Captive Portal Interaction
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2019

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

July 01, 2018

Captive Portal API
draft-ietf-capport-api-01

Abstract

This document describes an HTTP API that allows hosts to interact with a Captive Portal system.

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 January 2, 2019.

Copyright Notice

Table of Contents

1. Introduction

This document describes a HyperText Transfer Protocol (HTTP) Application Program Interface (API) that allows hosts 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 host has access to the Internet)
- A URI that a host's browser can present to a user to get out of captivity
- An encrypted connection (TLS for both the API and portal URI)

2. Workflow

The Captive Portal Architecture defines three steps of interaction between hosts and a Captive Portal service:

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

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

3. Enforcement, in which the enforcement device in the network blocks disallowed traffic, and sends ICMP messages to let hosts know they are blocked by the captive portal

This document is focused on the second step. It is assumed that the location of the Captive Portal API server has been discovered by the host as part of the first step. The mechanism for discovering the API Server endpoint is not covered by this document.

3. API Details

3.1. URI of Captive Portal API endpoint

The URI of the API endpoint MUST be accessed using HTTP over TLS (HTTPS) and SHOULD be served on port 443 [RFC2818]. The host SHOULD NOT assume that the URI for a given network attachment will stay the same, and SHOULD rely on the discovery or provisioning process each time it joins the network. Depending on how the Captive Portal system is configured, the URI may be unique for each host and between sessions for the same host.

For example, if the Captive Portal API server is hosted at example.org, the URI's of the API could be:

- "https://example.org/captive-portal/api"

- "https://example.org/captive-portal/api/X54PD"

3.1.1. 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 is aimed at allowing a user to be reasonably

confident that the entity providing the Captive Portal API has a valid certificate for the hostname in the URI (such as "example.com"). The hostname of the API SHOULD be displayed to the user in order to indicate the entity which is providing the API service.

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, a captive network could permit connections to OCSP responders or Certificate Revocation Lists (CRLs) that are referenced by certificates provided by the API server.

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 will need to 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 by being redirected to a web portal.

3.2. JSON Keys

The Captive Portal API data structures are specified in JavaScript Object Notation (JSON) [RFC7159]. 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 keys are defined at the top-level of the JSON structure returned by the API server:

- "permitted" (required, boolean): indicates whether or not the Captive Portal is open to the requesting host

- "user-portal-url" (required, string): provides the URL of a web portal that can be presented to a user to interact with

- "expire-date" (optional, string formatted as [RFC3339] datetime): indicates the date and time after which the host will be in a captive state

- "bytes-remaining" (optional, integer): indicates the number of bytes left, after which the host will be in a captive state

3.3. Example Exchange

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


```
GET /captive-portal/api/X54PD
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, 04 Dec 2013 05:07:35 GMT
Content-Type: application/captive+json

{
   "permitted": false,
   "user-portal-url": "https://example.org/portal.html"
   "expire-date": "2014-01-01T23:28:56.782Z"
}
```

4. Security Considerations

TBD: Provide complete security requirements and analysis.

4.1. Privacy Considerations

Information passed in this protocol may include a user's personal information, such as a full name and credit card details. Therefore, it is important that Captive Portal API Servers do not allow access to the Captive Portal API over unencrypted sessions.

5. IANA Considerations

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

Type name: application

Subtype name: captive+json

Required parameters: None

Optional parameters: None

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

Security considerations: See Section 4

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.

Additional information: None

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

Intended usage: COMMON

Restrictions on usage: None

Author: CAPPORT IETF WG

Change controller: IETF

6. 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.

7. References

7.1. Normative References

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

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[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>.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <https://www.rfc-editor.org/info/rfc5785>.

[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>.

[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>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <https://www.rfc-editor.org/info/rfc7159>.

7.2. Informative References

[I-D.ietf-capport-architecture] Larose, K. and D. Dolson, "CAPPORT Architecture", draft-ietf-capport-architecture-02 (work in progress), June 2018.

Authors' Addresses

Tommy Pauly (editor)
Apple Inc.
1 Infinite Loop
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