Status	CONFIRMED
MITRE ATT&CK	None
Detected by	pentest-orchestrator

Description

Three NHP-protected hosts (server2.opennhp.org/3.19.235.59, ac2.opennhp.org/3.151.110.18, ac.opennhp.org/3.151.225.91) were verified as completely undiscoverable using multiple techniques: curl HTTP/HTTPS (all timed out, exit 28), naabu port scan on ports 22,80,443,8080 (zero ports found), and bash /dev/tcp probes on ports 22,80,443 (all timed out, exit 124). This confirms the NHP SPA (Single Packet Authorization) zero-trust model is working as designed - these hosts present zero attack surface to unauthorized scanners.

Technical Analysis

All three protected hosts showed zero response to any probe technique. No TCP ports open, no ICMP response, no HTTP response. The hosts are effectively invisible on the network, confirming NHP SPA is operational.

Exploitation Commands

```
curl -sk --max-time 5 https://server2.opennhp.org (EXIT:28 timeout)
curl -sk --max-time 5 https://ac2.opennhp.org (EXIT:28 timeout)
curl -sk --max-time 5 https://ac.opennhp.org (EXIT:28 timeout)
naabu -host 3.19.235.59 -p 22,80,443,8080 (0 ports found)
naabu -host 3.151.110.18 -p 22,80,443,8080 (0 ports found)
naabu -host 3.151.225.91 -p 22,80,443,8080 (0 ports found)
timeout 5 bash -c "echo >/dev/tcp/3.19.235.59/80" (EXIT:124)
timeout 5 bash -c "echo >/dev/tcp/3.151.110.18/443" (EXIT:124)
```

```
timeout 5 bash -c "echo >/dev/tcp/3.151.225.91/22" (EXIT:124)
```

Remediation

No action needed - this is the expected behavior of NHP-protected hosts.

33. [INFO] Nginx Version Disclosure on All Exposed Hosts

Severity	INFO
CVSS	0.0 -
Host	server.opennhp.org
Endpoint	https://server.opennhp.org, https://agent.opennhp.org, https://reg.opennhp.org, https://relay.opennhp.org, https://demologin.opennhp.org, https://auth-plugin.opennhp.org
Category	information_disclosure
Status	CONFIRMED
MITRE ATT&CK	None
Detected by	pentest-orchestrator

Description

All exposed hosts disclose the exact nginx version (nginx/1.30.2) in the Server response header. While this is a low-risk informational finding, it aids attackers in fingerprinting the infrastructure and searching for known vulnerabilities in specific versions.

Technical Analysis

The Server header reveals the exact software and version, aiding targeted vulnerability research.

Exploitation Commands

```
curl -sk -I https://server.opennhp.org
```

Raw Response

```
server: nginx/1.30.2
```

Remediation

Add 'server_tokens off;' to nginx configuration to suppress version information.

34. [INFO] NHP Version and Build Hash Disclosed in Custom Header

Severity	INFO
CVSS	0.0 -
Host	server.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example
Category	information_disclosure
Status	CONFIRMED
MITRE ATT&CK	None
Detected by	pentest-orchestrator

Description

The auth-plugin API endpoints (server.opennhp.org, auth-plugin.opennhp.org, demologin.opennhp.org) expose a custom HTTP header 'access-control-nhp-ver' that reveals the exact NHP version (0.8.0.260720075832) and full git commit hash (ae05338caee146ada0208d1d84375fa9e20975d4). This provides precise software version identification to attackers.

Technical Analysis

The custom CORS header reveals the exact NHP build version and git commit hash. This is likely intentional for the public demo to show the running version.

Raw Response

```
access-control-nhp-ver: 0.8.0.260720075832/ae05338caee146ada0208d1d84375fa9e20975d4
```

Remediation

Consider removing the version/commit hash from response headers in production deployments. For a public demo, this may be acceptable.

35. [INFO] Cryptographic Keys Exposed in Public config.json (By Design)

Severity	INFO
CVSS	0.0 -
Host	agent.opennhp.org
Endpoint	https://agent.opennhp.org/config.json
Category	information_disclosure
Status	CONFIRMED
MITRE ATT&CK	None
Detected by	pentest-orchestrator

Description

The file https://agent.opennhp.org/config.json exposes NHP cryptographic key material including agentPrivateKey, agentPublicKey, serverPublicKey, and serverPublicKeySM2 for both Cluster 1 and Cluster 2. The file contains a comment: '_comment: PUBLIC DEMO CONFIG - agentPrivateKey is intentionally public; see deploy-demo-v2.yml.' This is by-design for the public demo and documented. The keys exposed are: Cluster 1 agentPrivateKey=1vlJ3OKc83FT2bwMcF3M0BtaNs7xSj4QBW2wdt2mU7E=, Cluster 2 agentPrivateKey=IFgOgHRnVSUAJgzPaxsdhGLxwK2hsav4cXDgnqFhx6w=.

Technical Analysis

The config.json file is served statically by nginx and contains full NHP agent private keys. The file explicitly marks these as intentionally public for the demo. In a production deployment, this would be a critical finding (private key exposure), but for this demo it is by-design and rated as info.

Exploitation Commands

```
curl -sk https://agent.opennhp.org/config.json
```

Raw Response


```
{\"_comment\":\"PUBLIC DEMO CONFIG - agentPrivateKey is intentionally public; see deploy-demo-v2.yml.\",\"cipherScheme\":\"curve25519\",\"transport\":\"relay\",\"relayUrl\":\"https://relay.opennhp.org/relay\",\"serverPublicKey\":\"eVr33jqYo2nt5lIEUeDu6gJNHxpnhfY0TRJ4iVIFniM=\",\"agentPrivateKey\":\"1vIJ3OKc83FT2bwMcF3M0BtaNs7xSj4QBW2wdt2mU7E=\",...}
```


Remediation

Ensure production deployments never serve private keys via static web files. Use environment variables or secure vaults for key management.

36. [INFO] OpenSSH 8.7 Version Disclosure on Exposed Host

Severity	INFO
CVSS	0.0 -
Host	agent.opennhp.org
Endpoint	agent.opennhp.org:22
Category	information_disclosure
Status	CONFIRMED
MITRE ATT&CK	None
Detected by	pentest-orchestrator

Description

The SSH service on agent.opennhp.org (3.149.105.15:22) and reg.opennhp.org, relay.opennhp.org (same IP) discloses OpenSSH 8.7. The SSH server accepts publickey, gssapi-keyex, and gssapi-with-mic authentication methods. SHA1-based HMAC algorithms are also enabled.

Technical Analysis

SSH version disclosure aids in targeting known vulnerabilities. Password authentication is disabled (good), but SHA1 HMAC algorithms are still enabled (weak).

Exploitation Commands

```
nuclei -u https://agent.opennhp.org -severity info
```

Raw Response

```
ssh-server-enumeration agent.opennhp.org:22 SSH-2.0-OpenSSH_8.7\nssh-auth-methods agent.op
ennhp.org:22 [\"publickey\",\"gssapi-keyex\",\"gssapi-with-mic\"]\nssh-sha1-hmac-algo agen
t.opennhp.org:22
```

Remediation

Disable SHA1-based HMAC algorithms in sshd_config. Consider hiding the SSH banner version if possible.

37. [INFO] NHP Version and Git Commit Hash Disclosed in HTTP Headers

Severity	INFO
CVSS	0.0 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N
Host	server/auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/
Category	information_disclosure
Status	CONFIRMED
MITRE ATT&CK	T1592 Gather Victim Host Information
Detected by	go-web-agent

Description

All responses from auth-plugin.opennhp.org and server.opennhp.org include a custom header 'access-control-nhp-ver' that reveals the exact NHP software version (0.8.0), build timestamp (260720075832 = July 20, 2026

07:58:32), and the full Git commit hash (ae05338caee146ada0208d1d84375fa9e20975d4). This information helps an attacker identify the exact source code revision to look for known vulnerabilities.

Technical Analysis

The custom HTTP header 'access-control-nhp-ver' is sent on every response. It contains the full semver version, a build timestamp, and the complete Git commit SHA. While not directly exploitable, this narrows the attack surface for an adversary who can clone the OpenNHP repository and diff that exact commit for vulnerabilities.

Exploitation Commands

```
curl -skD- "https://auth-plugin.opennhp.org/" -o /dev/null
```

Raw Request

```
GET / HTTP/2
Host: auth-plugin.opennhp.org
```

Raw Response

```
HTTP/2 200
server: nginx/1.30.2
access-control-nhp-ver: 0.8.0.260720075832/ae05338caee146ada0208d1d84375fa9e20975d4
```

Evidence / Logs

```
access-control-nhp-ver: 0.8.0.260720075832/ae05338caee146ada0208d1d84375fa9e20975d4
```

Remediation

Remove or obfuscate the access-control-nhp-ver header in production. Build metadata should not be exposed to external clients.

38. [INFO] Server Version and Build Hash Disclosure via Custom CORS Header (access-control-nhp-ver)

Severity	INFO
CVSS	0.0 -

Severity	INFO
Host	auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=login
Category	information_disclosure
Status	CONFIRMED
MITRE ATT&CK	T1592.004 Gather Victim Host Information: Client Configurations
Detected by	browser-playwright

Description

The auth-plugin.opennhp.org server exposes a custom CORS header 'access-control-nhp-ver' containing the exact build version (0.8.0.260720075832) and the full Git commit hash (ae05338caee146ada0208d1d84375fa9e20975d4). This information uniquely identifies the exact code revision running on the server. Additionally, the nginx/1.30.2 version is exposed via the standard 'server' header. On the agent SDK page, the SDK version 'v0.8.0' is also displayed in the UI.

Technical Analysis

The custom access-control-nhp-ver header leaks the exact build version and Git commit hash. This information could be used by attackers to identify the exact source code revision, check for known vulnerabilities in that specific version, or cross-reference with the public GitHub repository to find security-relevant changes. The nginx version is also exposed, though this is common. For an open-source project, this is lower severity since the code is public, but it still reveals the exact deployment version.

Exploitation Commands

```
curl -sI https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=login
```

Evidence / Logs

```
Response Headers:
access-control-nhp-ver: 0.8.0.260720075832/ae05338caee146ada0208d1d84375fa9e20975d4
server: nginx/1.30.2
Breakdown:
NHP Version: 0.8.0
```

Build timestamp: 260720075832
Git commit: ae05338caee146ada0208d1d84375fa9e20975d4

Remediation

Consider removing the access-control-nhp-ver header from production responses, or at minimum remove the Git commit hash. Hide the nginx version by adding 'server_tokens off' to the nginx configuration.

39. [INFO] Demo Cryptographic Private Keys Exposed in config.json (Intentionally Public)

Severity	INFO
CVSS	0.0 -
Host	agent.opennhp.org
Endpoint	https://agent.opennhp.org/config.json
Category	information_disclosure
Status	CONFIRMED
MITRE ATT&CK	T1552.004 Unsecured Credentials: Private Keys
Detected by	browser-playwright

Description

The file https://agent.opennhp.org/config.json is publicly accessible and contains NHP agent private keys, server public keys, and cluster configuration for two server clusters. The config.json file includes a comment explicitly stating '_comment: PUBLIC DEMO CONFIG - agentPrivateKey is intentionally public; see deploy-demo-v2.yml'. The keys are Curve25519 and SM2 keys used for the NHP knock protocol. These are demo keys intentionally published for the public demo, not production secrets. The config.json includes Cache-Control: no-store header.

Technical Analysis

The config.json file at agent.opennhp.org exposes cryptographic private keys for the NHP agent. These keys are used to perform the NHP 'knock' protocol which grants temporary access to protected servers. The file itself

contains a comment explicitly marking these as intentionally public demo keys. The keys are also populated into form fields on the SDK demo page. While these are demo keys, in a production scenario, exposing agent private keys would allow unauthorized users to impersonate the agent and gain access to protected resources.

Exploitation Commands

```
Playwright: page.evaluate(async () => { const res = await fetch('/config.json'); return aw
ait res.json(); })
```

Evidence / Logs

```
config.json contents include:
agentPrivateKey: "1vIJ3OKc83FT2bwMcF3M0BtaNs7xSj4QBW2wdt2mU7E=" (Cluster 1)
agentPublicKey: "jrzTQZsh1pO7vMB+b8cgvKRI4v0ziQdHdGCpJpntzSs="
serverPublicKey: "eVr33jqYo2nt5lIEUeDu6gJNHxpnhfY0TRJ4iVIFniM="
agentPrivateKey: "IFgOgHRnVSUAJgzPaxsdhGLxwK2hsav4cXDgnqFhx6w=" (Cluster 2)
_comment: "PUBLIC DEMO CONFIG - agentPrivateKey is intentionally public"
```

Remediation

For production deployments: never serve private keys via static files. Use environment variables, key management services, or per-user key provisioning. The demo approach is acceptable for demonstration purposes but should have prominent warnings about not using these keys in production.

40. [INFO] innerHTML Usage with IP Address from External API (ipify.org) - Limited DOM XSS Sink

Severity	INFO
CVSS	0.0 -
Host	auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=login
Category	xss_dom
Status	UNCONFIRMED
MITRE ATT&CK	T1059.007 Command and Scripting Interpreter: JavaScript

Severity	INFO
Detected by	browser-playwright

Description

Technical Analysis

DOM analysis identified 8 innerHTML assignments across the three pages. Most are safe: translation strings from hardcoded objects, element clearing operations, or escaped content via escapeHtml(). The only potentially exploitable sink is the IP display which uses template literal interpolation with data from ipify.org. However, this would require compromising a third-party API to exploit, making it a theoretical rather than practical risk. No user-controlled data reaches any innerHTML sink through URL parameters, form inputs, or hash fragments.

Exploitation Commands

```
Playwright: page.evaluate() - analyzed all innerHTML assignments in inline JavaScript
```

Evidence / Logs

```
innerHTML sinks found:
Auth-plugin page:
1. document.getElementById('scanHint').innerHTML = t.scanHint; (hardcoded translation st
rings - not exploitable)
2. ipDisplay.innerHTML = `IPv4: ${ipv4} | IPv6: ${ipv6}`; (external API response - theor
etical risk)
Agent page:
3. el.innerHTML = t(el.getAttribute('data-i18n-html')); (i18n translations - hardcoded)
4. logEl.innerHTML += `<span>...${escapeHtml(msg)}</span>`; (escaped - safe)
5. sel.innerHTML = ''; (clearing element - safe)
6. logEl.innerHTML = ''; (clearing element - safe)
```

Remediation

Replace innerHTML with textContent for the IP display since it only contains text content. Use textContent instead of innerHTML whenever the content doesn't need HTML rendering.

41. [INFO] Unvalidated Redirect from Server-Controlled redirectUrl (No Client-Side Validation)

Severity	INFO

Severity	INFO
CVSS	0.0 -
Host	auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid
Category	open_redirect
Status	UNCONFIRMED
MITRE ATT&CK	T1036 Masquerading
Detected by	browser-playwright

Description

After successful authentication, the client-side code executes window.location.href = result.redirectUrl without any validation of the URL. The redirectUrl value comes from the server's JSON response. Testing confirmed that the server correctly validates the resid parameter and only returns known redirect URLs - requests with resid=evil.com or resid=javascript:alert(1) return 'resource error: failed to find resource'. Therefore, while the client lacks validation, the server-side control prevents open redirect exploitation in the current implementation. The lack of client-side URL validation remains a defense-in-depth concern.

Technical Analysis

The client-side JavaScript blindly trusts the redirectUrl from the server response and sets window.location.href to it. However, testing with various resid values confirms the server validates this parameter and only returns redirect URLs for known resources. An open redirect would only be possible if the server's resource configuration were compromised to include a malicious URL, or if a new resource with an attacker-controlled redirect URL were registered.

Exploitation Commands

```
Playwright: fetch with resid=evil.com -> "resource error: failed to find resource"
Playwright: fetch with resid=javascript:alert(1) -> "resource error: failed to find resour
ce"
```

Evidence / Logs

```
Open redirect test results:
resid=demo: {"redirectUrl":"https://ac.opennhp.org"} (valid)
resid=evil.com: {"errMsg":"resource error: failed to find resource"} (blocked by server)
resid=javascript:alert(1): {"errMsg":"resource error: failed to find resource"} (blocked
by server)
Client-side code (no URL validation):
window.location.href = result.redirectUrl;
```

Remediation

Add client-side validation of the redirectUrl: verify it matches an expected domain pattern (e.g., *.opennhp.org) before redirecting. This provides defense-in-depth even if server-side validation is present.

42. [INFO] Cryptographic Private Keys Populated in Editable DOM Input Fields

Severity	INFO
CVSS	0.0 -
Host	agent.opennhp.org
Endpoint	https://agent.opennhp.org
Category	information_disclosure
Status	CONFIRMED
MITRE ATT&CK	T1552.004 Unsecured Credentials: Private Keys
Detected by	browser-playwright

Description

The agent.opennhp.org SDK demo page populates cryptographic private keys from config.json into editable text input fields in the DOM. The agentPrivKey input field contains the Curve25519 private key '1vIJ3OKc83FT2bwMcF3M0BtaNs7xSj4QBW2wdt2mU7E=' directly in the DOM as a plaintext input value. This is expected behavior for a demo page, but the input type is 'text' (not 'password'), meaning the key is visible in the browser. The page explicitly notes these are public demo keys. Any browser extension, screenshot tool, or shoulder-surfing attack could capture these keys. For the reg.opennhp.org page, private keys would be stored in

localStorage after key generation (as indicated by the 'saved to localStorage + download' label in the UI).

Technical Analysis

DOM inspection confirms that cryptographic private keys are populated into regular text inputs (not password fields) and are visible in the rendered page. For the demo scenario, these keys are documented as intentionally public. However, the pattern of displaying private keys in plaintext input fields and storing them in localStorage establishes an insecure pattern that, if replicated in production applications using the SDK, would expose user-specific private keys to browser extensions, XSS attacks, and physical observation.

Exploitation Commands

```
Playwright: page.evaluate(() => document.getElementById('agentPrivKey').value)
```

Evidence / Logs

```
Input field values populated from config.json:
#agentPrivKey: "1vlJ3OKc83FT2bwMcF3M0BtaNs7xSj4QBW2wdt2mU7E=" (type="text", visible)
#serverPubKey: "eVr33jqYo2nt5llEUeDu6gJNHxpnhfY0TRJ4iVIFniM=" (type="text", visible)
reg.opennhp.org:
Private key display label: "Private Key (saved to localStorage + download)"
Indicates keys will be stored in browser localStorage
```

Remediation

For production applications: use password-type inputs for private keys, avoid storing private keys in localStorage (use sessionStorage or the Web Crypto API's non-extractable keys instead), and clear private key values from DOM elements when not actively needed.

43. [INFO] nhp-token Cookie Has Very Short Max-Age of 15 Seconds (May Cause Usability Issues)

Severity	INFO
CVSS	0.0 -
Host	auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid
Category	misconfiguration

Severity	INFO
Status	CONFIRMED
MITRE ATT&CK	None
Detected by	browser-playwright

Description

The nhp-token cookie set after authentication has a Max-Age of only 15 seconds (matching the opnTime=15 value in the auth response). This means the authentication cookie expires 15 seconds after it is set. While this is extremely short and limits the exposure window for token theft, it is consistent with the NHP (Network-infrastructure Hiding Protocol) design where access to the protected server is time-limited. The 5-second countdown redirect timer on the auth page is designed to redirect the user to ac.opennhp.org before the token expires. This is by design for the NHP protocol's zero-trust approach.

Technical Analysis

The nhp-token cookie has a 15-second TTL, matching the NHP protocol's opnTime field. This creates a very narrow access window where the user must be redirected to the protected resource before the token expires. This is a security-positive design feature of NHP (limiting exposure time), not a vulnerability.

Exploitation Commands

```
curl -s -D - "https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid&user
name=user&password=password" | grep set-cookie
```

Evidence / Logs

```
set-cookie: nhp-token=dGZTkNdmNBtOIGrHfQn4kR0yXGIWgtUeWPh1Pkjawco%253D; Path=/; Domain=ope
nnhp.org; Max-Age=15; HttpOnly; Secure
JSON response opnTime: 15 (matches Max-Age)
```

Remediation

No remediation needed - this is by design for the NHP protocol. Document this behavior clearly for users who may experience expired token errors if redirect is slow.

44. [INFO] Default/Weak Demo Credentials (user/password)

Severity	INFO
CVSS	0.0 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N
Host	server/auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example
Category	authentication_bypass
Status	CONFIRMED
MITRE ATT&CK	T1078.001 Valid Accounts: Default Accounts
Detected by	go-web-agent

Description

The authentication endpoint uses trivially guessable default credentials: username='user', password='password'. These are pre-filled in the login form HTML. While this is a public demo (credentials are intentionally visible in the page source as default form values), the combination of default credentials + no rate limiting + CORS wildcard creates a compounding risk. The credentials are valid across all three auth-serving hosts (auth-plugin.opennhp.org, server.opennhp.org, demologin.opennhp.org).

Technical Analysis

The default credentials user/password are embedded in the login page HTML as default input values and work for authentication. This is by design for the public demo, as confirmed by the _comment field in config.json stating 'PUBLIC DEMO CONFIG'. However, the finding is noted as it compounds with other issues (CORS, no rate limiting) to increase overall risk.

Exploitation Commands

```
curl -sk "https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid&username
=user&password=password"
```

Raw Request

```
GET /plugins/example?resid=demo&action=valid&username=user&password=password HTTP/2
Host: auth-plugin.opennhp.org
```

Raw Response

```
{"errCode":"0","resHost":{"demoServer":"ac.opennhp.org"},"opnTime":15,"agentAddr":"IP_PUBL
IC_003","acTokens":{"demoServer":"..."},"redirectUrl":"https://ac.opennhp.org"}
```

Evidence / Logs

```
Response: {"errCode":"0","resHost":{"demoServer":"ac.opennhp.org"},"opnTime":15,...}
HTML source contains: <input type="password" id="password" name="password" value="password
">
```

Remediation

For production deployments, require strong passwords and disable default credentials. Implement account provisioning workflows that require unique passwords.

45. [INFO] Relay API Verbose Error Messages Enable Endpoint Enumeration

Severity	INFO
CVSS	0.0 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N
Host	agent/reg/relay (3.149.105.15)
Endpoint	https://relay.opennhp.org/relay/
Category	information_disclosure
Status	CONFIRMED
MITRE ATT&CK	T1592 Gather Victim Host Information
Detected by	go-web-agent

Description

The relay.opennhp.org API returns detailed error messages that reveal the API structure and guide attackers to valid endpoints. POST /relay/ returns 'missing server id; use POST /relay/<serverId> (see GET /servers)' which directly instructs the attacker to enumerate servers. POST /relay/invalidId returns 'unknown server ID "invalidId"' confirming server ID validation. POST /relay/validId with short payload returns 'inner packet too short' revealing protocol expectations. These verbose errors provide a guided discovery path through the relay API.

Technical Analysis

Each error response progressively reveals the API structure. The first error points to /servers for enumeration. The second confirms server ID validation (string comparison). The third reveals the payload format expectation (binary NHP packet, not JSON). The fourth confirms the relay actually attempts to forward the data to the backend NHP server. These errors together constitute a complete guided tour of the relay API internals.

Exploitation Commands

```
curl -skD- -X POST "https://relay.opennhp.org/relay/" -H "Content-Type: application/json"
-d '{}'
curl -skD- -X POST "https://relay.opennhp.org/relay/demoServer" -H "Content-Type: applicat
ion/json" -d '{}'
curl -skD- -X POST "https://relay.opennhp.org/relay/7rFdWSvghe8" -H "Content-Type: applica
tion/json" -d '{}'
```

Raw Request

```
POST /relay/ HTTP/2
Host: relay.opennhp.org
Content-Type: application/json

{}
```

Raw Response

```
HTTP/2 400
content-type: text/plain; charset=utf-8

missing server id; use POST /relay/<serverId> (see GET /servers)
```

Evidence / Logs

```
POST /relay/ -> 400: "missing server id; use POST /relay/<serverId> (see GET /servers)"
POST /relay/demoServer -> 404: 'unknown server ID "demoServer"'
POST /relay/7rFdWSvghe8 {} -> 400: "inner packet too short"
POST /relay/7rFdWSvghe8 with data -> 504: "NHP Server timeout"
```

Remediation

Replace verbose error messages with generic ones. Use standard HTTP error codes without revealing API structure. Remove the 'see GET /servers' hint.

46. [INFO] Inconsistent HTTP Method Handling - PUT Returns 204, DELETE Returns 204

Severity	INFO
CVSS	0.0 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N
Host	server/auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example
Category	security_misconfiguration
Status	CONFIRMED
MITRE ATT&CK	T1592 Gather Victim Host Information
Detected by	go-web-agent

Description

The auth-plugin.opennhp.org endpoint exhibits inconsistent HTTP method handling. While Access-Control-Allow-Methods advertises 'GET, OPTIONS, POST', the server also accepts PUT and DELETE with 204 (No Content) responses. PATCH returns 404 and POST returns 404 (on the plugin endpoint). This inconsistency means: (1) The CORS Allow-Methods header doesn't match actual method handling, (2) PUT/DELETE are accepted but produce no content, suggesting they may trigger CORS preflight edge cases, (3) The server's method routing is overly permissive.

Technical Analysis

The server's HTTP method routing is inconsistent with its CORS headers. PUT and DELETE return 204 but are not listed in Access-Control-Allow-Methods. POST returns 404 despite being listed in Allow-Methods. This suggests the CORS middleware applies generic Allow-Methods headers without matching the actual route registrations. While not directly exploitable, this inconsistency could confuse security scanners and may

indicate that the CORS middleware is applied globally rather than per-route.

Exploitation Commands

```
curl -skD- -X PUT "https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid
&username=user&password=password"
curl -skD- -X DELETE "https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=va
lid&username=user&password=password"
curl -skD- -X PATCH "https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=val
id&username=user&password=password"
curl -skD- -X POST "https://auth-plugin.opennhp.org/plugins/example"
```

Raw Request

```
PUT /plugins/example?resid=demo&action=valid&username=user&password=password HTTP/2
Host: auth-plugin.opennhp.org
```

Raw Response

```
HTTP/2 204
access-control-allow-methods: GET, OPTIONS, POST
(Note: PUT not in Allow-Methods but returns 204)
```

Evidence / Logs

```
GET -> 200 (auth response)
OPTIONS -> 200 (CORS preflight, empty body)
POST -> 404 (not found)
PUT -> 204 (No Content) - not in Allow-Methods
DELETE -> 204 (No Content) - not in Allow-Methods
PATCH -> 404 (not found)
Access-Control-Allow-Methods: GET, OPTIONS, POST (does not match actual behavior)
```

Remediation

Ensure CORS Allow-Methods matches actual method handling. Return 405 Method Not Allowed for unsupported methods. Use a strict method check in the handler or route registration.

47. [INFO] Nginx Version Disclosure in Server Header

Severity	INFO

Severity	INFO
CVSS	0.0 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N
Host	server/auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/
Category	information_disclosure
Status	CONFIRMED
MITRE ATT&CK	T1592 Gather Victim Host Information
Detected by	go-web-agent

Description

All hosts expose the nginx version in the Server response header: 'server: nginx/1.30.2'. This reveals the exact web server version to potential attackers, enabling targeted exploitation of known vulnerabilities. Both 3.149.215.44 (auth-plugin, server) and 3.149.105.15 (demologin, agent, reg, relay) expose this header.

Technical Analysis

The nginx server header exposes the exact version (1.30.2) on all six hosts. This is a standard information disclosure that aids attacker reconnaissance by narrowing the search for applicable exploits and CVEs.

Exploitation Commands

```
curl -skD- "https://auth-plugin.opennhp.org/" -o /dev/null | grep server
```

Raw Request

```
GET / HTTP/2
Host: auth-plugin.opennhp.org
```

Raw Response

```
server: nginx/1.30.2
```

Evidence / Logs

```
All responses contain: server: nginx/1.30.2
```

Remediation

Add 'server_tokens off;' to the nginx configuration to suppress version information from the Server header.

48. [INFO] Authentication Failures Return HTTP 200 Instead of 401

Severity	INFO
CVSS	0.0 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N
Host	server/auth-plugin.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example
Category	security_misconfiguration
Status	CONFIRMED
MITRE ATT&CK	T1110 Brute Force
Detected by	go-web-agent

Description

The authentication endpoint returns HTTP 200 for all requests regardless of authentication success or failure. Failed logins return HTTP 200 with a JSON error message body rather than the semantically correct HTTP 401 Unauthorized. This makes it harder for WAFs, monitoring tools, and fail2ban-style systems to detect authentication failures. Similarly, invalid resource IDs and invalid actions also return HTTP 200. The only difference is the response body content type (application/json for success, text/plain for errors).

Technical Analysis

All authentication responses use HTTP 200 regardless of outcome. The success vs failure is only distinguishable by parsing the JSON body. This prevents HTTP-level security controls from detecting authentication failures. WAFs looking for 401 responses won't flag brute-force attempts. Log analysis tools

counting 4xx errors won't detect credential stuffing. The Content-Type difference (application/json for success vs text/plain for errors) is a secondary indicator but unreliable for security monitoring.

Exploitation Commands

```
curl -skD- "https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid&userna
me=wrong&password=wrong"
```

Raw Request

```
GET /plugins/example?resid=demo&action=valid&username=wrong&password=wrong HTTP/2
Host: auth-plugin.opennhp.org
```

Raw Response

```
HTTP/2 200
content-type: text/plain; charset=utf-8

{"errMsg": "auth error: user or password is incorrect"}
```

Evidence / Logs

```
Valid credentials: HTTP 200 + Content-Type: application/json + {"errCode":"0",...}
Invalid credentials: HTTP 200 + Content-Type: text/plain + {"errMsg": "auth error: user or
password is incorrect"}
Invalid resource: HTTP 200 + Content-Type: text/plain + {"errMsg": "resource error: failed
to find resource"}
Invalid action: HTTP 200 + Content-Type: text/plain + {"errMsg": "auth error: action inval
id"}
No handler: HTTP 200 + Content-Type: text/plain + {"errMsg": "no auth handler provided"}
```

Remediation

Return HTTP 401 Unauthorized for failed authentication, HTTP 404 for unknown resources/plugins, and HTTP 400 for invalid actions. Use proper HTTP status codes to enable WAF and monitoring integration.

49. [INFO] Agent Private Keys Exposed in Public Config.json (Intentional Demo)

Severity	INFO

Severity	INFO
CVSS	0.0 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N
Host	agent/reg/relay (3.149.105.15)
Endpoint	https://agent.opennhp.org/config.json
Category	information_disclosure
Status	CONFIRMED
MITRE ATT&CK	T1552.004 Unsecured Credentials: Private Keys
Detected by	go-web-agent

Description

The config.json files served from both agent.opennhp.org and reg.opennhp.org contain agent private keys, public keys, server topology, and relay URLs. The file explicitly states '_comment: PUBLIC DEMO CONFIG - agentPrivateKey is intentionally public; see deploy-demo-v2.yml.' Two sets of cluster credentials are exposed: Cluster 1 (agentPrivateKey: 1vlJ3OKc83FT2bwMcF3M0BtaNs7xSj4QBW2wdt2mU7E=) and Cluster 2 (agentPrivateKey: IFgOgHRnVSUAJgzPaxsdhGLxwK2hsav4cXDgnqFhx6w=). While intentionally public for the demo, this configuration pattern could be accidentally replicated in production deployments.

Technical Analysis

The config.json is accessible without authentication on both static-file hosts and contains agent private keys for the Curve25519 key exchange. The _comment field explicitly marks this as intentional for the demo. The keys enable crafting valid NHP agent packets for the demo infrastructure. This is by design (severity: info) but is noted because production deployments must ensure config files with private keys are not served statically.

Exploitation Commands

```
curl -sk "https://agent.opennhp.org/config.json"
curl -sk "https://reg.opennhp.org/config.json"
```

Raw Request

```
GET /config.json HTTP/2
```

Host: agent.opennhp.org

Raw Response

```
{
"_comment": "PUBLIC DEMO CONFIG - agentPrivateKey is intentionally public...",
"agentPrivateKey": "1vIJ3OKc83FT2bwMcF3M0BtaNs7xSj4QBW2wdt2mU7E=",
"agentPublicKey": "jrzTQZsh1pO7vMB+b8cgvKRI4v0ziQdHdGCpJpntzSs=",
...
}
```

Evidence / Logs

Both URLs return identical JSON with agentPrivateKey fields for 2 clusters
_comment field confirms intentional exposure for demo purposes

Remediation

For production: Never serve configuration files with private keys via static file hosting. Use environment variables or secure vault services for key management. Add .json to nginx location blocks that deny access to sensitive paths.

50. [INFO] Relay API Verbose Error Messages

Severity	INFO
CVSS	0.0 -
Host	relay.opennhp.org
Endpoint	https://relay.opennhp.org/relay
Category	information_disclosure
Status	CONFIRMED
MITRE ATT&CK	None
Detected by	golang

Severity	INFO

Description

The relay endpoint at relay.opennhp.org/relay returns verbose error messages when sent malformed requests. This includes details about expected NHP binary protocol format which aids attackers in understanding the internal protocol structure. However, the relay expects a specific NHP Noise handshake binary format and is not vulnerable to SSRF or HTTP forwarding attacks.

Technical Analysis

Verbose errors expose internal protocol details. Not directly exploitable as the relay expects NHP binary protocol.

Exploitation Commands

```
curl -sk -X POST https://relay.opennhp.org/relay -d '{\"test\":\"data\"}'
```

Remediation

Return generic error messages for malformed requests. Avoid revealing internal protocol structure details.

51. [INFO] Authentication Failures Return HTTP 200 Instead of 401

Severity	INFO
CVSS	0.0 -
Host	server.opennhp.org
Endpoint	https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid&username=wrong&password=wrong
Category	authentication_flaw
Status	CONFIRMED
MITRE ATT&CK	None
Detected by	golang

Severity	INFO

Description

Failed authentication attempts to the auth-plugin endpoint return HTTP 200 OK with an error message in the JSON body ({\"errMsg\": \"auth error: user or password is incorrect\"}) instead of the semantically correct HTTP 401 Unauthorized. While this doesn't directly create a security vulnerability, it violates REST API best practices and may confuse WAF/IDS rules that filter by HTTP status codes.

Technical Analysis

The server returns HTTP 200 for failed logins instead of HTTP 401, making it harder for security monitoring tools to detect brute-force attacks based on status codes.

Exploitation Commands

```
curl -sk -o /dev/null -w '%{http_code}' "https://auth-plugin.opennhp.org/plugins/example?resid=demo&action=valid&username=admin&password=wrong"
```

Raw Response

```
HTTP 200 OK\n{\"errMsg\": \"auth error: user or password is incorrect\"}
```

Remediation

Return HTTP 401 Unauthorized for failed authentication attempts.

4. NHP-PROTECTED HOSTS — ZERO ATTACK SURFACE

server2.opennhp.org, ac2.opennhp.org and ac.opennhp.org were probed with curl (--max-time), naabu (targeted ports), raw bash /dev/tcp, and ICMP. Every probe timed out with no response — no TCP handshake, no open port, no banner. The autonomous attack chain never began: there was nothing to fingerprint, enumerate, or exploit. This is the OpenNHP authentication-before-connection model working as designed.

5. CONCLUSION

Same autonomous attacker, same effort: 51 findings against reachable hosts, 0 against NHP-protected hosts. Discoverability, not patch level, is the deciding variable. OpenNHP relocates the attack surface from the network

to the agent credential; for anyone who does not already hold that credential — i.e. every real attacker — the protected service is simply unreachable.