Stream:	Internet Engineering Task Force (IETF)
RFC:	8952
Category:	Informational
Published:	November 2020
ISSN:	2070-1721
Authors:	K. Larose	D. Dolson	H. Liu
---	---	---	---
	Agilicus		*Google*

RFC 8952
Captive Portal Architecture

Abstract

This document describes a captive portal architecture. Network provisioning protocols such as DHCP or Router Advertisements (RAs), an optional signaling protocol, and an HTTP API are used to provide the solution.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8952.

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 of captive portals generally require a web server, some method to allow/block traffic, and some method to alert the user. Common methods of alerting the user in implementations prior to this work involve modifying HTTP or DNS traffic.

This document describes an architecture for implementing captive portals while addressing most of the problems arising for current captive portal mechanisms. The architecture is guided by these requirements:

- Current captive portal solutions typically implement some variations of forging DNS or HTTP responses. Some attempt man-in-the-middle (MITM) proxy of HTTPS in order to forge responses. Captive portal solutions should not have to break any protocols or otherwise act

in the manner of an attacker. Therefore, solutions **MUST NOT** require the forging of responses from DNS or HTTP servers or from any other protocol.

- Solutions **MUST** permit clients to perform DNSSEC validation, which rules out solutions that forge DNS responses. Solutions **SHOULD** permit clients to detect and avoid TLS man-in-the-middle attacks without requiring a human to perform any kind of "exception" processing.
- To maximize universality and adoption, solutions **MUST** operate at the layer of Internet Protocol (IP) or above, not being specific to any particular access technology such as cable, Wi-Fi, or mobile telecom.
- Solutions **SHOULD** allow a device to query the network to determine whether the device is captive, without the solution being coupled to forging intercepted protocols or requiring the device to make sacrificial queries to "canary" URIs to check for response tampering (see Appendix A). Current captive portal solutions that work by affecting DNS or HTTP generally only function as intended with browsers, breaking other applications using those protocols; applications using other protocols are not alerted that the network is a captive portal.
- The state of captivity **SHOULD** be explicitly available to devices via a standard protocol, rather than having to infer the state indirectly.
- The architecture **MUST** provide a path of incremental migration, acknowledging the existence of a huge variety of pre-existing portals and end-user device implementations and software versions. This requirement is not to recommend or standardize existing approaches, but rather to provide device and portal implementors a path to a new standard.

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 describe a mechanism for such devices to negotiate for unrestricted network access. A future document could provide a solution to devices without user interfaces. This document focuses on devices with user interfaces.

The architecture uses the following mechanisms:

- Network provisioning protocols provide end-user devices with a Uniform Resource Identifier (URI) [RFC3986] 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 [RFC8910]. Other protocols (such as RADIUS), Provisioning Domains [CAPPORT-PVD], or static configuration may also be used to convey this Captive Portal API URI. A device **MAY** query this API at any time to determine whether the network is holding the device in a captive state.
- A Captive Portal can signal User Equipment in response to transmissions by the User Equipment. This signal works in response to any Internet protocol and is not done by modifying protocols in band. This signal does not carry the Captive Portal API URI; rather, it provides a signal to the User Equipment that it is in a captive state.
- Receipt of a Captive Portal Signal provides a hint that User Equipment could be captive. In response, the device **MAY** query the provisioned API to obtain information about the network state. The device can take immediate action to satisfy the portal (according to its configuration/policy).

The architecture attempts to provide confidentiality, 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**", "**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.

1.2. Terminology

Captive Portal
: A network that limits the 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 Device
: The network equipment that enforces the traffic restriction. Also known as "Enforcement Device".

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

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

Captive Portal API
: An HTTP API allowing User Equipment to query information about its state of captivity within the Captive Portal. This information might include how to obtain full network access (e.g., by visiting a URI). Also known as "API".

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

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

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

Captive Portal Session
: Also referred to simply as the "Session", a Captive Portal Session is the association for a particular User Equipment instance that starts when it interacts with the Captive Portal and gains open access to the network and ends when the User Equipment moves back into the original captive state. The Captive Network maintains the state of each active Session and can limit Sessions based on a length of time or a number of bytes used. The Session is associated with a particular User Equipment instance using the User Equipment's identifier (see Section 3).

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 Enforcement Device, described in Section 2.4, until site-specific requirements have been met.

This document only considers devices with web browsers, with web applications being the means of satisfying Captive Portal Conditions. An example of such User Equipment is a smart phone.

The User Equipment:

- **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 non-spoofable mechanism for notifying the user of the Captive Portal.
- **SHOULD** have a web browser so that the user may navigate to the User Portal.
- **SHOULD** support updates to the Captive Portal API URI from the Provisioning Service.
- **MAY** prevent applications from using networks that do not grant full network access. For example, a device connected to a mobile network may be connecting to a captive Wi-Fi network; the operating system could avoid updating the default route to a device on the captive Wi-Fi network until network access restrictions have been lifted (excepting access to the User Portal) in the new network. 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 full access to the Captive Portal, (b) the requirements may be fulfilled by manually visiting the captive portal web application, and (c) legacy devices must continue to be supported.

If User Equipment supports the Captive Portal API, it **MUST** validate the API Server's TLS certificate (see [RFC2818]) according to the procedures in [RFC6125]. The API Server's URI is obtained via a network provisioning protocol, which will typically provide a hostname to be used in TLS server certificate validation, against a DNS-ID in the server certificate. If the API Server is identified by IP address, the iPAddress subjectAltName is used to validate the server certificate.

An Enforcement Device **SHOULD** allow access to any services that User Equipment could need to contact to perform certificate validation, such as Online Certificate Status Protocol (OCSP) responders, Certificate Revocation Lists (CRLs), and NTP servers; see Section 4.1 of [RFC8908] for more information. If certificate validation fails, User Equipment **MUST NOT** make any calls to the API Server.

The User Equipment can store the last response it received from the Captive Portal API as a cached view of its state within the Captive Portal. This state can be used to determine whether its Captive Portal Session is near expiry. For example, the User Equipment might compare a timestamp indicating when the Session expires to the current time. Storing state in this way can reduce the need for communication with the Captive Portal API. However, it could lead to the state becoming stale if the User Equipment's view of the relevant conditions (byte quota, for example) is not consistent with the Captive Portal API's.

2.2. Provisioning Service

The Provisioning Service is primarily responsible for providing a Captive Portal API URI to the User Equipment when it connects to the network, and later if the URI changes. The Provisioning Service could also be the same service that is responsible for provisioning the User Equipment for access to the Captive Portal (e.g., by providing it with an IP address). This section discusses two mechanisms that may be used to provide the Captive Portal API URI to the User Equipment.

2.2.1. DHCP or Router Advertisements

A standard for providing a Captive Portal API URI using DHCP or Router Advertisements is described in [RFC8910]. The captive portal architecture expects this URI to indicate the API described in Section 2.3.

2.2.2. Provisioning Domains

[CAPPORT-PVD] proposes a mechanism for User Equipment to be provided with Provisioning Domain (PvD) Bootstrap Information containing the URI for the 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 User Equipment to perform clear-text "canary" (see Appendix A) 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 and return the state of captivity for the equipment.

At minimum, the API **MUST** provide the state of captivity. Further, the API **MUST** be able to provide a URI for the User Portal. The scheme for the URI **MUST** be "https" so that the User Equipment communicates with the User Portal over TLS.

If the API receives a request for state that does not correspond to the requesting User Equipment, the API **SHOULD** deny access. Given that the API might use the User Equipment's identifier for authentication, this requirement motivates Section 3.2.2.

A caller to the API needs to be presented with evidence that the content it is receiving is for a version of the API that it supports. For an HTTP-based interaction, such as in [RFC8908], this might be achieved by using a content type that is unique to the protocol.

When User Equipment receives Captive Portal Signals, the User Equipment **MAY** query the API to check its state of captivity. The User Equipment **SHOULD** rate-limit these API queries in the event of the signal being flooded. (See Section 6.)

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

The API **MUST** use TLS to ensure server authentication. The implementation of the API **MUST** ensure both confidentiality 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 Device

The Enforcement Device component restricts the network access of User Equipment according to the site-specific policy. Typically, User Equipment is permitted access to a small number of services (according to the policies of the network provider) and is denied general network access until it satisfies the Captive Portal Conditions.

The Enforcement Device component:

- Allows traffic to pass for User Equipment that is permitted to use the network and has satisfied the Captive Portal Conditions.
- Blocks (discards) traffic according to the site-specific policy for User Equipment that has not yet satisfied the Captive Portal Conditions.
- Optionally signals User Equipment using the Captive Portal Signaling Protocol if certain traffic is blocked.
- Permits User Equipment that has not satisfied the Captive Portal Conditions to access necessary APIs and web pages to fulfill requirements for escaping captivity.
- Updates allow/block rules per User Equipment in response to operations from the User Portal.

2.5. Captive Portal Signal

When User Equipment first connects to a network, or when there are changes in status, the Enforcement Device could generate a signal toward the User Equipment. This signal indicates that the User Equipment might need to contact the API Server to receive updated information. For instance, this signal might be generated when the end of a Session is imminent or when

network access was denied. For simplicity, and to reduce the attack surface, all signals **SHOULD** be considered equivalent by the User Equipment as a hint to contact the API. If future solutions have multiple signal types, each type **SHOULD** be rate-limited independently.

An Enforcement Device **MUST** rate-limit any signal generated in response to these conditions. See Section 6.4 for a discussion of risks related to a Captive Portal Signal.

2.6. Component Diagram

The following diagram shows the communication between each component in the case where the Captive Portal has a User Portal and the User Equipment chooses to visit the User Portal in response to discovering and interacting with the API Server.

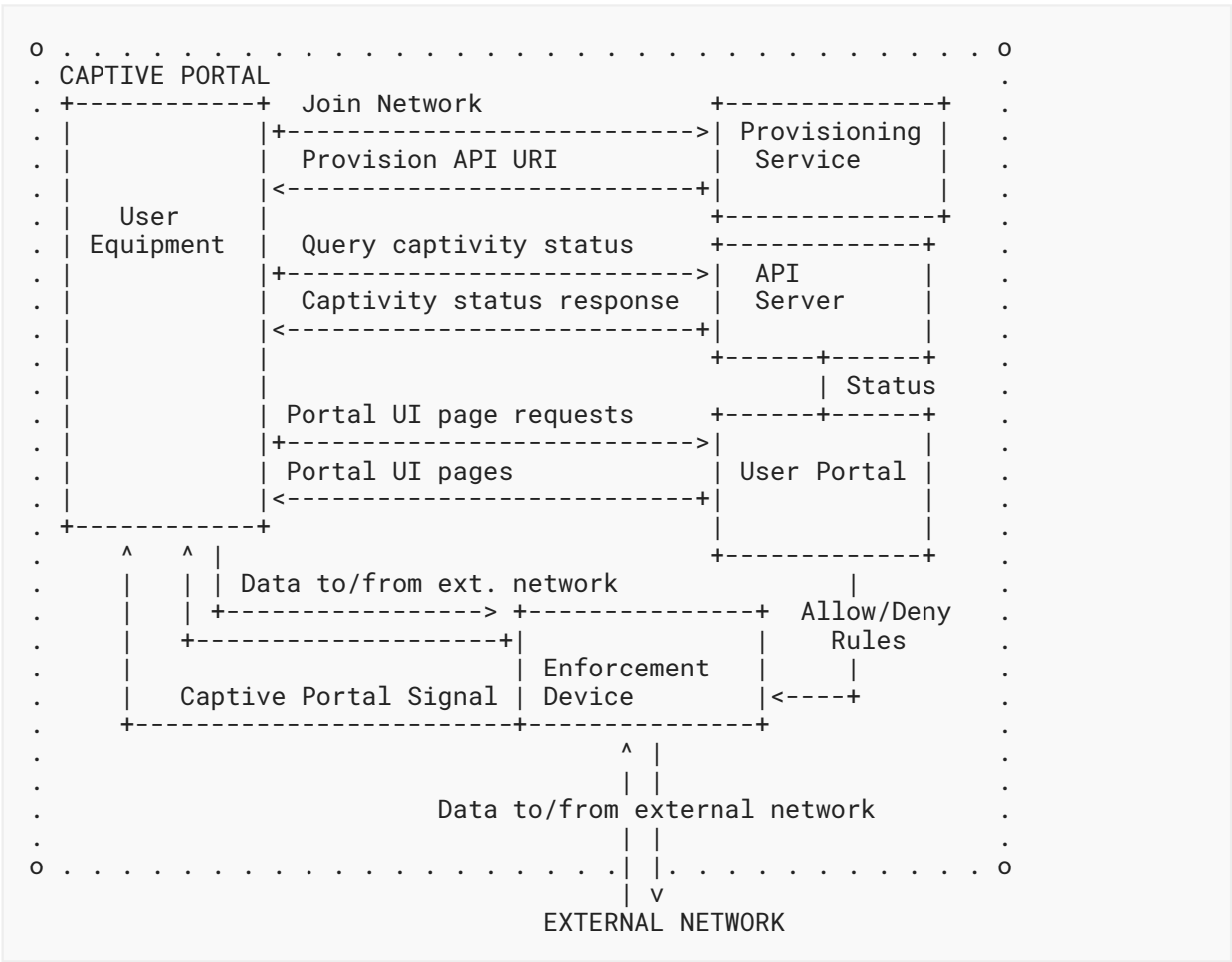

Figure 1: Captive Portal Architecture Component Diagram

In the diagram:

- During provisioning (e.g., DHCP), and possibly later, the User Equipment acquires the Captive Portal API URI.

- The User Equipment queries the API to learn of its state of captivity. If captive, the User Equipment presents the portal user interface from the User Portal to the user.
- Based on user interaction, the User Portal directs the Enforcement Device to either allow or deny external network access for the User Equipment.
- The User Equipment attempts to communicate to the external network through the Enforcement Device.
- The Enforcement Device either allows the User Equipment's packets to the external network or blocks the packets. If blocking traffic and a signal has been implemented, it may respond with a Captive Portal Signal.

The Provisioning Service, API Server, and User Portal are described as discrete functions. An implementation might provide the multiple functions within a single entity. Furthermore, these functions, combined or not, as well as the Enforcement Device, could be replicated for redundancy or scale.

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 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 characteristic. 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 instance is interacting with them. An identifier can be a field contained in packets sent by the User Equipment to the external network. Or, an identifier can 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

The Captive Portal **MUST** associate the User Equipment with an identifier that is unique among all of the User Equipment interacting with the Captive Portal at that time.

Over time, the User Equipment assigned to an identifier 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 instances. 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 instance to pretend to be another User Equipment instance, regardless of whether both are active at the same time. This property is particularly important when the User Equipment identifier is referenced externally by devices such as billing systems or when 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 that rely on the identity of the User Equipment, such as answering a query about whether the User Equipment is captive, the API Server needs to be able to relate a request 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 the packet should be forwarded to the external network. Since this decision depends on which User Equipment instance 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 Types of Identifiers

To evaluate whether a type of identifier is appropriate, one should consider every recommended property from the perspective of interactions among the components in the architecture. When comparing identifier types, choose the one that best satisfies all of the recommended properties. The architecture does not provide an exact measure of how well an identifier type satisfies a given property; care should be taken in performing the evaluation.

3.4. Example Identifier Types

This section provides some example identifier types, along with some evaluation of whether they are suitable types. The list of identifier types is not exhaustive; other types may be used. An important point to note is that whether a given identifier type is suitable 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 type has the property of being extremely difficult to spoof: the User Equipment is unaware of the property; one User Equipment instance cannot manipulate its interactions to appear as though it is another.

Further, if only a single User Equipment instance is attached to a given physical interface, then the identifier will be unique. If multiple User Equipment instances are attached to the network on the same physical interface, then this type 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 type 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 routability, 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 (either IPv4, IPv6, or a dual-stack [RFC4213] combination) 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 instance, 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. Media Access Control (MAC) Address

The MAC address of a device is often used as an identifier in existing implementations. This document does not discuss the use of MAC addresses within a captive portal system, but they can be used as an identifier type, subject to the criteria in Section 3.2.

3.5. Context-Free URI

A Captive Portal API needs to present information to clients that is unique to that client. To do this, some systems use information from the context of a request, such as the source address, to identify the User Equipment.

Using information from context rather than information from the URI allows the same URI to be used for different clients. However, it also means that the resource is unable to provide relevant information if the User Equipment makes a request using a different network path. This might happen when User Equipment has multiple network interfaces. It might also happen if the address of the API provided by DNS depends on where the query originates (as in split DNS [RFC8499]).

Accessing the API **MAY** depend on contextual information. However, the URIs provided in the API **SHOULD** be unique to the User Equipment and not dependent on contextual information to function correctly.

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

URIs could include some means of identifying the User Equipment in the URIs. However, including unauthenticated User Equipment identifiers in the URI 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. Note that the section is not normative; it describes only a subset of possible implementations.

4.1. Initial Connection

This section describes a possible workflow when User Equipment initially joins a Captive Portal.

1. The User Equipment joins the Captive Portal 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., [RFC8910]).
3. The User Equipment accesses the Captive Portal API to receive parameters of the Captive Portal, including the User Portal URI. (This step replaces the clear-text query to a canary URI.)
4. If necessary, the user navigates to the User Portal to gain access to the external network.
5. If the user interacted with the User Portal to gain access to the external network in the previous step, the User Portal indicates to the Enforcement Device that the User Equipment is allowed to access the external network.
6. The User Equipment attempts a connection outside the Captive Portal.
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 workflow 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 detects that the length of time left for its access has fallen below a threshold by comparing its stored expiry time with the current time.
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 User Portal URI again.
6. The user accepts the prompt displayed by the User Equipment.
7. The user extends their access through the User Portal via the User Equipment's user interface.
8. 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.

When the Provisioning Service updates the Captive Portal API URI, the User Equipment can retrieve updated state from the URI immediately, or it can wait as it normally would until the expiry conditions it retrieved from the old URI are about to expire.

5. IANA Considerations

This document has no IANA actions.

6. Security Considerations

6.1. 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 that a network is operated by who claims to be operating it. The Provisioning Domain Architecture [RFC7556] provides some discussion on authenticating an operator.

The user makes an informed choice to visit and trust the Captive Portal URI. Since the network provides the Captive Portal URI to the User Equipment, the network **SHOULD** do so securely so that the user's trust in the network can extend to their trust of the Captive Portal URI. For example, 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.

Further, the possibility of an on-path attacker in an attacking network introduces some risks. The attacker could redirect traffic to arbitrary destinations. The attacker could analyze the user's traffic leading to loss of confidentiality, or the attacker could modify the traffic inline.

6.2. Authenticated APIs

The solution described here requires that when the User Equipment needs to access the API Server, the User Equipment authenticates the server; see Section 2.1.

The Captive Portal API URI might change during the Captive Portal Session. The User Equipment can apply the same trust mechanisms to the new URI as it did to the URI it received initially from the Provisioning Service.

6.3. 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 that can be used to validate future interactions. The API **MUST** ensure the integrity of this information, as well as its confidentiality.

An attacker with access to this information might be able to masquerade as a specific User Equipment instance when interacting with the API, which could then allow them to masquerade as that User Equipment instance when interacting with the User Portal. This could give them the ability to determine whether the User Equipment has accessed the portal, deny the User Equipment service by ending their Session using mechanisms provided by the User Portal, or consume that User Equipment's quota. An attacker with the ability to modify the information could deny service to the User Equipment or cause them to appear as different User Equipment instances.

6.4. 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 an 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 signals to the User Equipment to query the API. It does not carry any information that may mislead or misdirect the User Equipment.
- Even when responding to the signal, the User Equipment securely authenticates with API Servers.
- The User Equipment limits the rate at which it accesses the API, reducing the impact of an attack attempting to generate excessive load on either the User Equipment or API. Note that because there is only one type of signal and one type of API request in response to the signal, this rate-limiting will not cause loss of signaling information.

6.5. User Options

The Captive Portal Signal could signal to 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 Signal indications for privacy reasons. However, there would be 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.

6.6. Privacy

Section 3 describes a mechanism by which all components within the Captive Portal are designed to use the same identifier to uniquely identify the User Equipment. This identifier could be abused to track the user. Implementers and designers of Captive Portals should take care to ensure that identifiers, if stored, are stored securely. Likewise, if any component communicates the identifier over the network, it should ensure the confidentiality of the identifier on the wire by using encryption such as TLS.

There are benefits to choosing mutable anonymous identifiers. For example, User Equipment could cycle through multiple identifiers to help prevent long-term tracking. However, if the components of the network use an internal mapping to map the identity to a stable, long-term value in order to deal with changing identifiers, they need to treat that value as sensitive information; an attacker could use it to tie traffic back to the originating User Equipment, despite the User Equipment having changed identifiers.

7. References

7.1. 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>.

[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>.

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

[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>.

[RFC8910] Kumari, W. and E. Kline, "Captive-Portal Identification in DHCP and Router Advertisements (RAs)", RFC 8910, DOI 10.17487/RFC8910, September 2020, <https://www.rfc-editor.org/info/rfc8910>.

7.2. Informative References

[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, <https://tools.ietf.org/html/draft-pfister-capport-pvd-00>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, DOI 10.17487/RFC4213, October 2005, <https://www.rfc-editor.org/info/rfc4213>.

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

[RFC8908] Pauly, T., Ed. and D. Thakore, Ed., "Captive Portal API", RFC 8908, DOI 10.17487/RFC8908, September 2020, <https://www.rfc-editor.org/info/rfc8908>.

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 an 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. This has been referred to as a "canary" request because, like the canary in the coal mine, it can be the first sign that something is wrong.

Another test that can be performed is a DNS lookup to a known address with an expected answer. If the answer differs from the expected answer, the equipment detects that a captive portal is present. DNS queries over TCP or HTTPS are less likely to be modified than DNS queries over UDP due to the complexity of implementation.

The different tests may produce different conclusions, varying by whether or not the implementation treats both TCP and UDP traffic and by which types of DNS are intercepted.

Malicious or misconfigured networks with a captive portal present may not intercept these canary requests and choose to pass them through or decide to impersonate, leading to the device having a false negative.

Acknowledgments

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

The authors thank Benjamin Kaduk for providing the content related to TLS certificate validation of the API Server.

The authors thank Michael Richardson for providing wording requiring DNSSEC and TLS to operate without the user adding exceptions.

The authors thank various individuals for their feedback on the mailing list and during the IETF 98 hackathon: David Bird, Erik Kline, Alexis La Goulette, Alex Roscoe, Darshak Thakore, and Vincent van Dam.

Authors' Addresses

Kyle Larose
Agilicus
Email: kyle@agilicus.com

David Dolson
Email: ddolson@acm.org

Heng Liu
Google
Email: liucougar@google.com