Workgroup: Captive Portal Interaction
Internet-Draft: draft-ietf-capport-api-08
Published: 18 June 2020
Intended Status: Standards Track
Expires: 20 December 2020

Authors: T. Pauly, Ed.   D. Thakore, Ed.
Apple Inc.   CableLabs

Captive Portal API

Abstract

This document describes an HTTP API that allows clients to interact with a Captive Portal system. With this API, clients can discover how to get out of captivity and fetch state about their Captive Portal sessions.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes a HyperText Transfer Protocol (HTTP) Application Program Interface (API) that allows clients to interact with a Captive Portal system. The API defined in this document has been designed to meet the requirements in the Captive Portal Architecture [I-D.ietf-capport-architecture]. Specifically, the API provides:

*The state of captivity (whether or not the client has access to the Internet)

*A URI of a user-facing web portal that can be used to get out of captivity

*Authenticated and encrypted connections, using TLS for connections to both the API and user-facing web portal

2. Terminology

This document leverages the terminology and components described in [I-D.ietf-capport-architecture] and additionally defines the following terms:

*Captive Portal Client: The client that interacts with the Captive Portal API is typically some application running on the User Equipment that is connected to the Captive Network. This is also referred to as the "client" in this document.

- Captive Portal API Server: The server exposing the APIs defined in this document to the client. This is also referred to as the "API server" in this document.

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.

3. Workflow

The Captive Portal Architecture defines several categories of interaction between clients and Captive Portal systems:

1. Provisioning, in which a client discovers that a network has a captive portal, and learns the URI of the API server.

2. API Server interaction, in which a client queries the state of captivity and retrieves the necessary information to get out of captivity.

3. Enforcement, in which the enforcement device in the network blocks disallowed traffic.

This document defines the mechanisms used in the second category. It is assumed that the location of the Captive Portal API server has been discovered by the client as part of Provisioning. A set of mechanisms for discovering the API Server endpoint is defined in [I-D.ietf-capport-rfc7710bis].

4. API Connection Details

The API server endpoint MUST be accessed over HTTP using an https URI [RFC2818], and SHOULD use the default https port. For example, if the Captive Portal API server is hosted at "example.org", the URI of the API could be "https://example.org/captive-portal/api"

The client SHOULD NOT assume that the URI of the API server for a given network will stay the same, and SHOULD rely on the discovery or provisioning process each time it joins the network.

As described in Section 3 of [I-D.ietf-capport-architecture], the identity of the client needs to be visible to the Captive Portal API server in order for the server to correctly reply with the client's portal state. If the identifier used by the Captive Portal system is the client's set of IP addresses, the system needs to ensure that the same IP addresses are visible to both the API server and the enforcement device.

If the API server needs information about the client identity that is not otherwise visible to it, the URI provided to the client during provisioning SHOULD be distinct per client. Thus, depending on how the Captive Portal system is configured, the URI will be unique for each client host and between sessions for the same client host.

For example, a Captive Portal system that uses per-client session URIs could use "https://example.org/captive-portal/api/X54PD39JV" as its API URI.

4.1. Server Authentication

The purpose of accessing the Captive Portal API over an HTTPS connection is twofold: first, the encrypted connection protects the integrity and confidentiality of the API exchange from other parties on the local network; and second, it provides the client of the API an opportunity to authenticate the server that is hosting the API. This authentication allows the client to ensure that the entity providing the Captive Portal API has a valid certificate for the hostname provisioned by the network using the mechanisms defined in [I-D.ietf-capport-rfc7710bis], by validating that a DNS-ID [RFC6125] on the certificate is equal to the provisioned hostname.

Clients performing revocation checking will need some means of accessing revocation information for certificates presented by the API server. Online Certificate Status Protocol [RFC6960] (OCSP) stapling, using the TLS Certificate Status Request extension [RFC6066] SHOULD be used. OCSP stapling allows a client to perform revocation checks without initiating new connections. To allow for other forms of revocation checking, especially for clients that do not support OCSP stapling, a captive network SHOULD permit connections to OCSP responders or Certificate Revocation Lists (CRLs) that are referenced by certificates provided by the API server. For more discussion on certificate revocation checks, see Section 6.5 of BCP 195 [RFC7525]. In addition to connections to OCSP responders and CRLs, a captive network SHOULD also permit connections to Network Time Protocol (NTP) [RFC5905] servers or other time-sync mechanisms to allow clients to accurately validate certificates.

Certificates with missing intermediate certificates that rely on clients validating the certificate chain using the URI specified in the Authority Information Access (AIA) extension [RFC5280] SHOULD NOT be used by the Captive Portal API server. If the certificates do require the use of AIA, the captive network MUST allow client access to the host specified in the URI.

If the client is unable to validate the certificate presented by the API server, it MUST NOT proceed with any of the behavior for API interaction described in this document. The client will proceed to interact with the captive network as if the API capabilities were not present. It may still be possible for the user to access the network if the network redirects a cleartext webpage to a web portal.

5. API State Structure

The Captive Portal API data structures are specified in JavaScript Object Notation (JSON) [RFC8259]. Requests and responses for the Captive Portal API use the "application/captive+json" media type. Clients SHOULD include this media type as an Accept header in their GET requests, and servers MUST mark this media type as their Content-Type header in responses.

The following key MUST be included in the top-level of the JSON structure returned by the API server:

- "captive" (boolean): indicates whether the client is in a state of captivity, i.e it has not satisfied the conditions to access the external network. If the client is captive (i.e. captive=true), it will still be allowed enough access for it to perform server authentication (Section 4.1).

The following keys can be optionally included in the top-level of the JSON structure returned by the API server:

- "user-portal-url" (string): provides the URL of a web portal that MUST be accessed over TLS with which a user can interact.

- "venue-info-url" (string): provides the URL of a webpage or site that SHOULD be accessed over TLS on which the operator of the network has information that it wishes to share with the user (e.g., store info, maps, flight status, or entertainment).

- "can-extend-session" (boolean): indicates that the URL specified as "user-portal-url" allows the user to extend a session once the client is no longer in a state of captivity. This provides a hint that a client system can suggest accessing the portal URL to the user when the session is near its limit in terms of time or bytes.

- "seconds-remaining" (number): an integer that indicates the number of seconds remaining, after which the client will be placed into a captive state. The API server SHOULD include this value if the client is not captive (i.e. captive=false) and the client session is time-limited, and SHOULD omit this value for captive clients (i.e. captive=true) or when the session is not time-limited.

- "bytes-remaining" (number): an integer that indicates the number of bytes remaining, after which the client will be in placed into a captive state. The byte count represents the sum of the total number of IP packet (layer 3) bytes sent and received by the client, including IP headers. Captive portal systems might not count traffic to whitelisted servers, such as the API server, but clients cannot rely on such behavior. The API server SHOULD include this value if the client is not captive (i.e. captive=false) and the client session is byte-limited, and SHOULD omit this value for captive clients (i.e. captive=true) or when the session is not byte-limited.

The valid JSON keys can be extended by adding entries to the Captive Portal API Keys Registry (Section 8). If a client receives a key that it does not recognize, it MUST ignore the key and any associated values. All keys other than the ones defined in this document as "required" will be considered optional.

Captive Portal JSON content can contain per-client data that is not appropriate to store in an intermediary cache. Captive Portal API servers SHOULD set the Cache-Control header field in any responses to "private", or a more restrictive value such as "no-store" [RFC7234].

Client behavior for issuing requests for updated JSON content is implementation-specific, and can be based on user interaction or the indications of seconds and bytes remaining in a given session. If at any point the client does not receive valid JSON content from the API server, either due to an error or due to receiving no response, the client SHOULD continue to apply the most recent valid content it had received; or, if no content had been received previously, proceed to interact with the captive network as if the API capabilities were not present.

6. Example Interaction

A client connected to a captive network upon discovering the URI of the API server will query the API server to retrieve information about its captive state and conditions to escape captivity. In this example, the client discovered the URI "https://example.org/captive-portal/api/X54PD39JV" using one of the mechanisms defined in [I-D.ietf-capport-rfc7710bis].

To request the Captive Portal JSON content, a client sends an HTTP GET request:

```
GET /captive-portal/api/X54PD39JV HTTP/1.1
Host: example.org
Accept: application/captive+json
```

The server then responds with the JSON content for that client:

```
HTTP/1.1 200 OK
Cache-Control: private
Date: Mon, 02 Mar 2020 05:07:35 GMT
Content-Type: application/captive+json

{
   "captive": true,
   "user-portal-url": "https://example.org/portal.html"
}
```

Upon receiving this information the client will use this information to direct the user to the web portal (as specified by the user-portal-url value) to enable access to the external network. Once the user satisfies the requirements for external network access, the client SHOULD query the API server again to verify that it is no longer captive.

When the client requests the Captive Portal JSON content after gaining external network access, the server responds with updated JSON content:

```
HTTP/1.1 200 OK
Cache-Control: private
Date: Mon, 02 Mar 2020 05:08:13 GMT
Content-Type: application/captive+json

{
   "captive": false,
   "user-portal-url": "https://example.org/portal.html",
   "venue-info-url": "https://flight.example.com/entertainment",
   "seconds-remaining": 326,
   "can-extend-session": true
}
```

7. Security Considerations

One of the goals of this protocol is to improve the security of the communication between client hosts and Captive Portal systems. Client traffic is protected from passive listeners on the local network by requiring TLS-encrypted connections between the client and the Captive Portal API server, as described in Section 4. All communication between the clients and the API server MUST be encrypted.

In addition to encrypting communications between clients and Captive Portal systems, this protocol requires a basic level of authentication from the API server, as described in Section 4.1. Specifically, the API server MUST present a valid certificate on which the client can perform revocation checks. This allows the client to ensure that the API server has authority for the hostname that was provisioned by the network using [I-D.ietf-capport-rfc7710bis]. Note that this validation only confirms that the API server matches what the network's provisioning mechanism (such as DHCP or IPv6 Router Advertisements) provided, and not validating the security of those provisioning mechanisms or the user's trust relationship to the network.

7.1. Privacy Considerations

Information passed between a client and the user-facing web portal may include a user's personal information, such as a full name and credit card details. Therefore, it is important that both the user-facing web portal and the API server that points a client to the web portal are only accessed over encrypted connections.

It is important to note that although communication to the user-facing web portal requires using TLS, the authentication only validates that the web portal server matches the name in the URI provided by the API server. Since this is not a name that a user typed in, the hostname of the web site that would be presented to the user may include "confusable characters" that can mislead the user. See Section 12.5 of [RFC8264] for a discussion of confusable characters.

8. IANA Considerations

IANA is requested to create a registration for an "application/captive+json" media type (Section 8.1) and a registry for fields in that format (Section 8.2).

8.1. Captive Portal API JSON Media Type Registration

This document registers the media type for Captive Portal API JSON text, "application/captive+json".

Type name: application

Subtype name: captive+json

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: Encoding considerations are identical to those specified for the "application/json" media type.

Security considerations: See Section 7

Interoperability considerations: This document specifies format of conforming messages and the interpretation thereof.

Published specification: This document

Applications that use this media type: This media type is intended to be used by servers presenting the Captive Portal API, and clients connecting to such captive networks.

Fragment identifier considerations: N/A

Additional information: N/A

Person and email address to contact for further information: See Authors' Addresses section

Intended usage: COMMON

Restrictions on usage: N/A

Author: CAPPORT IETF WG

Change controller: IETF

8.2. Captive Portal API Keys Registry

IANA is asked to create and maintain a new registry called "Captive Portal API Keys", which will reserve JSON keys for use in Captive Portal API data structures. The initial contents of this registry are provided in Section 5.

Each entry in the registry contains the following fields:

Key: The JSON key being registered, in string format.

Type: The type of the JSON value to be stored, as one of the value types defined in [RFC8259].

Description: A brief description explaining the meaning of the value, how it might be used, and/or how it should be interpreted by clients.

Specification: A reference to a specification that defines the key and explains its usage.

New assignments for Captive Portal API Keys Registry will be administered by IANA using the Specification Required policy [RFC8126]. The Designated Expert is expected to validate the existence of documentation describing new keys in a permanent publicly available specification, such as an Internet Draft or RFC. The expert is expected to validate that new keys have a clear meaning and do not create unnecessary confusion or overlap with existing keys. Keys that are specific to non-generic use cases, particularly ones that are not specified as part of an IETF document, are encouraged to use a domain-specific prefix.

9. Acknowledgments

This work in this document was started by Mark Donnelly and Margaret Cullen. Thanks to everyone in the CAPPORT Working Group who has given input.

10. References

10.1. Normative References

[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/info/rfc2119>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <https://www.rfc-editor.org/info/rfc6066>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer

Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <https://www.rfc-editor.org/info/rfc6960>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <https://www.rfc-editor.org/info/rfc7234>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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/info/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

10.2. Informative References

[I-D.ietf-capport-architecture]
Larose, K., Dolson, D., and H. Liu, "CAPPORT Architecture", Work in Progress, Internet-Draft, draft-ietf-capport-architecture-08, 11 May 2020, <http://www.ietf.org/internet-drafts/draft-ietf-capport-architecture-08.txt>.

[I-D.ietf-capport-rfc7710bis]
Kumari, W. and E. Kline, "Captive-Portal Identification in DHCP / RA", Work in Progress, Internet-Draft, draft-ietf-capport-rfc7710bis-07, 23 May 2020, <http://www.ietf.org/internet-drafts/draft-ietf-capport-rfc7710bis-07.txt>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <https://www.rfc-editor.org/info/rfc7525>.

[RFC8264]
Saint-Andre, P. and M. Blanchet, "PRECIS Framework: Preparation, Enforcement, and Comparison of Internationalized Strings in Application Protocols", RFC 8264, DOI 10.17487/RFC8264, October 2017, <https://www.rfc-editor.org/info/rfc8264>.

Authors' Addresses

Tommy Pauly (editor)
Apple Inc.
One Apple Park Way
Cupertino, California 95014,
United States of America

Email: tpauly@apple.com

Darshak Thakore (editor)
CableLabs
858 Coal Creek Circle
Louisville, CO 80027,
United States of America

Email: d.thakore@cablelabs.com