RTCWEB   Y. Fablet
Internet-Draft   Apple Inc.
Intended status: Informational   J. de Borst
Expires: April 18, 2020   J. Uberti
                                                          Q. Wang
                                                           Google
                                               October 16, 2019

Using Multicast DNS to protect privacy when exposing ICE candidates
draft-ietf-rtcweb-mdns-ice-candidates-04

Abstract

   WebRTC applications collect ICE candidates as part of the process of
   creating peer-to-peer connections. To maximize the probability of a
   direct peer-to-peer connection, client private IP addresses are
   included in this candidate collection. However, disclosure of these
   addresses has privacy implications. This document describes a way to
   share local IP addresses with other clients while preserving client
   privacy. This is achieved by concealing IP addresses with
   dynamically generated Multicast DNS (mDNS) names.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

As detailed in [IPHandling], exposing client private IP addresses by
default to web applications maximizes the probability of successfully
creating direct peer-to-peer connections between clients, but creates
a significant surface for user fingerprinting. [IPHandling]
recognizes this issue, but also admits that there is no current
solution to this problem; implementations that choose to use Mode 3
to address the privacy concerns often suffer from failing or
suboptimal connections in WebRTC applications. This is particularly
an issue on unmanaged networks, typically homes or small offices,
where NAT loopback may not be supported.

This document proposes an overall solution to this problem by
providing a mechanism for WebRTC implementations to register
ephemeral mDNS [RFC6762] names for local private IP addresses, and
then provide those names, rather than the IP addresses, in their ICE
candidates. While this technique is intended to benefit WebRTC
implementations in web browsers, by preventing collection of private
IP addresses by arbitrary web pages, it can also be used by any
endpoint that wants to avoid disclosing information about its local
network to remote peers on other networks.

WebRTC and WebRTC-compatible endpoints [Overview] that receive ICE
candidates with mDNS names will resolve these names to IP addresses
and perform ICE processing as usual. In the case where the endpoint
is a web application, the WebRTC implementation will manage this
resolution internally and will not disclose the actual IP addresses
to the application.

2. Terminology

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 [RFC2119].

3. Description

This section uses the concept of ICE agent as defined in [RFC8445].
In the remainder of the document, it is assumed that each browsing
context (as defined in Section 7.1 of [HTMLSpec]) has its own ICE
agent.

3.1. ICE Candidate Gathering

This section outlines how mDNS should be used by ICE agents to
conceal local IP addresses.

3.1.1. Procedure

For each host candidate gathered by an ICE agent as part of the
gathering process described in [RFC8445], Section 5.1.1, the
candidate is handled as described below.

1. Check whether this IP address satisfies the ICE agent's policy
    regarding whether an address is safe to expose. If so, expose
    the candidate and abort this process.

2. Check whether the ICE agent has previously generated, registered,
    and stored an mDNS hostname for this IP address as per steps 3 to
    5. If it has, skip ahead to step 6.

3. Generate a unique mDNS hostname. The unique name MUST consist of
    a version 4 UUID as defined in [RFC4122], followed by ".local".

4. Register the candidate's mDNS hostname as defined in [RFC6762].
    The ICE agent SHOULD send an mDNS announcement for the hostname,
    but as the hostname is expected to be unique, the ICE agent
    SHOULD skip probing of the hostname.

5. Store the mDNS hostname and its related IP address in the ICE
    agent for future reuse.

6. Replace the IP address of the ICE candidate with its mDNS
    hostname and provide the candidate to the web application.

ICE agents can implement this procedure in any way as long as it
produces equivalent results. An implementation may for instance pre-
register mDNS hostnames by executing steps 3 to 5 and prepopulate an
ICE agent accordingly. By doing so, only step 6 of the above
procedure will be executed at the time of gathering candidates.

In order to prevent web applications from using this mechanism to
query for mDNS support in the local network, the ICE agent SHOULD
still provide mDNS candidates in step 6 even if the local network
does not support mDNS or mDNS registration fails in step 4.

This procedure ensures that an mDNS name is used to replace only one
IP address. Specifically, an ICE agent using an interface with both
IPv4 and IPv6 addresses MUST expose a different mDNS name for each
address.

3.1.2. Implementation Guidance

3.1.2.1. Registration

   Sending the mDNS announcement to the network can be delayed, for
   instance due to rate limits. An ICE agent SHOULD provide the
   candidate to the web application as soon as its mDNS name is
   generated, regardless of whether the announcement has been sent on
   the network.

3.1.2.2. Determining Address Privacy and Server-Reflexive Candidates

   Naturally, an address that is already exposed to the Internet does
   not need to be protected by mDNS, as it can be trivially observed by
   the web server or remote endpoint. However, determining this ahead
   of time is not straightforward; while the fact that an IPv4 address
   is private can sometimes be inferred by its value, e.g., whether it
   is an [RFC1918] address, the reverse is not necessarily true. IPv6
   addresses present their own complications, e.g., private IPv6
   addresses as a result of NAT64 [RFC6146].

   Instead, the determination of whether an address is public can be
   reliably made as part of the ICE gathering process. For each mDNS
   host candidate generated according the guidance above, the usual STUN
   [RFC5389] request is sent to a STUN server. This can be done for
   both IPv4 and IPv6 local addresses, provided that the application has
   configured both IPv4 and IPv6 STUN servers. If the STUN response
   returns the same value as the local IP address, this indicates the
   address is in fact public.

   Regardless of the result, a server-reflexive candidate will be
   generated; the transport address of this candidate is an IP address
   and therefore distinct from the hostname transport address of the
   associated mDNS candidate, and as such MUST NOT be considered
   redundant per the guidance in [RFC8445], Section 5.1.3. To avoid
   accidental IP address disclosure, this server-reflexive candidate
   MUST have its raddr field set to "0.0.0.0"/"::" and its rport field
   set to "9", as discussed in [ICESDP], Section 9.1.

   Once an address has been identified as public, the ICE agent MAY
   cache this information and omit mDNS protection for that address in
   future ICE gathering phases.

3.1.2.3. Special Handling for IPv6 Addresses

   As noted in [IPHandling], private IPv4 addresses are especially
   problematic because of their unbounded lifetime. However, the
   [RFC4941] IPv6 addresses recommended for WebRTC have inherent privacy

protections, namely a short lifetime and the lack of any stateful
information. Accordingly, implementations MAY choose to not conceal
[RFC4941] addresses with mDNS names as a tradeoff for improved peer-
to-peer connectivity.

3.1.2.4. mDNS Candidate Encoding

The mDNS name of an mDNS candidate MUST be used in the connection-
address field of its candidate attribute. However, when an mDNS
candidate would be the default candidate, typically because there are
no other candidates, its mDNS name MUST NOT be used in the
connection-address field of the SDP "c=" line, as experimental
deployment has indicated that many remote endpoints will fail to
handle such a SDP. In this situation, the IP address values
"0.0.0.0"/"::" and port value "9" MUST instead be used in the c= and
m= lines, similar to how the no-candidates case is handled in
[ICESDP], Section 4.3.1.

Any candidates exposed to the application via local descriptions MUST
be identical to those provided during candidate gathering (i.e., MUST
NOT contain private host IP addresses).

3.2. ICE Candidate Processing

This section outlines how received ICE candidates with mDNS names are
processed by ICE agents, and is relevant to all endpoints.

3.2.1. Procedure

For any remote ICE candidate received by the ICE agent, the following
procedure is used:

1. If the connection-address field value of the ICE candidate does
    not end with ".local" or if the value contains more than one ".",
    then process the candidate as defined in [RFC8445].

2. Otherwise, resolve the candidate using mDNS. The ICE agent
    SHOULD set the unicast-response bit of the corresponding mDNS
    query message; this minimizes multicast traffic, as the response
    is probably only useful to the querying node.

3. If it resolves to an IP address, replace the mDNS hostname of the
    ICE candidate with the resolved IP address and continue
    processing of the candidate as defined in [RFC8445].

4. Otherwise, ignore the candidate.

3.2.2. Implementation Guidance

An ICE agent may use a hostname resolver that transparently supports
both Multicast and Unicast DNS. In this case the resolution of a
".local" name may happen through Unicast DNS as noted in [RFC6762],
Section 3.

An ICE agent SHOULD ignore candidates where the hostname resolution
returns more than one IP address.

An ICE agent MAY add additional restrictions regarding the ICE
candidates it will resolve using mDNS, as this mechanism allows
attackers to send ICE traffic to devices with well-known mDNS names.

3.3. Additional Privacy Considerations

The goal of this mechanism is to keep knowledge of private host IP
addresses within the ICE agent while continuing to allow the
application to transmit ICE candidates. Besides keeping private host
IP addresses out of ICE candidates, implementations must take steps
to prevent these IP addresses from being exposed to web applications
through other means.

3.3.1. Statistics

Statistics related to ICE candidates that are accessible to the web
application MUST NOT contain the IP address of a local or remote mDNS
candidate; the mDNS name SHOULD be used instead.

In addition, a peer-reflexive remote candidate may be constructed
from a remote host IP address as a result of an ICE connectivity
check, as described in Section 7.3.1.3 of [RFC8445]. This check may
arrive before the candidate due to signaling or mDNS resolution
delays, as shown in the examples above.

To prevent disclosure of the host IP address to the application in
this scenario, statistics related to ICE candidates MUST NOT contain
the IP address of any peer-reflexive candidate, unless that IP has
already been learned through signaling of a candidate with the same
address and either the same or a different port; this includes cases
where the signaled candidate is discarded as redundant according to
Section 5.1.3 of [RFC8445].

3.3.2. Interactions With TURN Servers

When sending data to a TURN [RFC5766] server, the sending client
tells the server the destination IP and port for the data. This
means that if the client uses TURN to send to an IP that was obtained

by mDNS resolution, the TURN server will learn the underlying host IP
and port, and this information can then be relayed to the web
application, defeating the value of the mDNS wrapping.

To prevent disclosure of the host IP address to a TURN server, the
ICE agent MUST NOT form candidate pairs between its own relay
candidates and remote mDNS candidates. Note that the converse is not
an issue; the ICE agent MAY form candidate pairs between its own mDNS
candidates and remote relay candidates, as in this situation host IPs
will not be sent directly to the TURN server.

This restriction has no effect on connectivity; in the cases where
host IP addresses are private and need to be wrapped with mDNS names,
they will be unreachable from the TURN server, and as noted above,
the reverse path will continue to work normally.

3.3.3. Generated Name Reuse

It is important that use of registered mDNS hostnames is limited in
time and/or scope. Indefinitely reusing the same mDNS hostname
candidate would provide applications an even more reliable tracking
mechanism than the private IP addresses that this specification is
designed to hide. In the case of a web application, the use of
registered mDNS hostnames SHOULD be scoped by the web application
origin, and SHOULD have the lifetime of the page executing the web
application.

3.3.4. Specific Browsing Contexts

As noted in [IPHandling], privacy may be breached if a web
application running in two browsing contexts can determine whether it
is running on the same device. While the approach in this document
prevents the application from directly comparing local private IP
addresses, a successful local WebRTC connection can also present a
threat to user privacy. Specifically, when the latency of a WebRTC
connection latency is close to zero, the probability is high that the
two peers are running on the same device.

To avoid this issue, browsers SHOULD NOT register mDNS names for
WebRTC applications running in a third-party browsing context (i.e.,
a context that has a different origin than the top-level browsing
context), or a private browsing context.

3.3.5. Network Interface Enumeration

Even when local IP addresses are not exposed, the number of mDNS
hostname candidates can still provide a fingerprinting dimension.
This is in particular the case for network interfaces with limited

connectivity that will not generate server-reflexive or relay
candidates.

The more mDNS names an endpoint exposes through mDNS hostname
candidates, the higher the fingerprinting risk. One countermeasure
is to limit this number to a small value.

Note that no additional fingerprinting risk is introduced when
restricting mDNS hostname candidates to default route only.

3.3.6. Monitoring of Sessions

A malicious endpoint in the local network may also record other
endpoints who are registering, unregistering, and resolving mDNS
names. By doing so, they can create a session log that shows which
endpoints are communicating, and for how long. If both endpoints in
the session are on the same network, the fact they are communicating
can be discovered.

Mitigation of this threat is beyond the scope of this proposal.

4. Potential Limitations

4.1. Reduced Connectivity

With typical ICE, endpoints on the same network will usually be able
to establish a direct connection between their local IP addresses.
When using the mDNS technique, a direct connection is still possible,
but only if at least one side can properly resolve the provided mDNS
candidates. This may not be possible in all scenarios.

First, some networks may entirely disable mDNS. Second, mDNS queries
have limited scope. On large networks, this may mean that an mDNS
name cannot be resolved if the remote endpoint is too many segments
away.

When mDNS fails, ICE will attempt to fall back to either NAT hairpin,
if supported, or TURN relay if not. This may result in reduced
connectivity, reduced throughput and increased latency, as well as
increased cost in case of TURN relay.

During experimental testing of the mDNS technique across a set of
known mDNS-aware endpoints that had configured a STUN server but not
a TURN server, the observed impact to ICE connection rate was 2%
(relative) when mDNS was enabled on both sides, compared to when mDNS
was only enabled on one side. In this testing, the percentage of
connections that required STUN (i.e., went through a NAT) increased
from 94% to 97%, indicating that mDNS succeeded about half the time,

and fell back to NAT hairpin for the remainder. The most likely
explanation for the overall connection rate drop is that hairpinning
failed in some cases.

One potential mitigation, as discussed in Section 3.3, is to not
conceal candidates created from [RFC4941] IPv6 addresses. This
permits connectivity even in large internal networks or where mDNS is
disabled. Future versions of this document will include experimental
data regarding this option.

4.2. Connection Setup Latency

As noted in Section 3, ICE agents using the mDNS technique are
responsible for registering and resolving mDNS names as part of the
ICE process. These steps may delay establishment of a direct peer-
to-peer connection, compared to when raw local IP addresses are used.

Given that these mDNS registrations and queries are typically
occurring on a local network, any associated delays should be small.
Also, as noted in Section 3.1, pre-registration can be employed to
eliminate gathering delays entirely.

4.3. Backward Compatibility

For the most part, backward compatibility does not present a
significant issue for the mDNS technique. When an endpoint that
supports mDNS communicates with an endpoint that does not, the legacy
endpoint will still provide its local IP addresses, and accordingly a
direct connection can still be attempted, even though the legacy
endpoint cannot resolve the mDNS names provided by the new endpoint.
In the event the legacy endpoint attempts to resolve mDNS names using
Unicast DNS, this may cause ICE to take somewhat longer to fully
complete, but should not have any effect on connectivity or
connection setup time.

However, some legacy endpoints are not fully spec-compliant and can
behave unpredictably in the presence of ICE candidates that contain a
hostname, potentially leading to ICE failure. Some endpoints may
also fail to handle a connectivity check from an address that they
have not received in signaling. During the aforementioned
experimental testing, the connection rate when interacting with
endpoints that provided raw IP addresses (and therefore should be
unaffected) decreased by 3% (relative), presumably for these reasons.

5. Examples

The examples below show how the mDNS technique is used during ICE
processing. The first example shows a simple case, the next two
examples demonstrate how peer-reflexive candidates for local IP
addresses can be created due to timing differences, and the final
example shows a real-world case with IPv4, IPv6, and STUN.

5.1. Normal Handling

In this example, mDNS candidates are exchanged between peers and
resolved normally to obtain the corresponding IP addresses.

```
        ICE Agent 1 (192.0.2.1)          ICE Agent 2 (192.0.2.2)
     <Register mDNS |                          |
       name N1,     |                          |
       192.0.2.1>   |                          |
                    |------- mDNS Candidate N1 ------>|
                    |                          | <Register mDNS
                    |                          |   name N2,
                    |                          |   192.0.2.2>
                    |<------ mDNS Candidate N2 -------|
     <Resolve       |                          | <Resolve
      mDNS name N2> |                          |  mDNS name N1>
                    |<=== STUN check to 192.0.2.1 ====|
                    |==== STUN check to 192.0.2.2 ===>|
                    |                          |
```

The exchange of ICE candidates relies on out-of-band signaling, for
example, the SDP Offer/Answer procedure defined in [ICESDP]. In the
above example, the candidate attributes in the SDP messages to
exchange the mDNS candidates between ICE Agent 1 and 2 are as
follows:

ICE Agent 1:

a=candidate:1 1 udp 2122262783 1f4712db-ea17-4bcf-a596-105139dfd8bf.local
  54596 typ host

ICE Agent 2:

a=candidate:1 1 udp 2122262783 2579ef4b-50ae-4bfe-95af-70b3376ecb9c.local
  61606 typ host

5.2. Peer-reflexive Candidate From Slow Signaling

   In this example, a peer-reflexive candidate is generated because the
   mDNS candidate is signaled after the STUN checks begin.

```
           ICE Agent 1 (192.0.2.1)          ICE Agent 2 (192.0.2.2)
      <Register mDNS |                            |
        name N1,     |                            |
         192.0.2.1>  |                            |
                     |------- mDNS Candidate N1 ------>|
                     |                            | <Resolve
                     |                            |  mDNS name N1>
                     |<=== STUN check to 192.0.2.1 ====|
      prflx candidate |                           | <Register mDNS
    192.0.2.2 created |                           |  name N2,
                     |                            |  192.0.2.2>
                     |<------ mDNS Candidate N2 -------|
                     |                            |
```

5.3. Peer-reflexive Candidate From Slow Resolution

   In this example, a peer-reflexive candidate is generated because the
   mDNS resolution for name N2 does not complete until after the STUN
   checks are received.

```
           ICE Agent 1 (192.0.2.1)          ICE Agent 2 (192.0.2.2)
      <Register mDNS |                                | <Register mDNS
        name N1,     |                                |  name N2,
         192.0.2.1>  |                                |  192.0.2.2>
                     |------- mDNS Candidate N1 ------>|
                     |<------ mDNS Candidate N2 -------|
 <Resolve            |                                | <Resolve
  mDNS               |                                |  mDNS name N1>
   .                 |<=== STUN check to 192.0.2.1 ====|
   .   prflx candidate |                              |
   . 192.0.2.2 created |                              |
  name               |                                |
  N2>                |                                |
```

5.4. IPv4, IPv6, and STUN handling

   This last example demonstrates the overall ICE gathering process for
   two endpoints, each with a private IPv4 address and a public IPv6
   address. They preregister their mDNS names to speed up ICE
   gathering.

```
           ICE Agent 1                       ICE Agent 2
           192.168.1.1          STUN          192.168.1.2
```

```
                 2001:db8::1         Server         2001:db8::2
      ----------------------------------------------------------------------
                       Pre-registration of mDNS names
                         |               |               |
      <Register mDNS     |               |               | <Register mDNS
        name N1.1,       |               |               |  name N2.1,
        192.168.1.1>     |               |               |  192.168.1.2>
      <Register mDNS     |               |               | <Register mDNS
        name N1.2,       |               |               |  name N2.2,
        2001:db8::1>     |               |               |  2001:db8::2>
                         |               |               |
      - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                      ICE Agent 1 sends mDNS candidates
                         |               |               |
              <N1.1> |------- mDNS Candidate C1.1 ----->|
              <N1.2> |------- mDNS Candidate C1.2 ----->|
                         |               |               | <Resolve mDNS
                         |               |               |  name N1.1 to
                         |               |               |  192.168.1.1>
                         |               |               | <Resolve mDNS
                         |               |               |  name N1.2 to
                         |               |               |  2001:db8::1>
                         |               |               |
      - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                ICE Agent 1 sends server-reflexive candidates
                         |               |               |
       <192.168.1.1   |--Binding Req-->|               |
         is 192.0.2.1> |<-Binding Resp--|               |
             <192.0.2.1> |------ srflx Candidate C1.3 ----->|
      <2001:db8::1      |--Binding Req-->|               |
        is 2001:db8::1> |<-Binding Resp--|               |
          <2001:db8::1> |------ srflx Candidate C1.4 ----->| <Discard C1.4
                         |               |               |  as redundant>
                         |               |               |
      - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
              ICE Agent 2 sends mDNS candidates, resolution is slow
                         |               |               |
                         |<------ mDNS Candidate C2.1 ------| <N2.1>
                         |<------ mDNS Candidate C2.2 ------| <N2.2>
        <Resolve mDNS    |               |               |
         name N2.1 ...>  |               |               |
        <Resolve mDNS    |               |               |
         name N2.2 ...>  |               |               |
                         |               |               |
      - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
         ICE Agent 2 sends server-reflexive candidates, resolution completes
                         |               |               |
                         |               |<--Binding Req---| <192.168.1.2
```

```
                  |                  |---Binding Resp->|  is 192.0.2.2>
                  |<----- srflx Candidate C2.3 ------|  <192.0.2.2>
                  |                  |<--Binding Req---|  <2001:db8::2
                  |                  |---Binding Resp->|  is 2001:db8::2>
                  |<----- srflx Candidate C2.4 ------|  <2001:db8::2>
                  |                  |                 |
      <... N2.1 is |                 |                 |
        192.168.1.2> |               |                 |
      <... N2.2 is |                 |                 |
        2001:db8::2, |               |                 |
        discard C2.4> |              |                 |
                  |                  |                 |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                       ICE connectivity checks
                  |                  |                 |
      2001:db8::1 |<============= STUN =============|  2001:db8::2
      2001:db8::1 |============= STUN =============>|  2001:db8::2
      192.168.1.1 |<============= STUN =============|  192.168.1.2
      192.168.1.1 |============= STUN =============>|  192.168.1.2
        192.0.2.1 |    Failed <-- STUN -------------|  192.168.1.2
      192.168.1.1 |---------------STUN --> Failed    |  192.0.2.2
      2001:db8::1 |====== STUN(USE-CANDIDATE) ======>|  2001:db8::2
```

  Ice Agent 1 candidates:

    C1.1: candidate:1 1 udp 2122262783 9b36eaac-bb2e-49bb-bb78-
                    21c41c499900.local 10004 typ host
    C1.2: candidate:2 1 udp 2122262527 76c82649-02d6-4030-8aef-
                    a2ba3a9019d5.local 10006 typ host
    C1.3: candidate:1 1 udp 1686055167 192.0.2.1
                    30004 typ srflx raddr 0.0.0.0 rport 0
    C1.4: candidate:2 1 udp 1686054911 2001:db8::1
                    10006 typ srflx raddr 0.0.0.0 rport 0

  Ice Agent 2 candidates:

    C2.1: candidate:1 1 udp 2122262783 b977f597-260c-4f70-9ac4-
                    26e69b55f966.local 20004 typ host
    C2.2: candidate:2 1 udp 2122262527 ac4595a7-7e42-4e85-85e6-
                    c292abe0e681.local 20006 typ host
    C2.3: candidate:1 1 udp 1686055167 192.0.2.2
                    40004 typ srflx raddr 0.0.0.0 rport 0
    C2.4: candidate:2 1 udp 1686054911 2001:db8::2
                    20006 typ srflx raddr 0.0.0.0 rport 0

6. Security Considerations

6.1. mDNS Message Flooding

The implementation of this proposal requires the mDNS querying
capability of the browser for registering mDNS names or adding remote
ICE host candidates with such names. It also requires the mDNS
responding capability of either the browser or the operating platform
of the browser for registering, removing or resolving mDNS names. In
particular,

o the registration of name requires optional probing queries and
   mandatory announcing responses ([RFC6762], Section 8), and this is
   performed at the beginning of ICE gathering;

o the addition of remote ICE host candidates with mDNS names
   generates mDNS queries for names of each candidate;

o the removal of names could happen when the browsing context of the
   ICE agent is destroyed in an implementation, and goodbye responses
   should be sent to invalidate records generated by the ICE agent in
   the local network ([RFC6762], Section 10.1).

A malicious Web application could flood the local network with mDNS
messages by:

o creating browsing contexts that create ICE agents and start
   gathering of local ICE host candidates;

o destroying these local candidates soon after the name registration
   is done;

o adding fictitious remote ICE host candidates with mDNS names.

[RFC6762] defines a general per-question and per-record multicast
rate limiting rule, in which a given question or record on a given
interface cannot be sent less than one second since its last
transmission. This rate limiting rule however does not mitigate the
above attacks, in which new names, hence new questions or records,
are constantly created and sent. Therefore, a browser-wide mDNS
message rate limit MUST be provided for all mDNS queries and
responses that are dispatched during the ICE candidate gathering and
processing described in Section 3. A browser MAY implement more
specific rate limits, e.g., to ensure a single origin does not
prevent other origins from registering, unregistering, or resolving
mDNS names.

6.2. Malicious Responses to Deny Name Registration

   If the optional probing queries are implemented for the name
   registration, a malicious endpoint in the local network, which is
   capable of responding mDNS queries, could send responses to block the
   use of the generated names. This would lead to the discarding of
   this ICE host candidate as in Step 5 in Section 3.1.

   The above attack can be mitigated by skipping the probing when
   registering a name, which also conforms to Section 8 in [RFC6762],
   given that the name is randomly generated for the probabilistic
   uniqueness (e.g. a version 4 UUID) in Step 3 in Section 3.1.
   However, a similar attack can be performed by exploiting the negative
   responses (defined in [RFC6762], Section 8.1), in which NSEC resource
   records are sent to claim the nonexistence of records related to the
   gathered ICE host candidates.

   The existence of malicious endpoints in the local network poses a
   generic threat, and requires dedicated protocol suites to mitigate,
   which is beyond the scope of this proposal.

6.3. Unsolicited ICE Communications

   As noted in Section 4.2 of [RTCWebSecurity], an attacker may use ICE
   as a way to send unsolicited network traffic to specific endpoints.
   While this is not specific to mDNS hostname candidates, this
   technique makes it easier to target devices with well-known mDNS
   names.

   As noted in Section 3.2, ICE agents may decide to not resolve mDNS
   names, for example, if these names are not in the format defined by
   Section 3.1.

7. IANA Considerations

   This document requires no actions from IANA.

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>.

   [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally
              Unique IDentifier (UUID) URN Namespace", RFC 4122,
              DOI 10.17487/RFC4122, July 2005,
              <https://www.rfc-editor.org/info/rfc4122>.

   [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy
              Extensions for Stateless Address Autoconfiguration in
              IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007,
              <https://www.rfc-editor.org/info/rfc4941>.

   [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing,
              "Session Traversal Utilities for NAT (STUN)", RFC 5389,
              DOI 10.17487/RFC5389, October 2008,
              <https://www.rfc-editor.org/info/rfc5389>.

   [RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using
              Relays around NAT (TURN): Relay Extensions to Session
              Traversal Utilities for NAT (STUN)", RFC 5766,
              DOI 10.17487/RFC5766, April 2010,
              <https://www.rfc-editor.org/info/rfc5766>.

   [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762,
              DOI 10.17487/RFC6762, February 2013,
              <https://www.rfc-editor.org/info/rfc6762>.

   [RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive
              Connectivity Establishment (ICE): A Protocol for Network
              Address Translator (NAT) Traversal", RFC 8445,
              DOI 10.17487/RFC8445, July 2018,
              <https://www.rfc-editor.org/info/rfc8445>.

8.2. Informative References

   [HTMLSpec]
              "HTML Living Standard", n.d.,
              <https://html.spec.whatwg.org>.

   [ICESDP]   Keranen, A., "Session Description Protocol (SDP) Offer/
              Answer procedures for Interactive Connectivity
              Establishment (ICE)", April 2018,
              <https://tools.ietf.org/html/draft-ietf-mmusic-ice-sip-
              sdp>.

   [IPHandling]
              Shieh, G., "WebRTC IP Address Handling Requirements",
              April 2018, <https://tools.ietf.org/html/draft-ietf-
              rtcweb-ip-handling>.

[Overview]
           Alvestrand, H., "Overview: Real Time Protocols for
           Browser-based Applications", November 2017,
           <https://tools.ietf.org/html/draft-ietf-rtcweb-overview>.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G.,
           and E. Lear, "Address Allocation for Private Internets",
           BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996,
           <https://www.rfc-editor.org/info/rfc1918>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful
           NAT64: Network Address and Protocol Translation from IPv6
           Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146,
           April 2011, <https://www.rfc-editor.org/info/rfc6146>.

[RTCWebSecurity]
           Rescorla, E., "Security Considerations for WebRTC",
           January 2018,
           <https://tools.ietf.org/html/draft-ietf-rtcweb-security>.

[WebRTCSpec]
           Bruaroey, J., "The WebRTC specification", n.d.,
           <https://w3c.github.io/webrtc-pc/>.

Authors' Addresses

   Youenn Fablet
   Apple Inc.

   Email: youenn@apple.com

   Jeroen de Borst
   Google

   Email: jeroendb@google.com

   Justin Uberti
   Google

   Email: juberti@google.com

   Qingsi Wang
   Google

   Email: qingsi@google.com