CAPPORT Architecture

Abstract

   This document aims to document consensus on the CAPPORT
   architecture. DHCP or Router Advertisements, an optional signaling
   protocol, and an HTTP API are used to provide the solution. The role
   of Provisioning Domains (PvDs) is described.

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 18 August 2020.

Copyright Notice

Table of Contents

1. Introduction

In this document, "Captive Portal" is used to describe a network to
which a device may be voluntarily attached, such that network access
is limited until some requirements have been fulfilled. Typically a
user is required to use a web browser to fulfill requirements
imposed by the network operator, such as reading advertisements,
accepting an acceptable-use policy, or providing some form of
credentials.

Implementations generally require a web server, some method to
allow/block traffic, and some method to alert the user. Common
methods of alerting the user involve modifying HTTP or DNS traffic.

Problems with captive portal implementations have been described in
[[I-D.nottingham-capport-problem](#)]. [If that document cannot be

published, consider putting its best parts into an appendix of this document.]

This document standardizes an architecture for implementing captive portals that provides tools for addressing most of those problems. We are guided by these principles:

  *Solutions SHOULD NOT require the forging of responses from DNS or HTTP servers, or any other protocol. In particular, solutions SHOULD NOT require man-in-the-middle proxy of TLS traffic.

  *Solutions MUST operate at the layer of Internet Protocol (IP) or above, not being specific to any particular access technology such as Cable, WiFi or 3GPP.

  *Solutions MAY allow a device to be alerted that it is in a captive network when attempting to use any application on the network.

  *Solutions SHOULD allow a device to learn that it is in a captive network before any application attempts to use the network.

  *The state of captivity SHOULD be explicitly available to devices (in contrast to modification of DNS or HTTP, which is only indirectly machine-detectable by the client--by comparing responses to well-known queries with expected responses).

  *The architecture MUST provide a path of incremental migration, acknowledging a huge variety of portals and end-user device implementations and software versions.

A side-benefit of the architecture described in this document is that devices without user interfaces are able to identify parameters of captivity. However, this document does not yet describe a mechanism for such devices to escape captivity.

The architecture uses the following mechanisms:

  *Network provisioning protocols provide end-user devices with a URI for the API that end-user devices query for information about what is required to escape captivity. DHCP, DHCPv6, and Router-Advertisement options for this purpose are available in [RFC7710bis]. Other protocols (such as RADIUS), Provisioning Domains [I-D.pfister-capport-pvd], or static configuration may also be used. A device MAY query this API at any time to determine whether the network is holding the device in a captive state.

  *End-user devices can be notified of captivity with Captive Portal Signals in response to traffic. This notification should work

with any Internet protocol, not just clear-text HTTP. This
notification does not carry the portal URI; rather it provides a
notification to the User Equipment that it is in a captive state.
This document will specify requirements for a signaling protocol
which could generate Captive Portal Signals.

  *Receipt of a Captive Portal Signal informs an end-user device
   that it could be captive. In response, the device MAY query the
   provisioned API to obtain information about the network state.
   The device MAY take immediate action to satisfy the portal
   (according to its configuration/policy).

The architecture attempts to provide privacy, authentication, and
safety mechanisms to the extent possible.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in RFC 2119 [RFC2119].

1.2. Terminology

Captive Network: A network which limits communication of attached
devices to restricted hosts until the user has satisfied Captive
Portal Conditions, after which access is permitted to a wider set of
hosts (typically the internet).

Captive Portal Conditions: site-specific requirements that a user or
device must satisfy in order to gain access to the wider network.

Captive Portal Enforcement: The network equipment which enforces the
traffic restriction.

Captive Portal User Equipment: Also known as User Equipment. A
device which has voluntarily joined a network for purposes of
communicating beyond the constraints of the captive network.

Captive Portal Server: The web server providing a user interface for
assisting the user in satisfying the conditions to escape captivity.

Captive Portal API: Also known as API. An HTTP API allowing User
Equipment to query its state of captivity within the Captive Portal.

Captive Portal API Server: Also known as API Server. A server
hosting the Captive Portal API.

Captive Portal Signal: A notification from the network used to
inform the User Equipment that the state of its captivity could have
changed.

Captive Portal Signaling Protocol: Also known as Signaling Protocol.
The protocol for communicating Captive Portal Signals.

2. Components

2.1. User Equipment

The User Equipment is the device that a user desires to be attached
to a network with full access to all hosts on the network (e.g., to
have Internet access). The User Equipment communication is typically
restricted by the Captive Portal Enforcement, described in Section
2.4, until site-specific requirements have been met.

At this time we consider only devices with web browsers, with web
applications being the means of satisfying Captive Portal
Conditions.

  *An example interactive User Equipment is a smart phone.

  *SHOULD support provisioning of the URI for the Captive Portal API
   (e.g., by DHCP)

  *SHOULD distinguish Captive Portal API access per network
   interface, in the manner of Provisioning Domain Architecture
   [RFC7556].

  *SHOULD have a mechanism for notifying the user of the Captive
   Portal

  *SHOULD have a web browser so that the user may navigate the
   Captive Portal user interface.

  *MAY restrict application access to networks not granting full
   network access. E.g., a device connected to a mobile network may
   be connecting to a WiFi network; the operating system MAY avoid
   updating the default route until network access restrictions have
   been lifted (excepting access to the Captive Portal server). This
   has been termed "make before break".

None of the above requirements are mandatory because (a) we do not
wish to say users or devices must seek access beyond the captive
network, (b) the requirements may be fulfilled by manually visiting
the captive portal web application, and (c) legacy devices must
continue to be supported.

2.2. Provisioning Service

Here we discuss candidate mechanisms for provisioning the User
Equipment with the URI of the API to query captive portal state and
navigate the portal.

2.2.1. DHCP or Router Advertisements

A standard for providing a portal URI using DHCP or Router
Advertisements is described in [RFC7710bis]. The CAPPORT
architecture expects this URI to indicate the API described in
Section 2.3.

2.2.2. Provisioning Domains

Although still a work in progress, [I-D.pfister-capport-pvd]
proposes a mechanism for User Equipment to be provided with PvD
Bootstrap Information containing the URI for the JSON-based API
described in Section 2.3.

2.3. Captive Portal API Server

The purpose of a Captive Portal API is to permit a query of Captive
Portal state without interrupting the user. This API thereby removes
the need for a device to perform clear-text "canary" HTTP queries to
check for response tampering.

The URI of this API will have been provisioned to the User
Equipment. (Refer to Section 2.2).

This architecture expects the User Equipment to query the API when
the User Equipment attaches to the network and multiple times
thereafter. Therefore the API MUST support multiple repeated queries
from the same User Equipment, returning the current state of
captivity for the equipment.

At minimum the API MUST provide: (1) the state of captivity and (2)
a URI for a browser to present the portal application to the user.
The API SHOULD provide evidence to the caller that it supports the
present architecture.

When user equipment receives Captive Portal Signals, the user
equipment MAY query the API to check the state. The User Equipment
SHOULD rate-limit these API queries in the event of the signal being
flooded. (See Section 7.)

The API MUST be extensible to support future use-cases by allowing
extensible information elements.

The API MUST use TLS for privacy and server authentication. The
implementation of the API MUST ensure both privacy and integrity of
any information provided by or required by it.

This document does not specify the details of the API.

2.4. Captive Portal Enforcement

The Captive Portal Enforcement component restricts network access to
User Equipment according to site-specific policy. Typically User
Equipment is permitted access to a small number of services and is
denied general network access until it has performed some action.

The Captive Portal Enforcement component:

  *Allows traffic through for allowed User Equipment.

  *Blocks (discards) traffic for disallowed User Equipment.

  *May signal User Equipment using the Captive Portal Signaling
   protocol if traffic is blocked.

  *Permits disallowed User Equipment to access necessary APIs and
   web pages to fulfill requirements of exiting captivity.

  *Updates allow/block rules per User Equipment in response to
   operations from the Captive Portal back-end.

2.5. Captive Portal Signal

User Equipment may send traffic outside the captive network prior to
the Enforcement device granting it access. The Enforcement Device
rightly blocks or resets these requests. However, lacking a signal
from the Enforcement Device or interaction with the API server, the
User Equipment can only guess at whether it is captive.
Consequently, allowing the Enforcement Device to signal to the User
Equipment that there is a problem with its connectivity may improve
the user's experience.

An Enforcement Device may also want to inform the User Equipment of
a pending expiry of its access to the external network, so providing
the Enforcement Device the ability to preemptively signal may be
desirable.

A specific Captive Portal Signaling Protocol is out of scope for
this document. However, in order to ensure that future protocols fit
into the architecture, requirements for a Captive Portal Signaling
Protocol follow:

  1. The notification SHOULD NOT be easy to spoof. If an attacker
     can send spoofed notifications to the User Equipment, they can
     cause the User Equipment to unnecessarily access the API.
     Rather than relying solely on rate limits to prevent problems,
     a good protocol will strive to limit the feasibility of such
     attacks.

2. It SHOULD be possible to send the notification before the captive portal closes. This will help ensure seamless connectivity for the user, as the User Equipment will not need to wait for a network failure to refresh its login. On receipt of preemptive notification, the User Equipment can prompt the user to refresh.

3. The signal SHOULD NOT include any information other than an indication that traffic is restricted, which can be used as a prompt to contact the API.

The Captive Portal Signaling Protocol does not provide any means of indicating that the network prevents access to some destinations. The intent is to rely on the Captive Portal API and the web portal to which it points to communicate local network policies.

The Captive Portal Enforcement function MAY send Captive Portal Signals when disallowed User Equipment attempts to send to the network. These signals MUST be rate-limited to a configurable rate.

The signals MUST NOT be sent to the Internet devices. The indications are only sent to the User Equipment.

2.6. Component Diagram

The following diagram shows the communication between each component.

```
O . . . . . . . . . . . . . . . . . . . . . . . . . . . . O
. CAPTIVE NETWORK                                          .
.                                         +--------------+ .
. +-----------+   Provision API URI       | Provisioning | .
. |           |<--------------------------+|  Service    | .
. |   User    |                           +--------------+ .
. | Equipment |   Query captivity status   +------------+  .
. |           |+-------------------------->|   API      |  .
. |           |                           |  Server    |   .
. |           |                           +------+------+  .
. |           |                                  | Status  .
. |           |    Portal user interface  +------+------+  .
. |           |+------------------------->|  Web Portal |   .
. +-----------+                           |  Server     |   .
.     ^   ^ |                             +------------+   .
.     |   | |   Data                             |        .
.     |   | +---------------> +--------------+ Allow/Deny  .
.     |   +-------------------+|             |    Rules    .
.     |                        | Captive Portal|     |     .
.     |   Captive Portal Signal | Enforcement | <---+      .
.     +-------------------------+--------------+            .
.                              ^ |                         .
.                              | |                         .
.                  Data to/from external network           .
.                              | |                         .
O . . . . . . . . . . . . . .| |. . . . . . . . . . . O
                              | v
                      EXTERNAL NETWORK


         Figure 1: Captive Portal Architecture Component Diagram

   In the diagram:

     *During provisioning (e.g., DHCP), the User Equipment acquires the
      URI for the Captive Portal API.

     *The User Equipment queries the API to learn of its state of
      captivity. If captive, the User Equipment presents the portal
      user interface to the user.

     *The User Equipment attempts to communicate to the external
      network through the Captive Portal enforcement device.

     *The Captive Portal Enforcement device either allows the User
      Equipment's packets to the external network, or if a signal has
      been implemented, responds with a Captive Portal Signal.
```
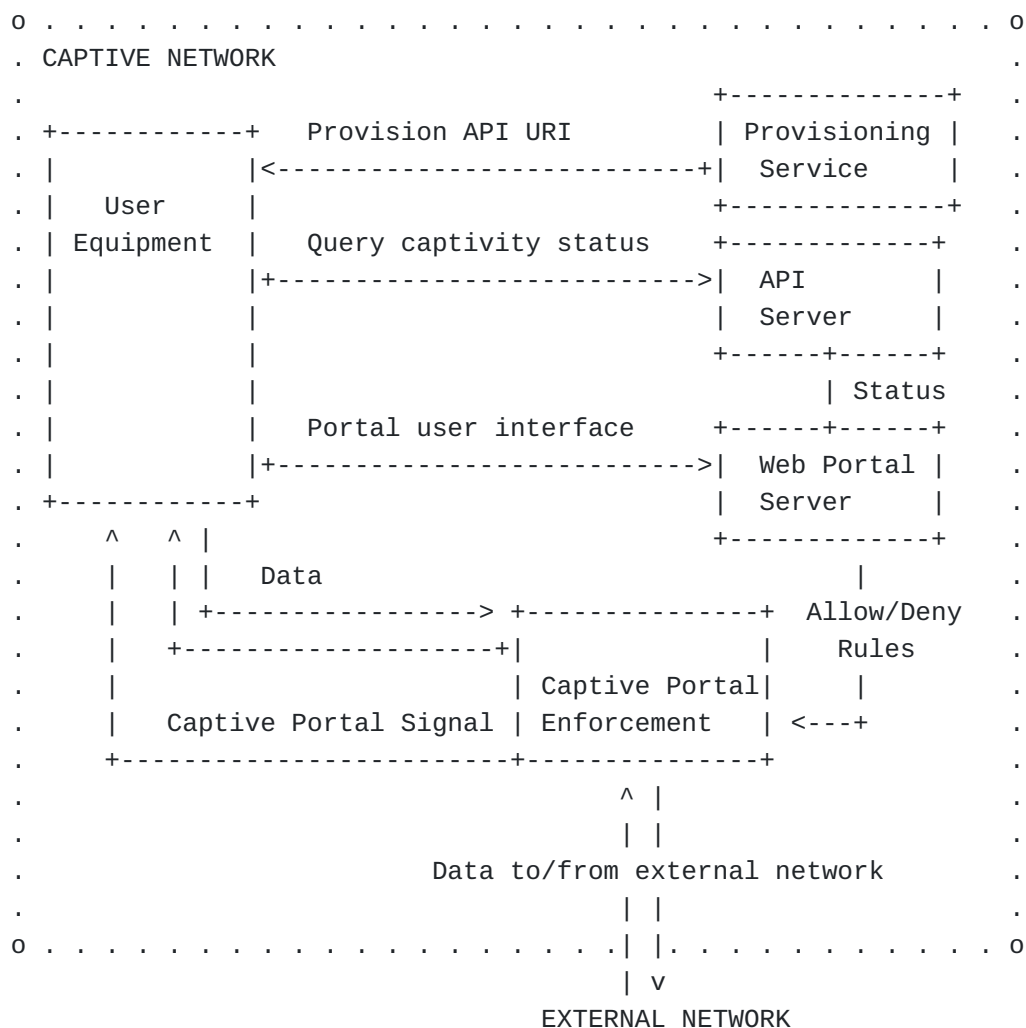
*The Web Portal Server directs the Captive Portal Enforcement
 device to either allow or deny external network access for the
 User Equipment.

Although the Provisioning Service, API Server, and Web Portal Server
functions are shown as discrete blocks, they could of course be
combined into a single element.

3. User Equipment Identity

Multiple components in the architecture interact with both the User
Equipment and each other. Since the User Equipment is the focus of
these interactions, the components must be able to both identify the
user equipment from their interactions with it, and be able to agree
on the identity of the user equipment when interacting with each
other.

The methods by which the components interact restrict the type of
information that may be used as an identifying characteristics. This
section discusses the identifying characteristics.

3.1. Identifiers

An Identifier is a characteristic of the User Equipment used by the
components of a Captive Portal to uniquely determine which specific
User Equipment is interacting with them. An Identifier MAY be a
field contained in packets sent by the User Equipment to the
External Network. Or, an Identifier MAY be an ephemeral property not
contained in packets destined for the External Network, but instead
correlated with such information through knowledge available to the
different components.

3.2. Recommended Properties

The set of possible identifiers is quite large. However, in order to
be considered a good identifier, an identifier SHOULD meet the
following criteria. Note that the optimal identifier will likely
change depending on the position of the components in the network as
well as the information available to them. An identifier SHOULD:

  *Uniquely Identify the User Equipment

  *Be Hard to Spoof

  *Be Visible to the API Server

  *Be Visible to the Enforcement Device

An identifier might only apply to the current point of network attachment. If the device moves to a different network location its identity could change.

3.2.1. Uniquely Identify User Equipment

In order to uniquely identify the User Equipment, at most one user equipment interacting with the other components of the Captive Portal MUST have a given value of the identifier.

Over time, the user equipment identified by the value MAY change. Allowing the identified device to change over time ensures that the space of possible identifying values need not be overly large.

Independent Captive Portals MAY use the same identifying value to identify different User Equipment. Allowing independent captive portals to reuse identifying values allows the identifier to be a property of the local network, expanding the space of possible identifiers.

3.2.2. Hard to Spoof

A good identifier does not lend itself to being easily spoofed. At no time should it be simple or straightforward for one User Equipment to pretend to be another User Equipment, regardless of whether both are active at the same time. This property is particularly important when the user equipment is extended externally to devices such as billing systems, or where the identity of the User Equipment could imply liability.

3.2.3. Visible to the API Server

Since the API Server will need to perform operations which rely on the identity of the user equipment, such as query whether it is captive, the API Server needs to be able to relate requests to the User Equipment making the request.

3.2.4. Visible to the Enforcement Device

The Enforcement Device will decide on a per packet basis whether it should be permitted to communicate with the external network. Since this decision depends on which User Equipment sent the packet, the Enforcement Device requires that it be able to map the packet to its concept of the User Equipment.

3.3. Evaluating an Identifier

To evaluate whether an identifier is appropriate, one should consider every recommended property from the perspective of interactions among the components in the architecture. When

comparing identifiers, choose the one which best satisfies all of
the recommended properties. The architecture does not provide an
exact measure of how well an identifier satisfies a given property;
care should be taken in performing the evaluation.

3.4. Examples of an Identifier

This section provides some examples of identifiers, along with some
evaluation of whether they are good identifiers. The list of
identifiers is not exhaustive. Other identifiers may be used. An
important point to note is that whether the identifiers are good
depends heavily on the capabilities of the components and where in
the network the components exist.

3.4.1. Physical Interface

The physical interface by which the User Equipment is attached to
the network can be used to identify the User Equipment. This
identifier has the property of being extremely difficult to spoof:
the User Equipment is unaware of the property; one User Equipment
cannot manipulate its interactions to appear as though it is
another.

Further, if only a single User Equipment is attached to a given
physical interface, then the identifier will be unique. If multiple
User Equipment is attached to the network on the same physical
interface, then this property is not appropriate.

Another consideration related to uniqueness of the User Equipment is
that if the attached User Equipment changes, both the API Server and
the Enforcement Device must invalidate their state related to the
User Equipment.

The Enforcement Device needs to be aware of the physical interface,
which constrains the environment: it must either be part of the
device providing physical access (e.g., implemented in firmware), or
packets traversing the network must be extended to include
information about the source physical interface (e.g. a tunnel).

The API Server faces a similar problem, implying that it should co-
exist with the Enforcement Device, or that the enforcement device
should extend requests to it with the identifying information.

3.4.2. IP Address

A natural identifier to consider is the IP address of the User
Equipment. At any given time, no device on the network can have the
same IP address without causing the network to malfunction, so it is
appropriate from the perspective of uniqueness.

However, it may be possible to spoof the IP address, particularly
for malicious reasons where proper functioning of the network is not
necessary for the malicious actor. Consequently, any solution using
the IP address should proactively try to prevent spoofing of the IP
address. Similarly, if the mapping of IP address to User Equipment
is changed, the components of the architecture must remove or update
their mapping to prevent spoofing. Demonstrations of return
routeability, such as that required for TCP connection
establishment, might be sufficient defense against spoofing, though
this might not be sufficient in networks that use broadcast media
(such as some wireless networks).

Since the IP address may traverse multiple segments of the network,
more flexibility is afforded to the Enforcement Device and the API
Server: they simply must exist on a segment of the network where the
IP address is still unique. However, consider that a NAT may be
deployed between the User Equipment and the Enforcement Device. In
such cases, it is possible for the components to still uniquely
identify the device if they are aware of the port mapping.

In some situations, the User Equipment may have multiple IP
addresses, while still satisfying all of the recommended properties.
This raises some challenges to the components of the network. For
example, if the user equipment tries to access the network with
multiple IP addresses, should the enforcement device and API Server
treat each IP address as a unique User Equipment, or should it tie
the multiple addresses together into one view of the subscriber? An
implementation MAY do either. Attention should be paid to IPv6 and
the fact that it is expected for a device to have multiple IPv6
addresses on a single link. In such cases, identification could be
performed by subnet, such as the /64 to which the IP belongs.

3.4.3. Context-free URL

The URLs provided in the API SHOULD contain all the information
necessary to render the resources requested: the resources should
not depend on ambient information, such as remote address on the
connection. This is to ensure that the content served from these
URLs is correct and meaningful to the User Equipment, even when
accessed from a network other than the one that contains the captive
portal. One consequence of this is that URLs provided in the API are
expected to be resolved using public DNS.

Though a URL might still correctly resolve when the UE makes the
request from a different network, it is possible that some functions
could be limited to when the UE makes requests using the captive
network. For example, payment options could be absent or a warning
could be displayed to indicate the payment is not for the current
connection.

URLs could include some means of identifying the User Equipment in the URLs. However, including unauthenticated User Equipment identifiers in the URL may expose the service to spoofing or replay attacks.

4. Solution Workflow

This section aims to improve understanding by describing a possible workflow of solutions adhering to the architecture.

4.1. Initial Connection

This section describes a possible work-flow when User Equipment initially joins a Captive Network.

1. The User Equipment joins the Captive Network by acquiring a DHCP lease, RA, or similar, acquiring provisioning information.

2. The User Equipment learns the URI for the Captive Portal API from the provisioning information (e.g., [RFC7710bis]).

3. The User Equipment accesses the Captive Portal API to receive parameters of the Captive Network, including web-portal URI. (This step replaces the clear-text query to a canary URL.)

4. If necessary, the User navigates the web portal to gain access to the external network.

5. The Captive Portal API server indicates to the Captive Portal Enforcement device that the User Equipment is allowed to access the external network.

6. The User Equipment attempts a connection outside the captive network

7. If the requirements have been satisfied, the access is permitted; otherwise the "Expired" behavior occurs.

8. The User Equipment accesses the network until conditions Expire.

4.2. Conditions About to Expire

This section describes a possible work-flow when access is about to expire.

1. Precondition: the API has provided the User Equipment with a duration over which its access is valid

2. The User Equipment is communicating with the outside network

3. The User Equipment's UI indicates that the length of time left
      for its access has fallen below a threshold

   4. The User Equipment visits the API again to validate the expiry
      time

   5. If expiry is still imminent, the User Equipment prompts the
      user to access the web-portal URI again

   6. The User extends their access through the web-portal

   7. The User Equipment's access to the outside network continues
      uninterrupted

4.3. Handling of Changes in Portal URI

A different Captive Portal API URI could be returned in the
following cases:

   *If DHCP is used, a lease renewal/rebind may return a different
    Captive Portal API URI.

   *If RA is used, a new Captive Portal API URI may be specified in a
    new RA message received by end user equipment.

Whenever a new Portal URI is received by end user equipment, it
SHOULD discard the old URI and use the new one for future requests
to the API.

5. Acknowledgments

The authors thank Lorenzo Colitti for providing the majority of the
content for the Captive Portal Signal requirements.

The authors thank various individuals for their feedback on the
mailing list and during the IETF98 hackathon: David Bird, Erik
Kline, Alexis La Goulette, Alex Roscoe, Darshak Thakore, and Vincent
van Dam.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

7.1. Trusting the Network

When joining a network, some trust is placed in the network
operator. This is usually considered to be a decision by a user on
the basis of the reputation of an organization. However, once a user

makes such a decision, protocols can support authenticating a
network is operated by who claims to be operating it. The
Provisioning Domain Architecture [RFC7556] provides some discussion
on authenticating an operator.

Given that a user chooses to visit a Captive Portal URI, the URI
location SHOULD be securely provided to the user's device. E.g., the
DHCPv6 AUTH option can sign this information.

If a user decides to incorrectly trust an attacking network, they
might be convinced to visit an attacking web page and unwittingly
provide credentials to an attacker. Browsers can authenticate
servers but cannot detect cleverly misspelled domains, for example.

7.2. Authenticated APIs

The solution described here assumes that when the User Equipment
needs to trust the API server, server authentication will be
performed using TLS mechanisms.

7.3. Secure APIs

The solution described here requires that the API be secured using
TLS. This is required to allow the user equipment and API Server to
exchange secrets which can be used to validate future interactions.
The API must ensure the integrity of this information, as well as
its confidentiality.

7.4. Risks Associated with the Signaling Protocol

If a Signaling Protocol is implemented, it may be possible for any
user on the Internet to send signals in attempt to cause the
receiving equipment to communicate with the Captive Portal API. This
has been considered, and implementations may address it in the
following ways:

  *The signal only informs the User Equipment to query the API. It
   does not carry any information which may mislead or misdirect the
   User Equipment.

  *Even when responding to the signal, the User Equipment securely
   authenticates with API Servers.

  *Accesses to the API Server are rate limited, limiting the impact
   of a repeated attack.

7.5. User Options

The Signal could inform the User Equipment that it is being held
captive. There is no requirement that the User Equipment do

something about this. Devices MAY permit users to disable automatic
reaction to captive-portal indications for privacy reasons. However,
there is the trade-off that the user doesn't get notified when
network access is restricted. Hence, end-user devices MAY allow
users to manually control captive portal interactions, possibly on
the granularity of Provisioning Domains.

8. References

8.1. 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>.

[RFC7556]   Anipko, D., Ed., "Multiple Provisioning Domain
            Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015,
            <https://www.rfc-editor.org/info/rfc7556>.

[RFC7710bis] Kumari, W. and E. Kline, "Captive-Portal Identification
            in DHCP / RA", Work in Progress, Internet-Draft, draft-
            ietf-capport-rfc7710bis-01, 12 January 2020, <https://
            tools.ietf.org/html/draft-ietf-capport-rfc7710bis-01>.

8.2. Informative References

[I-D.nottingham-capport-problem]
            Nottingham, M., "Captive Portals Problem Statement", Work
            in Progress, Internet-Draft, draft-nottingham-capport-
            problem-01, 4 April 2016, <http://www.ietf.org/internet-
            drafts/draft-nottingham-capport-problem-01.txt>.

[I-D.pfister-capport-pvd]
            Pfister, P. and T. Pauly, "Using Provisioning Domains for
            Captive Portal Discovery", Work in Progress, Internet-
            Draft, draft-pfister-capport-pvd-00, 30 June 2018,
            <http://www.ietf.org/internet-drafts/draft-pfister-
            capport-pvd-00.txt>.

Appendix A. Existing captive portal detection implementations

Operating systems and user applications may perform various tests
when network connectivity is established to determine if the device
is attached to a network with a captive portal present. A common
method is to attempt to make a HTTP request to a known, vendor
hosted endpoint with a fixed response. Any other response is
interpreted as a signal that a captive portal is present. This check
is typically not secured with TLS, as a network with a captive
portal may intercept the connection, leading to a host name

mismatch. Another test that can be performed is a DNS lookup to a known address with an expected answer. Such tests may be less reliable as the captive portal may only be intercepting TCP traffic and deliberately excluding the interception of DNS queries. DNS queries not using UDP may potentially fail this test if operating over TCP or DNS over HTTP. Malicious or misconfigured networks with a captive portal present may not intercept these requests and choose to pass them through or decide to impersonate, leading to the device having a false negative.

Authors' Addresses

Kyle Larose
Agilicus

Email: kyle@agilicus.com

David Dolson

Email: ddolson@acm.org

Heng Liu

Email: liucougar@google.com