Workgroup: Internet Engineering Task Force
Internet-Draft:
draft-ietf-capport-architecture-08
Published: 11 May 2020
Intended Status: Informational
Expires: 12 November 2020
Authors: K. Larose D. Dolson H. Liu
Agilicus Google

CAPPORT Architecture

Abstract

This document describes a 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

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.

This document standardizes an architecture for implementing captive portals while addressing most of the problems arising for current captive portal mechanisms. The architecture is 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 mobile telecom.

* 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 when it compares 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 Uniform Resource Identifier [RFC3986] (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 works in response to any Internet protocol, and is not done by modifying protocols in-band. This notification does not carry the portal URI; rather it provides a notification to the User Equipment that it is in a captive state.

* 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 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 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 Device: The network equipment which enforces the traffic restriction. Also known as Enforcement Device.

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 Enforcement Device, 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.

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 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 prevent applications from using networks that do not grant full network access. E.g., a device connected to a mobile network may be connecting to a captive 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) 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 network, (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]). An Enforcement Device SHOULD allow access to any services that User Equipment could need to contact to perform certificate validation, such as OCSP responders, CRLs, and NTP servers; see Section 4.1 of [I-D.ietf-capport-api] for more information. If certificate validation fails, User Equipment MUST NOT proceed with any of the behavior described above.

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 User Equipment 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 and return the state of captivity for the equipment.

At minimum, the API MUST provide: (1) the state of captivity and (2) a URI for the Captive Portal Server.

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 [I-D.ietf-capport-api] 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 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 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 site-specific policy. Typically User Equipment is permitted access to a small number of services 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.

* May signal 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 Captive Portal Server.

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.

An Enforcement Device MUST rate-limit any signal generated in response to these conditions. See Section 7.4 for a discussion of risks related to a Captive Portal Signal.

2.6. Component Diagram

The following diagram shows the communication between each component.

```
o . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . o
. CAPTIVE NETWORK                                                .
.                                            +--------------+   .
. +------------+   Provision API URI         | Provisioning |   .
. |            |<---------------------------+|  Service     |   .
. |   User     |                             +--------------+   .
. | Equipment  |   Query captivity status    +-------------+    .
. |            |+--------------------------->|  API        |    .
. |            |  Captivity status response  |  Server     |    .
. |            |<---------------------------+|             |    .
. |            |                             +------+------+    .
. |            |                                    | Status    .
. |            | Portal UI page requests     +------+------+    .
. |            |+--------------------------->|  Web Portal |    .
. |            | Portal UI pages             |  Server     |    .
. |            |<---------------------------+|             |    .
. +------------+                             |             |    .
.     ^   ^ |                                +-------------+    .
.     |   | | Data to/from ext. network             |         .
.     |   | +-----------------> +---------------+  Allow/Deny   .
.     |   +--------------------+|               |    Rules      .
.     |                         | Enforcement   |     |         .
.     |   Captive Portal Signal | Device        | <---+         .
.     +-------------------------+---------------+               .
.                                      ^ |                      .
.                                      | |                      .
.                          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 from the Web Portal Server to the user.

* Based on user interaction, the Web Portal Server 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.

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 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

Each instance of User Equipment interacting with the Captive Network MUST be given an identifier that is unique among User Equipment interacting 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. 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 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 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 which 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 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 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 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.

3.5. 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 UE.

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 UE makes a request using a different network path. This might happen when UE 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 UE and not dependent on contextual information to function correctly.

Though a URI 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.

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.

4.1. Initial Connection

This section describes a possible workflow 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 URI.)

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 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 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'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 that 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 Captive Portal 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 Signals 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.

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>.

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

[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>.

[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>.

8.2. Informative References

[I-D.ietf-capport-api] Pauly, T. and D. Thakore, "Captive Portal API", Work in Progress, Internet-Draft, draft-ietf-capport-api-06, 31 March 2020, <https://tools.ietf.org/html/draft-ietf-capport-api-06>.

[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>.

[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>.

[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>.

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. 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 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
Google

Email: liucougar@google.com