

Building Blocks for HTTP APIs
Internet-Draft
Intended status: Best Current Practice
Expires: 27 February 2026

R. Salz
M. Bishop
Akamai Technologies
M. Kleidl
Transloadit
26 August 2025

API Keys and Privacy
draft-ietf-httpapi-privacy-03

Abstract

Redirecting HTTP requests to HTTPS, a common pattern for human-facing web resources, can be an anti-pattern for authenticated HTTP API traffic. This document discusses the pitfalls and makes deployment recommendations for authenticated HTTP APIs. It does not specify a protocol.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at <https://ietf-wg-httpapi.github.io/draft-ietf-httpapi-privacy/draft-ietf-httpapi-privacy.html>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-ietf-httpapi-privacy/>.

Discussion of this document takes place on the Building Blocks for HTTP APIs Working Group mailing list (<mailto:httpapi@ietf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/httpapi/>. Subscribe at <https://www.ietf.org/mailman/listinfo/httpapi/>.

Source for this draft and an issue tracker can be found at <https://github.com/ietf-wg-httpapi/httpapi-privacy>.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 27 February 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
1.1. Conventions and Definitions	4
2. Server Recommendations	4
2.1. Pre-Connection Redirects	4
2.2. Connection Blocking	4
2.3. Credential Restriction	5
2.4. Disclosure Response	5
2.4.1. Credential Revocation	6
3. Client Recommendations	6
3.1. Implement Relevant Protocols	6
3.2. Respect Credential Restrictions	6
3.3. Disallow Insecure by Default	6
4. Security Considerations	7
5. IANA Considerations	7
6. References	7
6.1. Normative References	7
6.2. Informative References	8
Acknowledgments	8
Authors' Addresses	8

1. Introduction

It is a common pattern for HTTP servers to prefer serving resources over HTTPS. Because HTTPS uses TLS, clients receive authentication of the server and confidentiality of the resource bodies supplied by the server.

In order to implement this preference, HTTP servers often listen for unencrypted requests and respond with a 3XX status code directing the client to the equivalent resource over an encrypted connection. For unauthenticated web browsing, this is a reasonable user experience bridge. Users often type bare hostnames (not URIs) into a user agent; if the user agent defaults to an unencrypted connection, the server can correct this default and require the use of encryption. This pattern is so well established that many HTTP server and intermediary implementations have a prominently displayed option to enable it automatically.

When client authentication is used, more care must be taken. The client's initial request may include a Bearer token or other credential (such as a Cookie); once the request has been sent on the network, any passive attacker who can see the traffic can acquire this credential and use it.

If the server performs a redirect in this situation, it does not mitigate exposure of the credential. Further, because the request will ultimately succeed if the client follows the redirect, an application developer or user who accidentally configures an unencrypted API endpoint will not necessarily notice the misconfiguration.

This document describes actions API servers and clients should take in order to safeguard credentials. These recommendations are not directed at resources where no authentication is used.

Some have wondered if this document is really necessary. After all, we have been telling people not to send passwords and such in the clear for decades. Regrettably, this lesson seems to be largely forgotten by those developing Web-based APIs. The blog post that motivated this document, [BLOG], did a spot-check in May, 2024, and found over two dozen websites that were vulnerable to the issues listed here.

1.1. Conventions and Definitions

Although this document is not an IETF Standards Track publication, it adopts the conventions for normative language to provide clarity of instructions to the implementer. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Server Recommendations

2.1. Pre-Connection Redirects

Various mechanisms exist to inform clients that unencrypted requests to a server are never appropriate:

- * HTTP Strict Transport Security (HSTS) [RFC6797] informs clients who make a successful connection over HTTPS that secure connections are a requirement in the future.
- * HTTPS DNS records [RFC9460] inform clients at connection time to use only secure connections to the indicated server.

Neither mechanism is foolproof. An attacker with control of the network or the DNS server could block resolution of HTTPS records on a client connecting to a new server, while HSTS requires a successful prior connection to the server and relies on the client to implement persistent storage of the HSTS directive.

Used together, however, both approaches make clients less likely to send any requests over an insecure channel. HTTP API servers with authenticated endpoints SHOULD employ both mechanisms.

2.2. Connection Blocking

If an API request succeeds despite having an unencrypted endpoint configured, the developer or user is less likely to notice the misconfiguration. Where possible, it is advantageous for such a misconfiguration to fail immediately so that the error can be noticed and corrected.

HTTP API servers MAY induce such an early failure by not accepting unencrypted connections, e.g. on port 80. This makes it impossible for a client to send a credential over an insecure channel to the authentic server, as no such channel can be opened. HTTP API servers MAY alternatively restrict connections on port 80 to network sources which are more trusted, such as a VPN or virtual network interface.

However, this mitigation is limited against active network attackers, who can impersonate the HTTP API server and accept the client's insecure connection attempt.

2.3. Credential Restriction

Whenever possible, credentials should include an indicator to clients that the credential is restricted to secure contexts. For example, Cookie-based authentication SHOULD include the Secure attribute described in Section 4.1.2.5 of [RFC6265]. Bearer tokens MAY use the format described in [RFC8959] to indicate the expected usage to the client.

2.4. Disclosure Response

Some deployments may not find it feasible to completely block unencrypted connections, whether because the hostname is shared with unauthenticated endpoints or for infrastructure reasons. Therefore, HTTP API servers need a response for when a credential has been received over an insecure channel.

HTTP status code 403 (Forbidden) indicates that "the server understood the request but refuses to fulfill it" [HTTP]. While this is generally understood to mean that "the server considers [the credentials] insufficient to grant access," it also states that "a request might be forbidden for reasons unrelated to the credentials." HTTP API servers SHOULD return status code 403 to all requests received over an insecure channel, regardless of the validity of the presented credentials.

Because a difference in behavior would enable attackers to guess and check possible credentials, an HTTP API server MUST NOT return a different client response between a valid or invalid credential presented over an insecure connection. Differences in behavior MUST only be visible on subsequent use of the credential over a secure channel.

2.4.1. Credential Revocation

When a request is received over an unencrypted channel, the presented credential is potentially compromised. HTTP API servers SHOULD revoke such credentials immediately. When the credential is next used over a secure channel, the server MAY return an error that indicates why the credential was revoked.

Credentials in a request can take on different forms. API keys and tokens are simple modes for authentication, but can be abused by attackers to forge requests and hence should be revoked if compromised. Requests can also be authenticated using derived values, where they only include digital signatures or message authentication codes (MACs) derived from credentials but not the credentials themselves. Since an attacker cannot abuse the derived values to forge requests, the server MAY choose to not revoke the credentials in this case.

3. Client Recommendations

The following recommendations increase the success rate of the server recommendations above.

3.1. Implement Relevant Protocols

Clients SHOULD support and query for HTTPS records [RFC9460] when establishing a connection. This gives HTTP API servers an opportunity to provide more complete information about capabilities, some of which are security-relevant.

Clients SHOULD respect HSTS headers [RFC6797] received from a server. This includes implementing persistent storage of HSTS indications received from the server.

3.2. Respect Credential Restrictions

Clients MUST NOT send a Cookie with the Secure attribute [RFC6265] over an insecure channel. Clients MUST NOT send an Authorization header containing a token whose value begins with "secret-token:" over an insecure channel.

3.3. Disallow Insecure by Default

When authentication is used, clients SHOULD require an explicit indication from the user or caller that an insecure context is expected which is distinct from the provided URI. Depending on the interface, this might be a UI preference or an API flag.

Absent such an indication, clients of HTTP APIs MUST implement and use HTTPS exclusively.

4. Security Considerations

This entire document is about security of HTTP API interactions.

The behavior recommended in Section 2.4.1 creates the potential for a denial of service attack where an attacker guesses many possible credentials over an unencrypted connection in hopes of discovering and revoking a valid one. HTTP API servers implementing this mitigation MUST also guard against such attacks, such as by limiting the number of requests before closing the connection and rate-limiting the establishment of insecure connections.

5. IANA Considerations

This document has no IANA actions.

6. References

6.1. Normative References

- [HTTP] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <<https://www.rfc-editor.org/rfc/rfc9110>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/rfc/rfc2119>>.
- [RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <<https://www.rfc-editor.org/rfc/rfc6265>>.
- [RFC6797] Hodges, J., Jackson, C., and A. Barth, "HTTP Strict Transport Security (HSTS)", RFC 6797, DOI 10.17487/RFC6797, November 2012, <<https://www.rfc-editor.org/rfc/rfc6797>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/rfc/rfc8174>>.

- [RFC9460] Schwartz, B., Bishop, M., and E. Nygren, "Service Binding and Parameter Specification via the DNS (SVCB and HTTPS Resource Records)", RFC 9460, DOI 10.17487/RFC9460, November 2023, <<https://www.rfc-editor.org/rfc/rfc9460>>.

6.2. Informative References

- [BLOG] Viide, J., "Your API Shouldn't Redirect HTTP to HTTPS", May 2024, <<https://jviide.iki.fi/http-redirects>>.
- [RFC8959] Nottingham, M., "The "secret-token" URI Scheme", RFC 8959, DOI 10.17487/RFC8959, January 2021, <<https://www.rfc-editor.org/rfc/rfc8959>>.

Acknowledgments

We are grateful to Joachim Viide for his [BLOG] blog posting that brought up the issue.

Authors' Addresses

Rich Salz
Akamai Technologies
Email: rsalz@akamai.com

Mike Bishop
Akamai Technologies
Email: mbishop@evequefou.be

Marius Kleidl
Transloadit
Email: marius@transloadit.com