SIPCORE
Internet-Draft
Intended status: Standards Track
Expires: October 9, 2019

E. Burger
Georgetown University
B. Nagda
Massachusetts Institute of Technology

April 7, 2019

A Session Initiation Protocol (SIP) Response Code for Rejected Calls
draft-ietf-sipcore-rejected-06

Abstract

This document defines the 608 (Rejected) SIP response code. This response code enables calling parties to learn that an intermediary rejected their call attempt. The call will not be answered. As a 6xx code, the caller will be aware that future attempts to contact the same UAS will likely fail. The present use case driving the need for the 608 response code is when the intermediary is an analytics engine. In this case, the rejection is by a machine or other process. This contrasts with the 607 (Unwanted) SIP response code, which a human at the target UAS indicated the call was not wanted. In some jurisdictions this distinction is important. This document also defines the use of the Call-Info header field in 608 responses to enable rejected callers to contact entities that blocked their calls in error. This provides a remediation mechanism for legal callers that find their calls blocked.

Table of Contents

1. Introduction

The IETF has been addressing numerous issues surrounding how to handle unwanted and, depending on the jurisdiction, illegal calls [RFC5039]. Technologies such as STIR [RFC7340] and SHAKEN [SHAKEN] address the cryptographic signing and attestation, respectively, of signaling to ensure the integrity and authenticity of the asserted caller identity.

This document describes a new SIP response code, 608, which allows calling parties to learn that an intermediary rejected their call. As described below, we need a distinct indicator to differentiate between a user rejection and an intermediary's rejection of a call. In some jurisdictions, calls, even if unwanted by the user, may not be blocked unless there is an explicit user request. Moreover, users may misidentify the nature of a caller.

For example, a legitimate caller may call a user who finds the call to be unwanted. However, instead of marking the call as unwanted, the user may mark the call as illegal. With that information, an analytics engine may determine that all calls from that source should be blocked. However, in some jurisdictions blocking calls from that source for other users may not be legal. Likewise, one can envision jurisdictions that allow an operator to block such calls, but only if there is a remediation mechanism in place to address false positives.

Some call blocking services may return responses such as 604 (Does Not Exist Anywhere). This might be a strategy to try to get a destination's address removed from a calling database. However, other network elements might also interpret this to mean the user truly does not exist and might result in the user not being able to receive calls from anyone, even if wanted. In many jurisdictions, providing such false signaling is also illegal.

The 608 response code addresses this need of remediating falsely blocked calls. Specifically, this code informs the SIP User Agent Client (UAC) that an intermediary blocked the call and provides a redress mechanism that allows callers to contact the operator of the intermediary.

In the current call handling ecosystem, users can explicitly reject a call or later mark a call as being unwanted by issuing a 607 SIP response code (Unwanted) [RFC8197]. Figure 1 and Figure 2 show the operation of the 607 SIP response code. The UAS indicates the call was unwanted. As RFC8197 explains, not only does the called party desire to reject that call, they can let their proxy know that they consider future calls from that source unwanted. Upon receipt of the 607 response from the UAS, the proxy may send call information to a

call analytics engine. For various reasons described in RFC8197, if a network operator receives multiple reports of unwanted calls, that may indicate that the entity placing the calls is likely to be a source of unwanted calls for many people. As such, other customers of the service provider may want the service provider to automatically reject calls on their behalf.

Another value of the 607 rejection is presuming the proxy forwards the response code to the UAC, the calling UAC or intervening proxies will also learn the user is not interested in receiving calls from that sender.

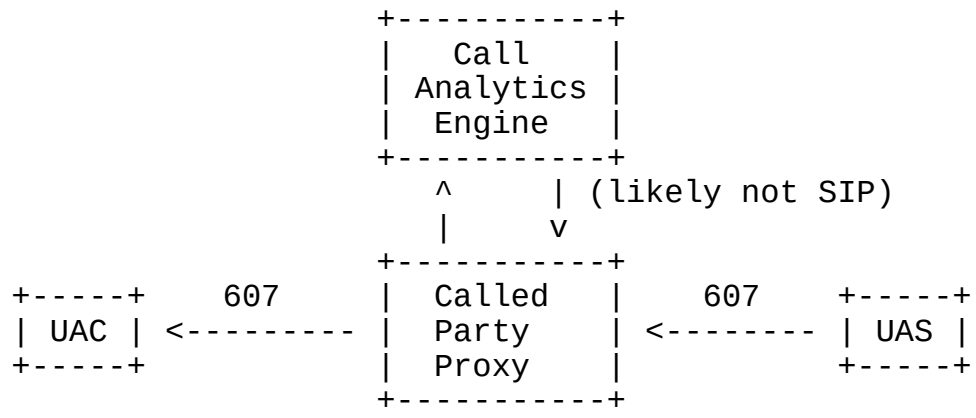

Figure 1: Unwanted (607) Call Flow

For calls rejected with a 607 from a legitimate caller, receiving a 607 response code can inform the caller to stop attempting to call the user. Moreover, if a legitimate caller believes the user is rejecting their calls in error, they can use other channels to contact the user. For example, if a pharmacy calls a user to let them know their prescription is available for pickup and the user mistakenly thinks the call is unwanted and issues a 607 response code, the pharmacy, having an existing relationship with the customer, can send the user an email or push a note to the pharmacist to ask the customer to consider not rejecting their calls in the future.

Many systems that allow the user to mark the call unwanted (e.g., with the 607 response code) also allow the user to change their mind and unmark such calls. This mechanism is relatively easy to implement as the user usually has a direct relationship with the service provider that is blocking calls.

However, things become more complicated if an intermediary, such as a third-party provider of call management services that classifies

calls based on the relative likelihood that the call is unwanted, misidentifies the call as unwanted. Figure 3 shows this case. Note that the UAS typically does not receive an INVITE since the called party proxy rejects the call on behalf of the user. In this situation, it would be beneficial for the caller to learn who rejected the call, so they can correct the misidentification.

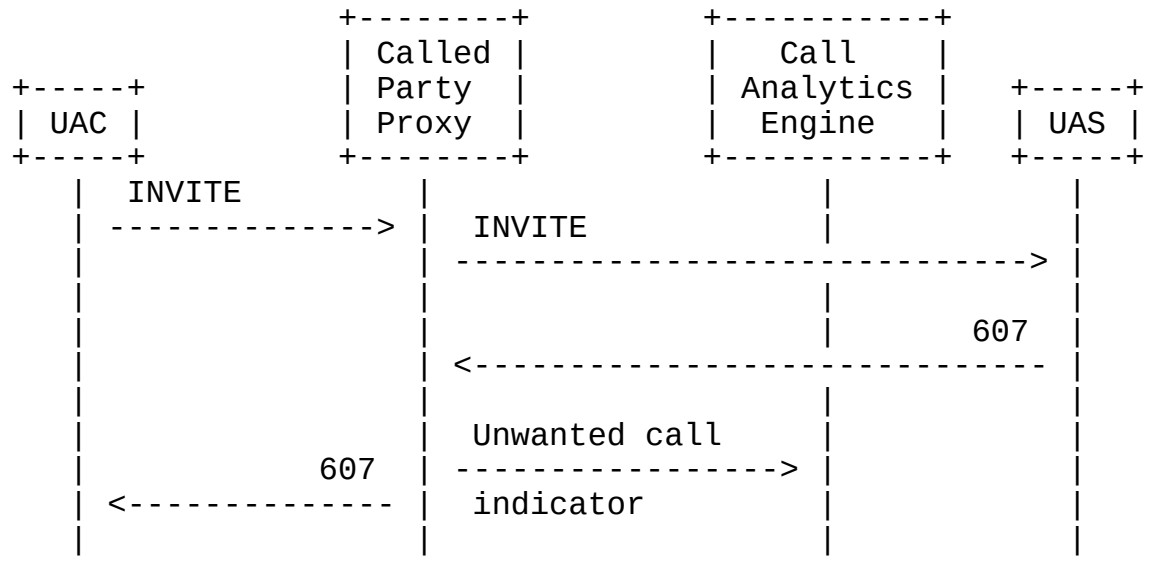

Figure 2: Unwanted (607) Ladder Diagram

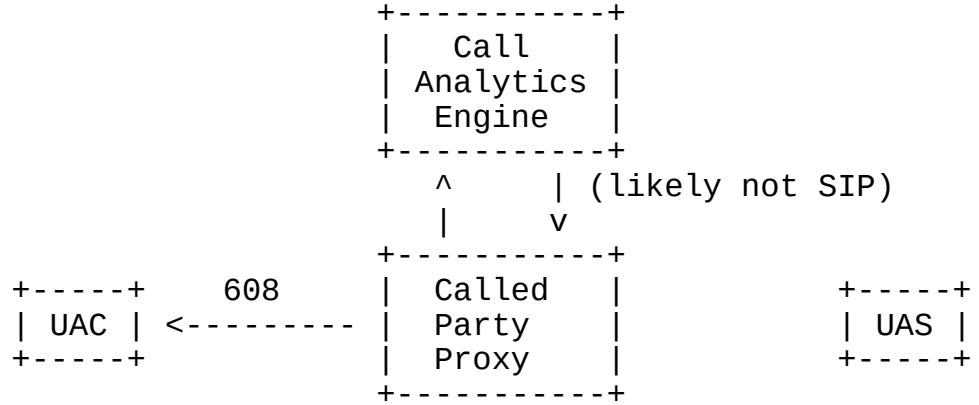

Figure 3: Rejected (608) Call Flow

In this situation, one might be tempted to have the intermediary use the 607 response code. 607 indicates to the caller the subscriber does not want the call. However, RFC8197 specifies that one of the uses of 607 is to inform analytics engines that a user (human) has

rejected a call. The problem here is that network elements downstream from the intermediary might interpret the 607 as coming from a user (human) that has marked the call as unwanted, as opposed to coming from an algorithm using statistics or machine learning to reject the call. An algorithm can be vulnerable to the base rate fallacy base rate fallacy [BaseRate] algorithm rejecting the call. In other words, those downstream entities should not rely on another entity 'deciding' the call is unwanted. By distinguishing between a (human) user rejection and an intermediary engine's statistical rejection, a downstream network element that sees a 607 response code can weight it as a human rejection in its call analytics.

It is useful for blocked callers to have a redress mechanism. One can imagine that some jurisdictions will require it. However, we must be mindful that most of the calls that will be blocked will, in fact, be illegal and eligible for blocking. Thus, providing alternate contact information for a user would be counterproductive to protecting that user from illegal communications. This is another reason we do not propose to simply allow alternate contact information in a 607 response message.

One might ask why we cannot use the same mechanism an analytics service provider offers their customers that lets them correct a call blocked in error? The reason is while there is an existing relationship between the customer (called party) and the analytics service provider, it is unlikely there is a relationship between the caller and the analytics service provider. Moreover, there are numerous call blocking providers in the ecosystem. As such, we need a mechanism for indicating an intermediary rejected a call that also provides contact information for the operator of that intermediary, without exposing the target user's contact information.

The protocol described in this document uses existing IETF protocol mechanisms for specifying the redress mechanism. In the Call-Info header passed back to the UAC, we send additional information specifying a redress address. We choose to encode the redress address using jCard [RFC7095]. Conveniently, we use jCard rather than vCard [RFC6350] as we have a standard marshaling mechanism for creating a canonical representation of a JSON [RFC8259] object, such as a jCard, and a standard presentation format for such an object, namely JWS [RFC7515]. The SIP community is familiar with this concept as it is the mechanism used by STIR [RFC8224].

The jCard encoding might seem unnecessary at first, but it is essential to preventing potential network attacks. Suppose, for example, that the redress address was simply passed as a header value. One can imagine an adverse agent that maliciously spoofs a 608 response with the same redress address to a large number of

active callers, who may then all send redress requests to the specified address (the basis for a denial-of-service attack). The process would occur as follows: (1) a malicious agent senses INVITE requests from a variety UACs and (2) spoofs 608 responses with an unsigned redress address before the intended receivers can respond, causing (3) the UACs to all contact the redress address at once. The jCard encoding allows the UAC to verify the blocking intermediary's identity before contacting the redress address. This guards against a malicious agent spoofing 608 responses, preventing the denial-of-service attack. Thus, if the jCard address is unreachable or undecipherable, either (1) a malicious agent is lying about the jCard or (2) the redress mechanism is misconfigured.

2. Terminology

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.

3. Protocol Operation

For clarity, this section uses the term 'intermediary' as the entity that acts as a SIP User Agent Server (UAS) on behalf of the user in the network, as opposed to the user's UAS (colloquially, but not necessarily, their phone). The intermediary could be a back-to-back user agent (B2BUA) or a SIP Proxy.

Figure 4 shows an overview of the call flow for a rejected call.

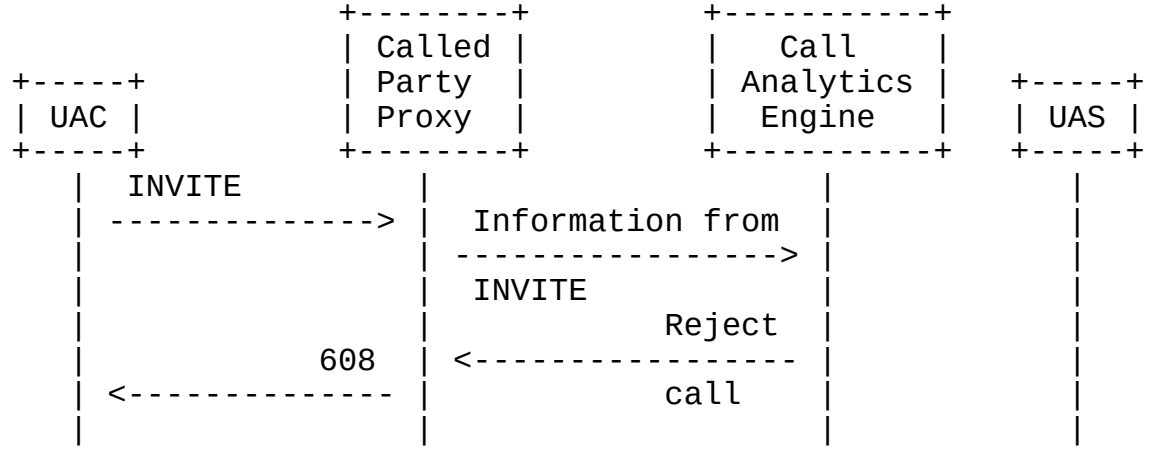

Figure 4: Rejected (608) Ladder Diagram

3.1. Intermediary Operation

An intermediary MAY issue the 608 response code in a failure response for an INVITE, MESSAGE, SUBSCRIBE, or other out-of-dialog SIP [RFC3261] request to indicate that an intermediary rejected the offered communication as unwanted by the user. An intermediary MAY issue the 608 as the value of the "cause" parameter of a SIP reason-value in a Reason header field [RFC3326].

If an intermediary issues a 608 code and there are not indicators the calling party will use the contents of the Call-Info header field for malicious purposes (see Section 6), the intermediary MUST include a Call-Info header field in the response.

If there is a Call-Info header field, it MUST have the ’purpose’ parameter of ’jwscard’. The value of the Call-Info header field MUST refer to a valid JWS [RFC7515] encoding of a jCard [RFC7095] object.

Proxies need to be mindful that a downstream intermediary may reject the attempt with a 608 while other paths may still be in progress. In this situation, the requirements stated in Section 16.7 of RFC3261 [RFC3261] apply. Specifically, the proxy should cancel pending transactions and must not create any new branches. Note this is not a new requirement but simply pointing out the existing 6xx protocol mechanism in SIP.

3.2. jCard Construction

The intermediary constructs the JWS as follows.

3.2.1. JOSE Header

The JOSE header MUST include the typ, alg, and x5u parameters from JWS [RFC7515]. The typ parameter MUST have the value "vcard+json". Implementations MUST support ES256 as JWA [RFC7518] defines it, and MAY support other registered signature algorithms. Finally, the x5u parameter MUST be a URI that resolves to the public key certificate corresponding to the key used to digitally sign the JWS.

3.2.2. JWT Payload

The payload contains two JSON values. The first JWT claim that MUST be present is the iat (issued at) claim [RFC7519]. The "iat" MUST be set to the date and time of the issuance of the 608 response. This mandatory component protects the response from replay attacks.

The second JWT claim that MUST be present is the jcard claim. Section 5.3 describes the registration. In the construction of the

jcard claim, the "jcard" MUST include at least one of the URL, EMAIL, TEL, or ADR properties. UACs supporting this specification MUST be prepared to receive a full jCard. Call originators (at the UAC) can use the information returned by the jCard to contact the intermediary that rejected the call to appeal the intermediary's blocking of the call attempt. What the intermediary does if the blocked caller contacts the intermediary is outside the scope of this document.

3.2.3. JWS Signature

JWS [RFC7515] specifies the procedure for calculating the signature over the jCard JWT. Section 4 of this document has a detailed example on constructing the JWS, including the signature.

3.3. UAC Operation

A UAC conforming to this specification MUST include the sip.608 feature capability indicator in the Feature-Caps header field of the INVITE request.

Upon receiving a 608 response, UACs perform normal SIP processing for 6xx responses.

3.4. Legacy Interoperation

If the UAC indicates support for 608 and the intermediary issues a 608, life is good as the UAC will receive all the information it needs to remediate an erroneous block by an intermediary. However, what if the UAC does not understand 608? For example, how can we support callers from a legacy, non-SIP public switched network connecting to the SIP network via a media gateway?

We address this situation by having the first network element that conforms with this specification play an announcement in the media. See Section 3.5 for requirements on the announcement. The simple rule is a network element that inserts the sip.608 feature capability MUST be able to convey at a minimum how to contact the operator of the intermediary that rejected the call attempt.

The degenerate case is the intermediary is the only element that understands the semantics of the 608 response code. Obviously, any SIP device will understand that a 608 response code is a 6xx error. However, there are no other elements in the call path that understand the meaning of the value of the Call-Info header field. The intermediary knows this is the case as the INVITE request will not have the sip.608 feature capability. In this case, one can consider the intermediary to be the element 'inserting' a virtual sip.608

feature capability. If the caveats described in Section 3.5 and Section 6 do not hold, the intermediary MUST play the announcement.

Now we take the case where a network element that understands the 608 response code receives an INVITE for further processing. A network element conforming with this specification MUST insert the sip.608 feature capability, per the behaviors described in Section 4.2 of [RFC6809].

Do note that even if a network element plays an announcement describing the contents of the 608 response message, the network element MUST forward the 608 response code message as the final response to the INVITE.

One aspect of using a feature capability is that only the network elements that will either consume (UAC) or play an announcement (media gateway, session border controller (SBC [RFC7092]), or proxy) need to understand the sip.608 feature capability. The rest of the infrastructure does not need to be modified, assuming that the other network elements conform to Section 16.6 of [RFC3261], specifically they will pass headers such as "Feature-Caps: *;+sip.608" unmodified.

3.5. Announcement Requirements

There are a few requirements on the element that handles the announcement for legacy interoperation.

As noted above, the element that inserts the sip.608 feature capability is responsible for conveying the information referenced by the Call-Info header field in the 608 response message. However, this specification does not mandate how to convey that information.

Let us take the case where a telecommunications service provider controls the element inserting the sip.608 feature capability. It would be reasonable to expect the service provider would play an announcement in the media path towards the UAC (caller). It is important to note the network element should be mindful of the media type requested by the UAC as it formulates the announcement. For example, it would make sense for an INVITE that only indicated audio codecs in the SDP [RFC4566] to result in an audio announcement. However, if the INVITE only indicated a real-time text codec and the network element is able to render the information in the requested media format, the network element MUST send the information in a text format, not an audio format.

It is also possible for the network element inserting the sip.608 feature capability to be under the control of the same entity that controls the UAC. For example, a large call center might have legacy

UACs, but have a modern outbound calling proxy that understands the full semantics of the 608 response code. In this case, it is enough for the outbound calling proxy to digest the Call-Info information and handle the information digitally, rather than 'transcoding' the Call-Info information for presentation to the caller.

4. Examples

These examples are not normative, do not include all protocol elements, and may have errors. Review the protocol documents for actual syntax and semantics of the protocol elements.

4.1. Full Exchange

Given an INVITE (shamelessly taken from [SHAKEN]):

```
INVITE sip:+12155550113@tel.one.example.net SIP/2.0
Max-Forwards: 69
Contact: <sip:+12155550112@2001:db8::12:50207;rinstance=9da3088f36cc>
To: <sip:+12155550113@tel.one.example.net>
From: "Alice" <sip:+12155550112@tel.two.example.net>;tag=614bdb40
Call-ID: 79048YzkxNDA5NTI1MzA0OWFjOTFkMmFlODhiNTI2OWQ1ZTI
P-Asserted-Identity: "Alice"<sip:+12155550112@tel.two.example.net>,
    <tel:+12155550112>
CSeq: 2 INVITE
Allow: SUBSCRIBE, NOTIFY, INVITE, ACK, CANCEL, BYE, REFER, INFO,
    MESSAGE, OPTIONS
Content-Type: application/sdp
Date: Tue, 16 Aug 2016 19:23:38 GMT
Feature-Caps: *;+sip.608
Identity:
eyJhbGciOiJFUzI1NiIsInR5cCI6InBhc3Nwb3J0IiwicHB0Ijoic2hha2VuIiwieDV1I
joiaHR0cDovL2NlcnQtYXV0aC5wb2Muc3lzLmNvbWNhc3QubmV0L2V4YW1wbGUuY2VydC
J9eyJhdHRlc3QiOiJBIiwiZGVzdCI6eyJ0biI6IisxMjE1NTU1MTIxMyJ9LCJpYXQiOiI
xNDcxMzc1NDE4Iiwib3JpZyI6eyJ0biI64oCdKzEyMTU1NTUxMjEyIn0sIm9yaWdpZCI6
IjEyM2U0NTY3LWU4OWItMTJkMy1hNDU2LTQyNjY1NTQ0MDAwMCJ9._28kAwRWnheXyA6n
Y4MvmK5JKHZH9hSYkWI4g75mnq9Tj2lW4WPm0PlvudoGaj7wM5XujZUTb_3MA4modoDtC
A;info=<http://cert.example2.net/example.cert>;alg=ES256
Content-Length: 153

v=0
o=- 13103070023943130 1 IN IP6 2001:db8::177
c=IN IP6 2001:db8::177
t=0 0
m=audio 54242 RTP/AVP 0
a=sendrecv
```

An intermediary could reply:

```
SIP/2.0 608 Rejected
Via: SIP/2.0/UDP 2001:db8::177:60012;branch=z9hG4bK-524287-1
From: "Alice" <sip:+12155550112@tel.two.example.net>;tag=614bdb40
To: <sip:+12155550113@tel.one.example.net>
Call-ID: 79048YzkxNDA5NTI1MzA0OWFjOTFkMmFlODhiNTI2OWQ1ZTI
CSeq: 2 INVITE
Call-Info: <https://block.example.net/complaint.json>;purpose=jwscard
```

The location https://block.example.net/complaint.json resolves to a JWS. The JWS would be constructed as follows.

The JWS header of this example jCard could be:

```
{ {"alg":"ES256"},
  {"typ":"vcard+json"},
  {"x5u":"https://certs.example.net/reject_key.cer"} }
```

Now, let us construct a minimal jCard. For this example, the jCard refers the caller to an email address, bitbucket@blocker.example.net:

```
["vcard",
  [
    ["version", {}, "text", "4.0"],
    ["fn", {}, "text", "Robocall Adjudication"],
    ["email", {"type":"work"},
              "text", "bitbucket@blocker.example.net"]
  ]
]
```

With this jCard, we can now construct the JWT:

```
{
  "iat":1546008698,
  "jcard":["vcard",
    [
      ["version", {}, "text", "4.0"],
      ["fn", {}, "text", "Robocall Adjudication"],
      ["email", {"type":"work"},
                "text", "bitbucket@blocker.example.net"]
    ]
  ]
}
```

In order to calculate the signature, we need to encode the JOSE header and JWT into base64. As an implementation note, one can trim

whitespace in the JSON objects to save a few bytes. UACs MUST be prepared to receive pretty printed, compact, or bizarrely formatted JSON. For the purposes of this example, we leave the objects with pretty whitespace. Speaking of pretty vs. machine formatting, these examples have line breaks in the base64 encodings for ease of publication in the RFC format. The specification of base64 allows for these line breaks and the decoded text works just fine. However, those extra line break octets would affect the calculation of the signature. As such, implementations MUST NOT insert line breaks into the base64 encodings of the JOSE header or JWT. This also means UACs MUST be prepared to receive arbitrarily long octet streams from the URI referenced by the Call-Info SIP header.

base64 of JOSE header:

```
eyB7ImFsZyI6IkVTMjU2In0sCiAgeyJ0eXAiOiJ2Y2FyZCtqc29uIn0sCiAgeyJ4
NXUiOiJodHRwczovL2NlcnRzLmV4YW1wbGUubmV0L3JlamVjdF9rZXkuY2VyIn0g
fQo=
```

base64 of JWT:

```
ewogICJpYXQiOjE1NDYwMDg2OTgsCiAgImpjYXJkIjpbInZjYXJkIiwKICAgIFsK
ICAgICAgWyJ2ZXJzaW9uIiwge30sICJ0ZXh0IiwgIjQuMCJdLAogICAgICBbImZu
Iiwge30sICJ0ZXh0IiwgIlJvYm9jYWxsIEFkanVkaWNhdGlvbiJdLAogICAgICBb
ImVtYWlsIiwgeyJ0eXBlIjoid29yayJ9LCAKICAgICAgICAgICAgICAgICJ0ZXh0
IiwgImJpdGJ1Y2tldEBibG9ja2VyLmV4YW1wbGUubmV0Il0KICAgIF0KICBdCn0K
```

In this case, the object to be signed (remembering this is just a single, long line; the line breaks are for ease of review but do not appear in the actual text being signed is as follows:

```
eyB7ImFsZyI6IkVTMjU2In0sCiAgeyJ0eXAiOiJ2Y2FyZCtqc29uIn0sCiAgeyJ4
NXUiOiJodHRwczovL2NlcnRzLmV4YW1wbGUubmV0L3JlamVjdF9rZXkuY2VyIn0g
fQo=
.
ewogICJpYXQiOjE1NDYwMDg2OTgsCiAgImpjYXJkIjpbInZjYXJkIiwKICAgIFsK
ICAgICAgWyJ2ZXJzaW9uIiwge30sICJ0ZXh0IiwgIjQuMCJdLAogICAgICBbImZu
Iiwge30sICJ0ZXh0IiwgIlJvYm9jYWxsIEFkanVkaWNhdGlvbiJdLAogICAgICBb
ImVtYWlsIiwgeyJ0eXBlIjoid29yayJ9LCAKICAgICAgICAgICAgICAgICJ0ZXh0
IiwgImJpdGJ1Y2tldEBibG9ja2VyLmV4YW1wbGUubmV0Il0KICAgIF0KICBdCn0K
```

We use the following X.509 PKCS #8-encoded ECDSA private key, also shamelessly taken from [SHAKEN]), as an example key for signing the hash of the above text. Do NOT use this key in real life! It is for example purposes only. At the very least, we would strongly recommend the key being encrypted at rest.

```
-----BEGIN PRIVATE KEY-----
MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgi7q2TZvN9VDFg8Vy
qCP06bETrR2v8MRvr89rn4i+UAahRANCAAQWfaj1HUETpoNCrOtp9KA8o0V79IuW
ARKt9C1cFPkyd3FBP4SeiNZxQhDrD0tdBHls3/wFe8++K2FrPyQF9vuh
-----END PRIVATE KEY-----
```

The resulting JWS, using the above key on the above object, renders the following ECDSA P-256 SHA-256 digital signature.

```
MEUCIQCF2nv/eKvnGQNELZglQTpWbYtzbEf97xH4zKnkLx7S0QIgIl2f5ehMOwjM
TS+skjf1163ihH5+yIHQS3quklEt/9o=
```

Thus, the JWS stored at https://blocker.example.net/complaints.json, would contain:

```
eyB7ImFsZyI6IkVTMjU2In0sCiAgeyJ0eXAiOiJ2Y2FyZCtqc29uIn0sCiAgeyJ4
NXUiOiJodHRwczovL2NlcnRzLmV4YW1wbGUubmV0L3JlamVjdF9rZXkuY2VyIn0g
fQo=.ewogICJpYXQiOjE1NDYwMDg2OTgsCiAgImpjYXJkIjpbInZjYXJkIiwKICA
gIFsKICAgICAgWyJ2ZXJzaW9uIiwge30sICJ0ZXh0IiwgIjQuMCJdLAogICAgICB
bImZuIiwge30sICJ0ZXh0IiwgIlJvYm9jYWxsIEFkanVkaWNhdGlvbiJdLAogICA
gICBbImVtYWlsIiwgeyJ0eXBlIjoid29yayJ9LCAKICAgICAgICAgICAgICAgICJ
0ZXh0IiwgImJpdGJ1Y2tldEBibG9ja2VyLmV4YW1wbGUubmV0Il0KICAgIF0KICB
dCn0K.MEUCIQCF2nv/eKvnGQNELZglQTpWbYtzbEf97xH4zKnkLx7S0QIgIl2f5e
hMOwjMTS+skjf1163ihH5+yIHQS3quklEt/9o=
```

4.2. Web Site jCard

For an intermediary that provides a Web site for adjudication, the jCard could contain the following. Note the calculation of the JWS is not shown; the URI reference in the Call-Info header field would be to the JWS of the signed jCard.

```
["vcard",
  [
    ["version", {}, "text", "4.0"],
    ["fn", {}, "text", "Robocall Adjudication"],
    ["url", {"type":"work"},
            "text", "https://blocker.example.net/adjudication-form"]
  ]
]
```

4.3. Multi-modal jCard

For an intermediary that provides a telephone number and a postal address, the jCard could contain the following. Note the calculation of the JWS is not shown; the URI reference in the Call-Info header field would be to the JWS of the signed jCard.

```
["vcard",
  [
    ["version", {}, "text", "4.0"],
    ["fn", {}, "text", "Robocall Adjudication"],
    ["adr", {"type":"work"}, "text",
      ["Argument Clinic",
       "12 Main St","Anytown","AP","000000","Somecountry"]
    ]
    ["tel", {"type":"work"}, "uri", "tel:+1-555-555-0112"]
  ]
]
```

Note that it is up to the UAC to decide which jCard contact modality, if any, it will use.

4.4. Legacy Interoperability

Figure 5 depicts a call flow illustrating legacy interoperability. In this non-normative example, we see a UAC that does not support the full semantics for 608. However, there is an SBC that does support 608. Per RFC6809 [RFC6809], the SBC can insert "*;+sip.608" into the Feature-Caps header field for the INVITE. When the intermediary, labeled "Called Party Proxy" in the figure, rejects the call, it knows it can simply perform the processing described in this document. Since the intermediary saw the sip.608 feature capability, it knows it does not need to send any media describing whom to contact in the event of an erroneous rejection.

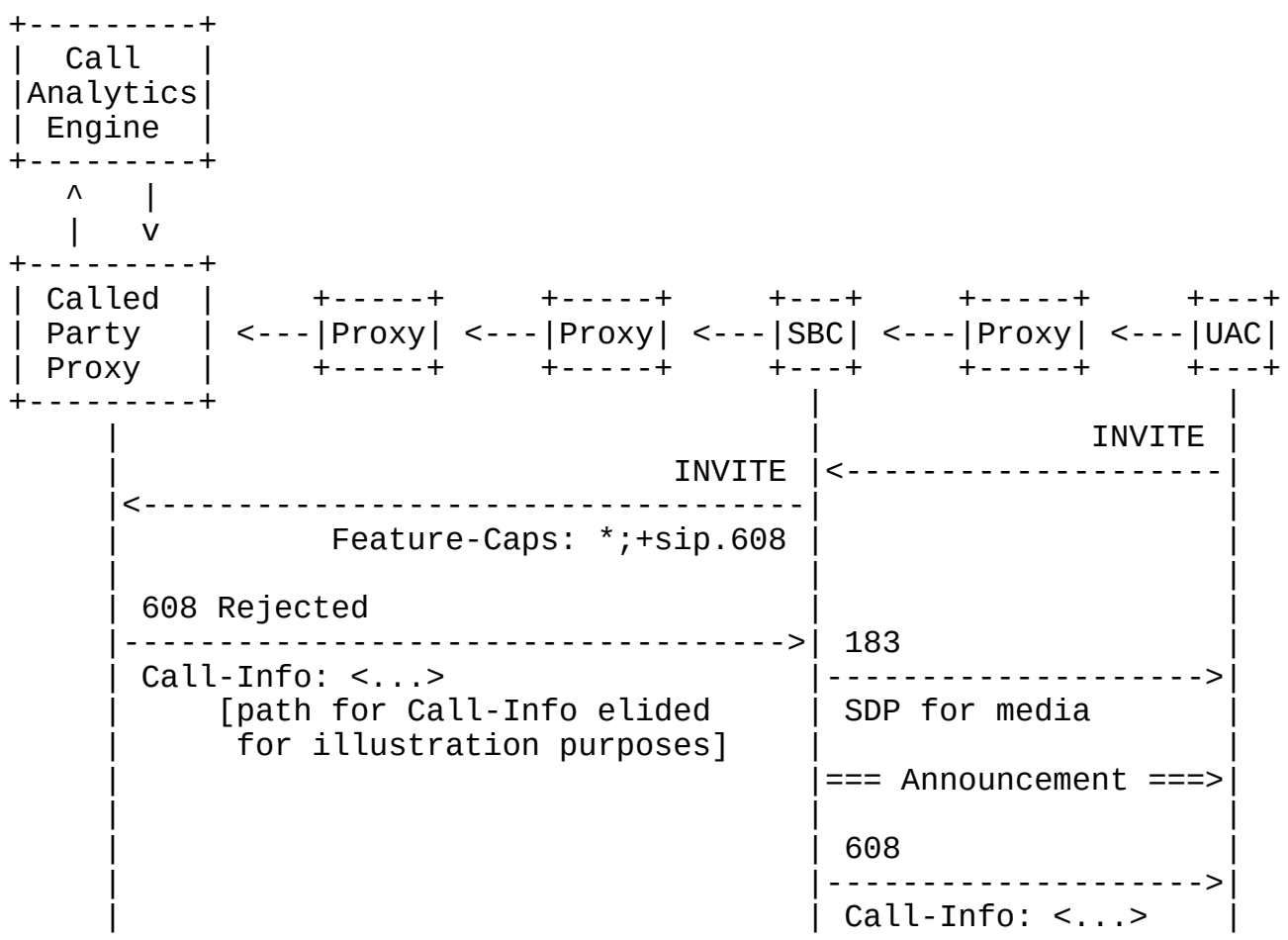

Figure 5: Legacy Operation

When the SBC receives the 608 response code, it correlates that with the original INVITE from the UAC. The SBC remembers that it inserted the sip.608 feature capability, which means it is responsible for somehow alerting the UAC the call failed and whom to contact. At this point the SBC can play a prompt, either natively or through a mechanism such as NETANN [RFC4240], that sends the relevant information in the appropriate media to the UAC.

As an example, the SBC could extract the FN and TEL jCard fields and play something like a special information tone (see Telcordia SR-2275 [SR-2275] section 6.21.2.1 or ITU-T E.180 [ITU.E.180.1998] section 7), followed by "Your call has been rejected by ...", followed by a text-to-speech translation of the FN text, followed by "You can reach them on", followed by a text-to-speech translation of the telephone number in the TEL field.

Note the SBC also still sends the full 608 response code, including the Call-Info header, towards the UAC.

5. IANA Considerations

5.1. SIP Response Code

This document defines a new SIP response code, 608 in the "Response Codes" subregistry of the "Session Initiation Protocol (SIP) Parameters" registry defined in [RFC3261].

Response code: 608

Description: Rejected

Reference: [RFCXXXX]

5.2. SIP Feature-Capability Indicator

This document defines the feature capability sip.608 in the "SIP Feature-Capability Indicator Registration Tree" registry defined in [RFC6809].

Name: sip.608

Description: This feature capability indicator, when included in a Feature-Caps header field of an INVITE request, indicates that the entity associated with the indicator will be responsible for indicating to the caller any information contained in the 608 SIP response code, specifically the value referenced by the Call-Info header.

Reference: [RFCXXXX]

5.3. JSON Web Token Claim

This document defines the new JSON Web Token claim in the "JSON Web Token Claims" sub-registry created by [RFC7519]. Section 3.2.2 defines the syntax. The required information is:

Claim Name: jcard

Claim Description: jCard data

Change Controller: IESG

Reference: [RFCXXXX], [RFC7095]

5.4. Call-Info Purpose

This document defines the new predefined value "jwscard" for the "purpose" header field parameter of the Call-Info header field. This modifies the "Header Field Parameters and Parameter Values" subregistry of the "Session Initiation Protocol (SIP) Parameters" registry by adding this RFC as a reference to the line for the header field "Call-Info" and parameter name "purpose":

Header Field: Call-Info

Parameter Name: purpose

Predefined Values: Yes

Reference: [RFCXXXX]

6. Security Considerations

Intermediary operators need to be mindful of whom they are sending the 608 response. There is a risk that a truly malicious caller is being rejected. This raises two issues. The first is the caller, now alerted that the call is being automatically rejected, may change their call behavior to defeat call blocking systems. The second, and more significant risk, is that by providing a contact in the Call-Info field, the intermediary may be giving the malicious caller a vector for attack. In other words, the intermediary will be publishing an address that a malicious actor may use to launch an attack on the intermediary. Because of this, intermediary operators may wish to configure their response to only include a Call-Info field for INVITE or other initiating methods that are signed and pass validation by STIR [RFC8224].

Another risk is for an attacker to flood a proxy that supports the sip.608 feature with INVITE requests that lack the sip.608 feature capability in order to direct the SDP to a victim’s device. Because the mechanism described here can result in an audio file being sent to the target of the Contact header field, an attacker could use the mechanism described by this document as an amplification attack, given a SIP INVITE can be under 1 kilobyte and an audio file can be hundreds of kilobytes. One remediation for this is for devices that insert a sip.608 feature capability only transmit media to what is highly likely to be the actual source of the call attempt. A method for this is to only play media in response to an INVITE that is signed and passed validation by STIR [RFC8224].

Yet another risk is a malicious entity or the intermediary itself can generate a malicious 608 response with a jCard referring to a

malicious agent. For example, the recipient of a 608 may receive a TEL URI in the vCard. When the recipient calls that address, the malicious agent could ask for personally identifying information. However, instead of using that information to verify the recipient's identity, they are phishing the information for nefarious ends. As such, we strongly recommend the recipient validates to whom they are communicating with if asking to adjudicate an erroneously rejected call attempt. Since we may also be concerned about intermediate nodes modifying contact information, we can address both of these issues with a single solution. The remediation is to require the intermediary to sign the jCard. Signing the jCard provides integrity protection. In addition, one can imagine mechanisms such as used by SHAKEN [SHAKEN] to use signing certificate issuance as a mechanism for traceback to the entity issuing the jCard, for example tying the identity of the subject of the certificate to the To field of the initial SIP request, as if the intermediary was vouching for the From field of a SIP request with that identity.

Since the decision of whether to include Call-Info in the 608 response is a matter of policy, one thing to consider is whether a legitimate caller can ascertain whom to contact without such information being included in the 608. For example, in some jurisdictions, if the terminating service provider is the intermediary, the caller can look up who the terminating service provider is based on the routing information for the dialed number. As such, the Call-Info jCard could be redundant information. However, the factors going into a particular service provider's or jurisdiction's choice of whether or not to include Call-Info is outside the scope of this document.

7. Acknowledgements

This document liberally lifts from [RFC8197] in its text and structure. However, the mechanism and purpose of 608 is quite different than 607. Any errors are the current editor's and not the editor of RFC8197. Thanks also go to Ken Carlberg of the FCC, Russ Housley, Paul Kyzivat, and Tolga Asveren for their suggestions on improving the draft. Tolga's suggestion to provide a mechanism for legacy interoperability served to expand the draft by 50%. In addition, Tolga came up with the jCard attack. Finally, Christer Holmberg as always provided a close reading and fixed a SIP feature capability bug found by Yehoshua Gev.

Finally, Bhavik Nagda provided clarifying edits as well and more especially wrote and tested an implementation of the 608 response code in Kamailio. Code is available at <https://github.com/nagdab/608_Implementation> .

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>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC3326] Schulzrinne, H., Oran, D., and G. Camarillo, "The Reason Header Field for the Session Initiation Protocol (SIP)", RFC 3326, DOI 10.17487/RFC3326, December 2002, <https://www.rfc-editor.org/info/rfc3326>.

[RFC6809] Holmberg, C., Sedlacek, I., and H. Kaplan, "Mechanism to Indicate Support of Features and Capabilities in the Session Initiation Protocol (SIP)", RFC 6809, DOI 10.17487/RFC6809, November 2012, <https://www.rfc-editor.org/info/rfc6809>.

[RFC7095] Kewisch, P., "jCard: The JSON Format for vCard", RFC 7095, DOI 10.17487/RFC7095, January 2014, <https://www.rfc-editor.org/info/rfc7095>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[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

[BaseRate] Bar-Hillel, M., "The Base-Rate Fallacy in Probability Judgements", 4 1977, < https://apps.dtic.mil/docs/citations/ADA045772>.

[ITU.E.180.1998] International Telecommunications Union, "Technical characteristics of tones for the telephone service", ITU Recommendation E.180/Q.35, March 1998.

[RFC4240] Burger, E., Ed., Van Dyke, J., and A. Spitzer, "Basic Network Media Services with SIP", RFC 4240, DOI 10.17487/RFC4240, December 2005, <https://www.rfc-editor.org/info/rfc4240>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <https://www.rfc-editor.org/info/rfc4566>.

[RFC5039] Rosenberg, J. and C. Jennings, "The Session Initiation Protocol (SIP) and Spam", RFC 5039, DOI 10.17487/RFC5039, January 2008, <https://www.rfc-editor.org/info/rfc5039>.

[RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, DOI 10.17487/RFC6350, August 2011, <https://www.rfc-editor.org/info/rfc6350>.

[RFC7092] Kaplan, H. and V. Pascual, "A Taxonomy of Session Initiation Protocol (SIP) Back-to-Back User Agents", RFC 7092, DOI 10.17487/RFC7092, December 2013, <https://www.rfc-editor.org/info/rfc7092>.

[RFC7340] Peterson, J., Schulzrinne, H., and H. Tschofenig, "Secure Telephone Identity Problem Statement and Requirements", RFC 7340, DOI 10.17487/RFC7340, September 2014, <https://www.rfc-editor.org/info/rfc7340>.

[RFC8197] Schulzrinne, H., "A SIP Response Code for Unwanted Calls", RFC 8197, DOI 10.17487/RFC8197, July 2017, <https://www.rfc-editor.org/info/rfc8197>.

[RFC8224] Peterson, J., Jennings, C., Rescorla, E., and C. Wendt, "Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 8224, DOI 10.17487/RFC8224, February 2018, <https://www.rfc-editor.org/info/rfc8224>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[SHAKEN] Alliance for Telecommunications Industry Solutions (ATIS) and the SIP Forum, "Signature-based Handling of Asserted information using toKENs (SHAKEN)", ATIS 1000074, 1 2017, <https://www.sipforum.org/download/sip-forum-twg-10-signature-based-handling-of-asserted-information-using-tokens-shaken-pdf/?wpdmdl=2813>.

[SR-2275] Telcordia, "Bellcore Notes on the Networks", Telcordia SR-2275, October 2000.

Authors' Addresses

Eric W. Burger
Georgetown University
37th & O St, NW
Washington, DC 20057
USA

Email: eburger@standardstrack.com

Bhavik Nagda
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139
USA

Email: nagdab@gmail.com