Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 2, 2018

J. Rosenberg
C. Jennings
Cisco Systems
March 1, 2018

Bootstrapping STIR Deployments with Self-Signed Certs and Callbacks
draft-rosenberg-stir-callback-00

Abstract

Robocalling has become an increasing problem in the Public Switched Telephone Network (PSTN). A partial remedy for it is the provision of an authenticated caller ID in the PSTN, which today is lacking. Secure Telephone Identity Revisited (STIR) provides this through the usage of signed payloads in Session Initiation Protocol (SIP) calls. However, STIR deployment requires a global certificate system which allows for worldwide issuance of certifications that attest to which numbers a provider is responsible for. Such a system is likely to take years to rollout. To accelerate STIR deployment, this draft proposes a technique wherein STIR can be used without certificates that attest to number ownership. This is done through a combination of self-signed certificates, reverse callbacks and cached validations.

Table of Contents

1. Problem Statement

Robocalling has become an increasing problem in the Public Switched Telephone Network (PSTN). Efforts to prevent it - such as the do-not-call list - have so far proven ineffective. Recently, robocallers have gotten even more crafty, and are tailoring the caller ID of incoming calls to match the area codes and exchanges of

the recipients in order to increase the likelihood that targets pick up the phone.

Part of the reason robocalling is possible is that the PSTN doesn't provide a way to authenticate caller ID. This problem has gotten worse through the deployment of the Session Initiation Protocol (SIP) [RFC3261] along with widespread availability of APIs (as an example, Twilio), which allow third parties to easily, at low cost, place calls with desired caller IDs to anywhere in the world.

To remedy this, the Secure Telephony Identity (STIR) working group has undertaken to provide a way for e2e authenticated caller ID in SIP-based networks [RFC8224] [RFC8225] [RFC8226]. The core concept is to enable a signature over the SIP INVITE, the signature covering key SIP fields including the From header field containing the caller ID. The signature uses a certificate which is signed by an entity to whom the target has a trust chain, and more importantly, the certificate claims as part of its structure, the phone numbers that the calling party is permitted to claim.

The primary challenge to deployment of STIR is the certification process. It requires a global certification system which can issue certificates to providers across the world, and furthermore, has the processes and database accesses required to assert the set of phone numbers owned by any carrier using the system. This is likely to require coordination amongst telcos, governments, regulators, and telco providers across the globe. Its scope of complexity is similar to ENUM [RFC2916] , which required a similar global infrastructure. ENUM was never successfully deployed.

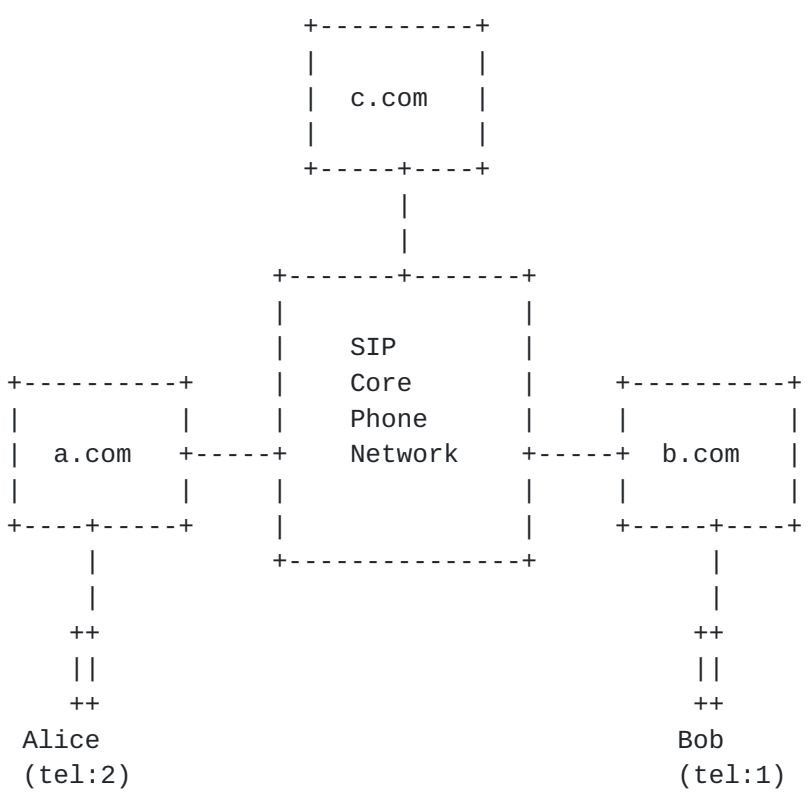
This document proposes a way to accelerate STIR deployments by relaxing the need for any such certification authority. It works with traditional self-signed certificates, and requires only that the calling domain and receiving domain support the protocol defined in this specification. This makes it much easier to deploy. If and when certificates with number ownership are deployed, they can easily co-exist with this proposal, phasing it out over time.

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. Overview of Operation

Consider the following reference architecture:



Alice and Bob are telephone subscribers with phone numbers 2 and 1 respectively, using service providers a.com and b.com respectively. These two providers are connected to each other over a SIP network, which provides routing of calls between providers. A key assumption in this proposal is that this core network accurately routes calls to a specific number in a way which attackers cannot circumvent easily. It also assumes that sufficient portions of this core phone network are now SIP based, enabling delivery of SIP extension values between the originating and terminating providers. This second constraint is identical to in-band STIR. Note however that this proposal does not require SIP to the endpoints; it only assumes SIP between the originating and terminating call agents. While those agents could be SIP proxies or B2BUA, they could also be traditional circuit switched agents with SIP interfaces. We refer to this generically as a call agent.

Alice places a call to Bob's telephone number. It arrives at Alice's agent - the calling agent. The calling agent has a self-signed certificate (the solution also works with traditional domain based

certificates). Alice's agent uses this certificate to sign the INVITE as specified in [RFC8224] and [RFC8225]. The INVITE includes a Supported header field with the value stir-callback.

This passes through the core SIP network, which ultimately delivers the call to the receiving agent based on traditional SIP routing logic.

When the call arrives at Bob's agent, it verifies the signature per [RFC8224]. Bob's agent maintains a cache, called the validation cache, which is a mapping from caller IDs to public keys. When the call arrives, Bob checks whether the caller ID matches an entry in the cache. If there is no match - which is the case for the first call from this caller ID - Bob's agent performs a verifying callback to check the validity of the caller ID.

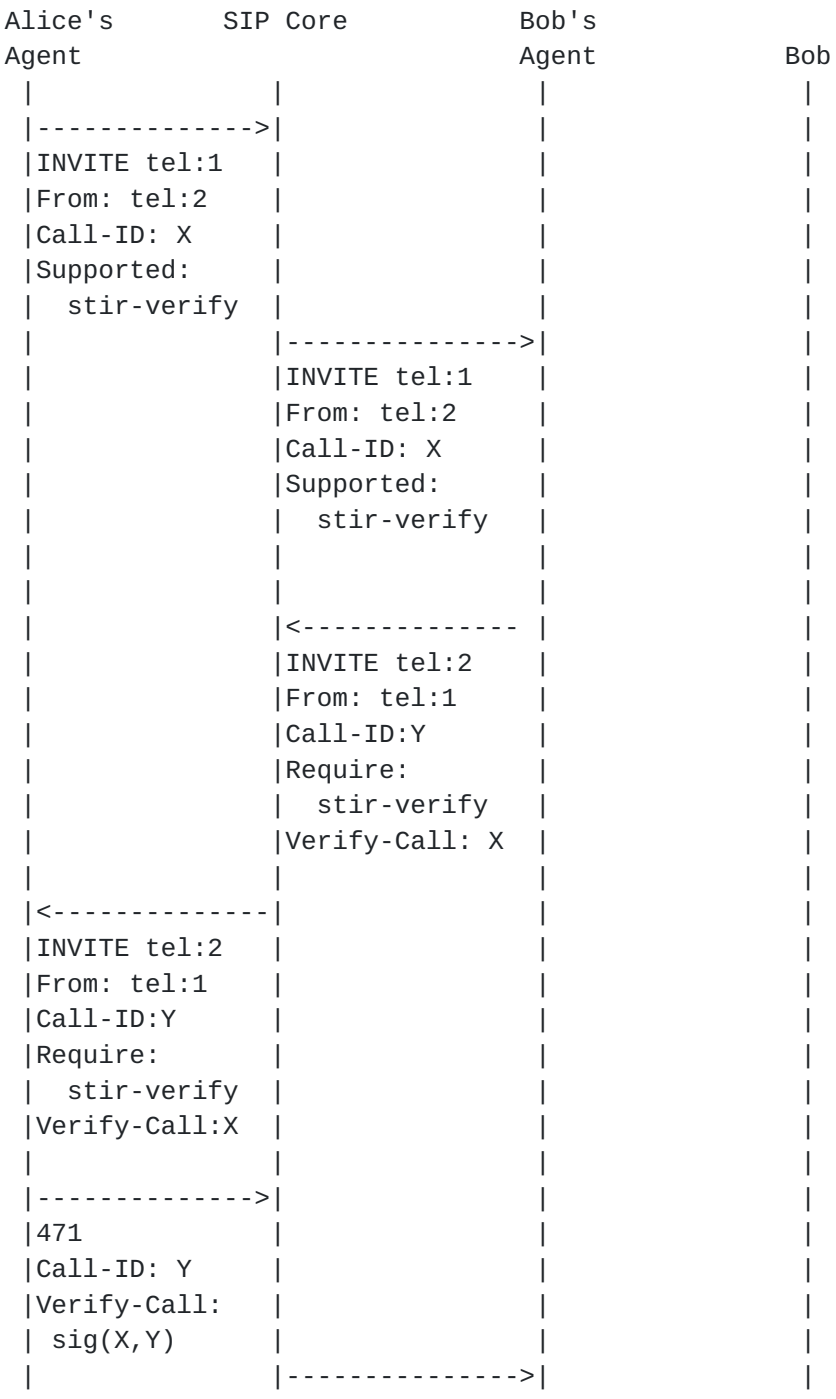
To perform this callback, Bob's agent holds onto the incoming INVITE from Alice, and generates a completely separate INVITE, targeted back towards the number from the incoming caller ID. The verifying INVITE includes a Require header field with the value stir-callback. It also includes SDP, though the contents of this SDP are not relevant as they will never be used. The verifying INVITE also includes the Verify-Call header field. This header field is populated with value taken from the Identity header field of the incoming INVITE from Alice.

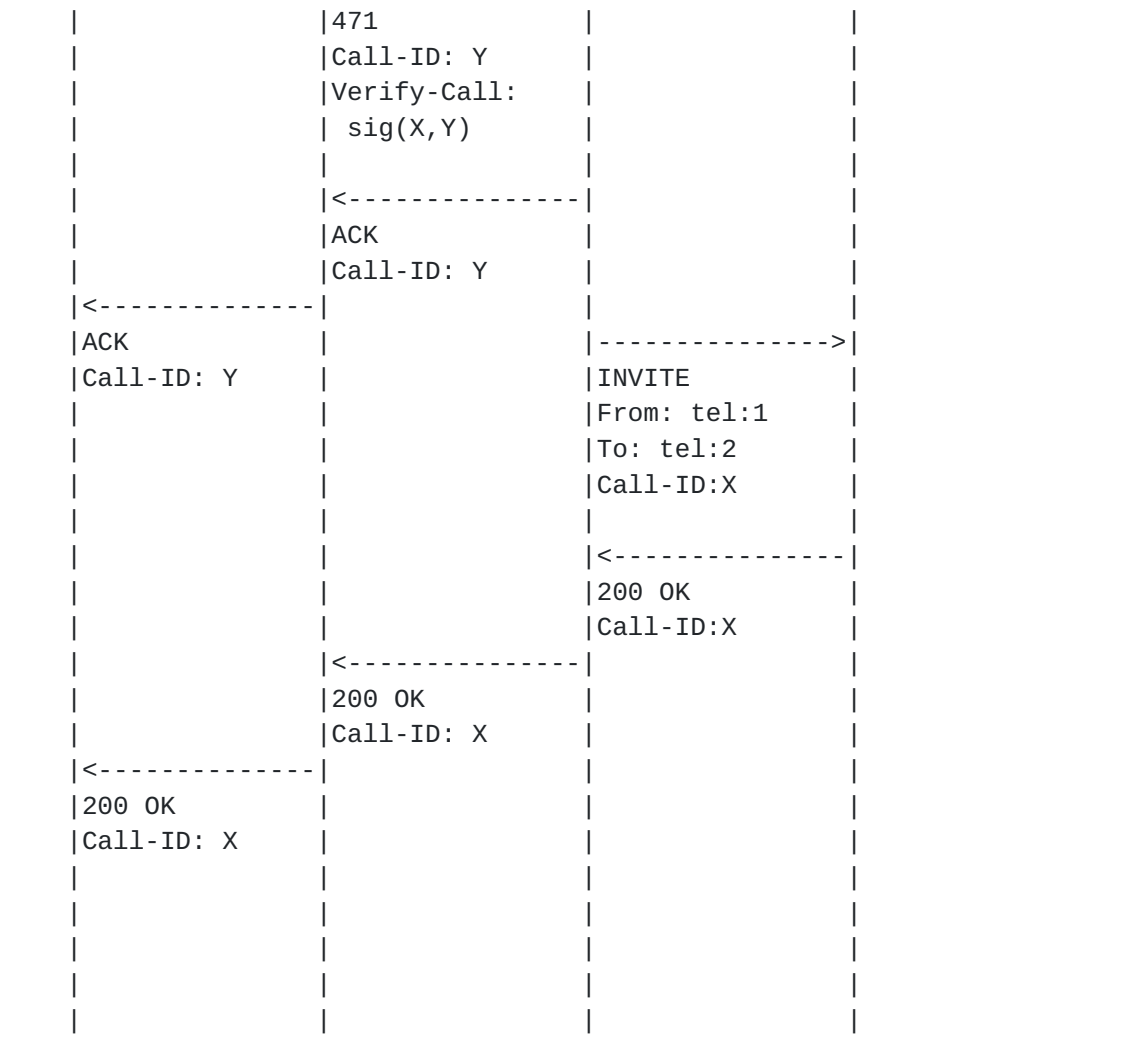
The SIP core network will route the verifying INVITE towards the agent which owns Alice's number. There are three possible cases to consider.

1. The CallerID was correct. In this case, the verifying INVITE will return to one of Alice's call agents. The agent sees the presence of the Require: stir-callback header field. This tells the agent that this is not actually a real call to be completed towards Alice, but rather, a verifying callback to check that Alice's agent really meant to place the original call. As such, Alice's agent extracts the certifcate and signature values from the Verify-Call header field, and checks if they reprsent a valid certificate for signatures from Alice. If it is correct, Alice's agent rejects the INVITE with a 471 response code. This is a new response code which means the call itself should not proceed, but the receiving agent recognizes the the information in the Verify-Call header field as valid. Alice's agent creates a signature over the Call-ID in the incoming INVITE as well as the value in the Verify-Call header field, and includes this signature in the response, in the Verify-Call header field. When this error code reaches Bob's agent, Bob's agent verifies the signature using the public key from the inbound INVITE. Once this has verified,

Bob's agent knows that the caller-ID in the original INVITE was valid. Bob's agent adds the caller-ID to its cache of validated numbers and associates it with the public key from the certificate. Any future calls with this certificate and caller ID from that source will be trusted and not require the verifying callback.

The sequence diagram for this case:





1. Alice's agent presented a false caller ID, and the agent which owns that false caller ID supports this extension. The verifying INVITE will route through the SIP core but arrive at a different agent, that of c.com. That agent supports the stir-verify option tag. However, when goes to validate the values from the Verify-Call header field, it will fail. In that case, it rejects the INVITE with a 472 response code. This is another new response code, which means the call itself should not proceed, and furthermore, the receiving agent did not recognize the information in the Verify-Call header field as valid. When Bob's agent receives this, it rejects the incoming INVITE with a 472 as well, informing Alice's agent that it rejected the call due to an invalid caller ID.

2. Alice's agent presented a false caller ID, and the agent which owns that false caller ID does not support this extension. When the verifying INVITE arrives at c.com's agent, it will reject the INVITE as normal with a 420 response code due to the presence of

the unsupported Require option tag. This is routed back to Bob's agent. The receipt of a 420 could signify a malicious caller ID, but could also indicate that there was an intermediate PSTN gateway in the SIP core, in which case the caller ID could be authentic. In this case, Bob's agent MAY complete the call towards the caller.

Each agent builds its own cache of validated certificates for caller ID values. These caches do not need to be shared between providers; they are purely localized to a single administrative entity. The cache entries are invalidated based on the lifetime of the certificate, or through the receipt of an incoming INVITE whose caller ID matches a cache entry, but with a different public key in the certificate. This can happen legitimately due to a number port. In such a case, the receiving agent removes the cache entry and re-performs the validation callback.

Open Issue: Should a new public key invalidate previos ones or should multiple public keys for same caller ID be allowed.

The design proposed here uses an INVITE in the reverse direction, rather than an OPTIONS request or another extension, to maximize the probability that the verifying call actually traverses the SIP core. The significant number of SBCs and other entities which are not likely to pass OPTIONS or non-INVITE requests makes this the best approach for success. It also ensures that the same policy that would be use to route a real call, routes the verifying call.

The presence of the Require header field in the verifying INVITE is critical to the operation of the solution. It prevents the verifying INVITE from actually ringing a real phone, which would be quite annoying.

4. Interactions with RFC 8226

This mechanism provides a technique for deploying STIR prior to the availability of RFC 8226 certificates. It also works nicely in conjunction with incremental deployment of RFC 8226.

In the case where an originating agent supports both this specification and RFC 8226, it would use the RFC 8226 certificates which cryptographically assure its ownership of the number in the From header field. When this is received at the terminating agent, if that agent supports both RFC 8226 and this specification, it first checks for the presence of the RFC 8226 certificate. If present and valid, it proceeds with the call and no verifying callback is required. If the certificate is RFC 8226 compliant but the number

does not match the one in the From header field, or there was no RFC 8226 certificate present, the verifying INVITE is generated.

The consequence of this co-existence is that the volume of verifying callbacks decreases as RFC 8226 is deployed, and the overall system provides verified caller ID the entire time.

5. SS7 Interactions

In reality, significant portions of the PSTN traffic between carriers remain powered by SS7 and not SIP. If that happens, the verifying INVITE might hit an SS7 gateway which is not an agent acting on behalf of Alice.

There are two subcases. In one case, the SS7 gateway does not support this extension. When that happens, the INVITE is rejected with a 420. As described above, Bob's agent will pass the call to Bob. If however the SS7 gateway does support this extension, it still rejects the request with a 420 error code. This is because the overall system - the PSTN - does not support the extension and the call cannot be passed through the PSTN.

TODO: consider specifying an SS7 gateway function and corresponding SS7 extension; this extension needs only a single bit to pass through the SS7 network, and two bits in the call rejection message. It is worth noting that SS7 extensions may be needed to pass the PASSporT information. Need to investigate if that is possible.

6. Formal Protocol Specification

This specification defines behavior for two entities - an originating agent and a terminating agent.

An entity acting as an originating or terminating agent can be a proxy or a B2BUA. However, it MUST be the registrar of record for the user on whose behalf it operates.

6.1. Originating Agent Behavior

6.1.1. On Receipt of incoming INVITE

When an originating agent is acting as an outbound proxy on behalf of the user and receives an outbound INVITE from a user (no Require header field with a value of stir-verify), it MUST include a Supported header field in the INVITE with a value of stir-verify. It MUST add an entry to a table, the pending transactions table.

Furthermore, the originating agent MUST follow the procedures defined in [RFC8224] and [RFC8225] to compute a passport and create a signature over it. It MAY utilize either a self-signed certificate or a traditional domain based certificate.

6.1.2. On Receipt of a Verifying INVITE

When an originating agent receives an INVITE with a Require header field containing the value stir-verify, it MUST examine the INVITE for the presence of a Verify-Call header field. If this header field is not present, the originating agent MUST reject the INVITE with a 400 error code. If the header field is present, the agent extracts the value there, and checks that it represets a valid PASSporT signature using any self singned certificates for the caller ID.

If it is valid, it MUST reject the incoming INVITE with a response code of 471. If it is not valid, it MUST reject the incoming INVITE with a 472 response code.

A response with a 471 response code MUST contain a signature, placed into the Verify-Call header field in the response. This signature is computed by taking the caller ID from the incoming INVITE, concatenating it with the value present in the Verify-Call header field, and then using that as an input to the signature function. TODO: provide detailed spec on signature function.

Open Issue: is this signature in 471 needed?

6.2. Terminating Agent Behavior

6.2.1. On Receipt of Incoming INVITE

When a terminating agent receives an incoming request for a user on whose behalf it operates, it checks for the existence of the Supported header field with a value of stir-verify. If not present, the agent SHOULD pass the call to the targeted user. If present, the agent behaves as follows.

The agent SHOULD maintain a validation cache. This cache is indexed by E.164 number, and contains as a value the public key of the certificate for the agent that was validated as being authoritative for that number.

The agent extracts the number from the From header field of the incoming INVITE. It performs the validation processing defined in [RFC8224] to verify the signature. Once validated, it checks the value of the From header field against the cache.

If there is a matching cache entry, and the public key in the cache entry matches that of the certificate, the agent SHOULD forward the original INVITE towards the called party.

If there is a matching cache entry, but the public key in the cache entry does not match that of the certificate, the agent MUST invalidate the cache entry and proceed as if there was no match.

If there was no matching entry in the cache, the agent constructs a new INVITE header field. The Request-URI and To header field of this INVITE MUST match that of the From header field from the incoming INVITE. The From header field MUST be set to the value from the To header field in the incoming INVITE. The request MUST contain a Require header field with value stir-verify. The request MUST contain any valid SDP offer [RFC3264]. This request MUST then be sent towards the request URI in the same way it would have been sent had it been received from its own user.

The agent sets a timer, with a RECOMMENDED value of 5 seconds. This represents the maximum amount of time the agent will wait for a response to the verifying INVITE before passing the call onwards to the the target of the incoming call.

6.2.2. On Receipt of a Response to the Verifying INVITE

If the terminating agent receives a 471 response to the verifying INVITE, it MUST look for the presence of a Verify-Call header field in the response. If not present, the original INVITE is rejected with a 472, and it MUST NOT add an entry to its validation cache. The signature from this Verify-Call header field is verified, and checked to match against the public key used in the incoming INVITE. If not valid, the original INVITE is rejected with a 472, and it MUST NOT add an entry to its validation cache. If the signature is valid, It SHOULD add an entry to its validation cache. This cache is indexed by the caller ID present in the From header field of the original INVITE. Its value is the public key from the certificate in the incoming INVITE.

If the terminating agent receives a 472 response to the verifying INVITE, it MUST NOT add an entry to its validation cache. It SHOULD reject the original INVITE with a 472 error response. If the terminating agent receives a 420 response to the verifying INVITE, it MUST NOT add an entry to its validation cache. It SHOULD forward the original INVITE towards the called party.

6.2.3. On expiration of the timer

If the 5 second timer fires before a response has been received to the verifying INVITE, the agent SHOULD CANCEL the verifying INVITE. It SHOULD forward the original INVITE towards the called party.

7. Security Considerations

The primary purpose of this specification is to improve the security of caller ID in the public SIP-based phone network. We can consider three actors in the system, and examine malicious behavior from each. These actors are the caller, the callee, and the agent receiving the verifying INVITE.

7.1. Attacks from the Calling Agent

The primary attack the caller can launch is to place a call with a faked caller ID. Preventing this attack is the primary purpose of this specification. This specification prevents it under the assumption that the SIP core network provides forward routability, and therefore, the caller ID is valid if the agent that placed the call, would also receive a call placed towards that callerID. This relationship is verified with the signature over the callerID in both INVITE requests.

It is possible in this system for the calling agent to lie about the callerID, but for the fake caller ID to be associated with the number space owned by that agent. In that case, the calling agent can verify its own faked caller ID. However, since the originating agent is in purview of the usage of its own numbers, there is little that can be done to solve this attack, and in many regards it is not an attack. As an example, outbound call center calls frequently "lie" about the caller ID by placing the company main number in the callerID. Since both are owned by the same administrative entity, this is an acceptable use case.

In a different attack, the calling agent is malicious. It doesn't lie about its callerID in the outbound INVITE. However, when the verifying call arrives, the calling agent rejects it with a 472, indicating that the caller ID was faked. The only affect of this action would have is to cause the verify call placed by the calling agent to be rejected, and therefore seems to serve no purpose.

An additional consideration is whether the mechanism specified here can be used as a denial of service attack. Consider a malicious originating agent which purposefully inserts a fake caller ID, not to be delivered to the called party, but to trigger a verifying INVITE to the agent which actually owns that phone number. Indeed, based on

this specification, the terminating agent will in fact generate such an INVITE. However, since the attacker must emit a single INVITE in order to cause the terminating agent to generating a single INVITE, there is no amplification possible.

7.2. Attacks from the Called Agent

Consider the case where the called agent is malicious. The calling agent A is not malicious, and places a legitimate call with a valid caller ID (tel:2) to agent B. Agent B places a new call (not a verifying call) to a third agent, agent C, using the same Call-ID as the incoming INVITE it just received, and claims the caller ID tel:2. When agent C places a verifying call for this caller ID, tel:2, it will be routed back to agent A. In this case, because there is in fact a valid call in progress from agent A with that caller ID, the verifying call will succeed. This will cause agent C to believe that agent A legitimately owns the caller ID tel:2, and agent C now caches the certificate from agent B. Agent B is now free, at will to place calls towards agent C with the fake caller ID.

This is prevented through the usage of the signatures in the 471 response codes. In this attack, the signature used by A to sign the response will use its own public certificate. This will not match the one used in the inbound INVITE from B to C which triggered the verifying call. Therefore, B will reject the incoming INVITE and will not update its validation cache.

7.3. Attacks from the agent receiving the Verifying INVITE

In the case where the caller is malicious, and so is the agent receiving the verifying INVITE, it is possible (even without collusion) that the agent receiving the verifying INVITE responds with a 471 to the verifying INVITE, even though it doesn't actually own the number in question. It might do this in an attempt to pollute the cache of the called agent with an invalid entry.

This is prevented through the usage of signatures in the 471 response. Since the agent receiving the verifying INVITE is not the same as the calling agent, and there is no collusion in which private keys are shared, the signature in the 471 will not match that of the incoming INVITE. This will cause the incoming INVITE to be rejected, and no valid cache entry is added.

8. IANA Considerations

This specification registers a new SIP option code and two new response codes.

8.1. sip-verify Option Tag

This section registers a new SIP option-tag, sip-verify. The required information for this registration, as specified in RFC 3261, is:

Name: sip-verify

Description: This option code indicates support for verification of caller ID using a verifying INVITE. When present in a Supported header field, if informs the recipient that it can, and should, generate a verifying INVITE to confirm the caller ID. When present in a Require header field, it tells the receiving agent that the purpose of the INVITE is to validate that a prior call had been placed, and that the INVITE should not actually be passed to the target of the INVITE.

8.2. Response Code 471

This section registers a new SIP response code, 471. The required information for this registration, as specified in RFC 3261, is:

RFC Number: NOTE TO RFC-EDITOR: replace with the RFC number of this specification.

Response Code Number: 471

Default Reason Phrase: Caller ID Verified

8.3. Response Code 472

This section registers a new SIP response code, 472. The required information for this registration, as specified in RFC 3261, is:

RFC Number: NOTE TO RFC-EDITOR: replace with the RFC number of this specification.

Response Code Number: 472

Default Reason Phrase: Caller ID Not Verified

8.4. Verify-Call Header

TODO

9. Acknowledgments

Thanks for Richard Barnes for identifying the attacks described in the Security Considerations section.

10. References

10.1. 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>.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <https://www.rfc-editor.org/info/rfc3264>.

[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>.

[RFC8225] Wendt, C. and J. Peterson, "PASSporT: Personal Assertion Token", RFC 8225, DOI 10.17487/RFC8225, February 2018, <https://www.rfc-editor.org/info/rfc8225>.

10.2. Informative References

[RFC2916] Faltstrom, P., "E.164 number and DNS", RFC 2916, DOI 10.17487/RFC2916, September 2000, <https://www.rfc-editor.org/info/rfc2916>.

[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>.

[RFC8226] Peterson, J. and S. Turner, "Secure Telephone Identity Credentials: Certificates", RFC 8226, DOI 10.17487/RFC8226, February 2018, <https://www.rfc-editor.org/info/rfc8226>.

Authors' Addresses

Jonathan Rosenberg
Cisco Systems

Email: jdrosen@jdrosen.net

Cullen Jennings
Cisco Systems

Email: fluffy@iii.ca