

Comments on draft-ietf-simple-message-sessions-00

This is a big improvement over the previous draft. It is much easier to understand and a some of the key issues are nicely resolved. I found it way easier to understand and think about. I like it.

The comments below are a bit of a mess but given the meeting this week I wanted to get something out ASAP. Thanks for getting the draft out well before the meeting.

1. Authentication to relays during VISIT

Problem: Right now the security of the system is based on an attacker not knowing the URL the bind returned. An attacker does not need to guess the URL, they just have to wait until they see it and then kill the original request and substitute their own VISIT.

Solution: The URL returned in the BIND contains a password (in the password portion of the URL). This is passed in the SDP. The client that is going to do the VISIT, removes the password from the URL and does the VISIT with the stripped. If the client is Digest challenged by a relay that has set the realm to the same thing as the host port in of the URI, then the client uses this password to compute the response to the challenge.

In the case that all the SIP and MSRP were sent unsecured over TCP this won't help much but in the secure case with more than one relay where TLS was used for the hop by hop and the SDP was encrypted end to end using S/MIME, this is absolutely required for the hosting relay to authenticate the client that is doing the VISIT.

I think it is a really bad idea to stay with the current "hard to guess URI scheme" given how easy it is to fix but ... If we do stay with this scheme, Section 5.3 should say that there need to be at least 32 bits of crypto random (not pseudo random) numbers. It is might turn out to be hard to generate this much randomness on the servers due to the large number of BIND messages per second it supports. If this is the case then one should switch to generating the random part of the URI on the endpoint and sending it up to the relay which can verify uniqueness and send back.

2. Support for more than two relays

First let me introduce some terminology that can help make the one and two relay case easier to describe. It does not change how they work from what is in the current draft. I will refer to the "hosting relay" as the relay that the URI passed in the SDP resolves to. The other relay is a "non hosting relay" for the session.

The first thing a relay should do when it receives a message is to check if it is relay referred to in the host portion of the URI. If it is does, then it should behave as the draft states. If the URI does not match the host of this relay, then the relay should forward the message to the host in the URI. When a request is forwarded like this, the relay becomes a non hosting relay for this session. Two TCP or TLS connections are still

formed for the session and the relay. This is pretty much what the current draft does in the case of VISIT requests.

There are lots of arguments and reasons about why there will never be more than two relays – personally I find them somewhat similar to the arguments about why SIP phones will never have outbound proxies. In the end, it's hard to really know if there will be cases with more than two relays or not so I'm proposing a six word change to the draft. It will not make any difference to the complexity of endpoints. It will not make any difference to the complexity of relays that choose not to support more than two relays, it will not make any difference to the protocol on the wire. The change is this – we add a few words on to the end of the sentence “If the URI does not match the host of this relay, then the relay should forward the message to the host in the URI” so that it says “If the URI does not match the host of this relay, then the relay should forward the message to the host in the URI or another host specified in provisioning.”

If it turns out only two relays are needed, this change did not complicate anything but on the other hand it leaves considerable flexibility for a variety of reasons. I will point out that a relay that S/MIME signs and encrypts all of your messages on your behalf will be one of the nice things that might come out a having more than two relays. Many of the things you get will not be nice from the end users point of view. Keep in mind when constructing arguments about why we don't need relays that the information transferred via IM is much closer to the type of information transferred over http than the audio or video content transferred over RTP.

3. Less than one relay.

How does an endpoint know that a relay is not needed? Need some way of addressing this. Configuration is one way. Using STUN to find out if your address is globally routable would be another. Another possibility is an ICE like mechanism where the offer could contain two address with one being preferred and the far end tries both to select one that works. I don't care too much but I think this needs to be described a bit more.

4. Encryption of bodies of SEND messages

Section 9.3 just need more details and clarification. Big picture looks right on but will need more than this to be able to implement it. Who picks the session keying material? The offer could pick some then allow the answer to add some more. I'd be happy to write some text

5. Scalability of the Solution

I think the choices we are making on scalability are OK but it seems well worth pointing out the limitation we are accepting and why the alternative was not chosen. The alternative was not chosen because we would need to develop a transport that worked for multiplexing (TCP did not) and allowed per multiplexed channel flow control (SCTP did not). Any comments on BEEP?

It is probably worth doing some calculations on size of system and what the bottle neck limiting relays is going to be. Of course it would be nice if the bottle neck was bandwidth to the relay. This solution is going to blow chunks on the most common off the self

operating systems and is going to be fine for folks that can build systems with TCP implementations that can support many more than 65k TCP connections on a single IP address. Some of the high end web stuff has clearly demonstrated that this is possible. Apache and IIS have clearly demonstrated the limits of common OS for handling lots of TCP connections – If you could do more – Apache and IIS would.

Which gets me to my second point – Given you form a new TLS connection in the BIND or VISIT for every new IM session, everybody better implement session-resumption and have really big caches or public key accelerator hardware in the relays.

The more I think about traffic between two large IM providers, the more I suspect that a proxy doing page-mode SIP messages with TLS connections will be able to support more users than a MSRP relay doing TLS.

I don't know if the stuff above is right or wrong. But think about it. We have said that page-mode SIMPLE is not scaleable – we better make a compelling argument that MSRP is scaleable or everyone will just go do page-mode.

I'm not asking for any change to MSRP in regards to this topic but I do think the draft should have some discussion of if this protocol does meet the scalability requirements that caused a new protocol to be invented. Stating those requirements would make this easier.

6. Head of Line blocking

The problems is much reduced from the previous draft. There is still a little bit of a problem from the fact that both sides send message from A to B and B to A multiplexed over the same connection. Consider a case with no relays where A is sending the 5G file, and B wants to send an IM to A saying “dude stop, this is going to cost me a fortune over GPRS”. There is no way for B to get the 200 saying A received the message.

I see two possibilities – ignore this problem, or open separate connections for A to B and B to A. The second solution only reduces the scalability of the relays by a factor of 2 but given the other choices made about scalability and lack of requirements or concern about it I can't imagine how a constant factor of two could be an issue.

If I had to vote, I'd probably vote for the just ignore the problem solution.

7. The direction negotiation stuff

Make it clear what endpoint MUST implement. I think I am probably in favor of the direction stuff but I have two big problems with it:

- 1) If passive mode is allowed, you can send trivial SIP UDP messages to a ton of clients that then do a DOS attack on the relay the clients use as their relay. This problem may not be worth worrying about.
- 2) If yahoo and AOL both deploy systems that only do passive for outbound and active for inbound – they can both claim they are fully standards compliant yet their users won't be able to talk to each other. This is not their fault, it is the fault of the standard that allowed such a broken design. I feel strongly that the protocol should be written in a way that two compliant endpoints can communicate – I'm fine if their operators configure them in such a way that does not allow this.

We need to make it clear in the text that endpoints **MUST** implement active and that endpoints **MAY** refuse an offer passive because of DDOS concerns.

8. Security Required?

Was not clear to me if TLS was a **MUST** implement or not. I think it should be. The type of clients were it would be difficult to implement TLS will all use page mode anyways. What TLS crypt suites are required?

9. The Digest Auth

The way this is currently written is somewhat broken. There is nothing that ties the digest to the rest of the message so any cut and paste would work fine. To fix this include the S-URL and TR-ID in the hash. I would also add a realm to CAAuth header, make it clear that a message can contain more than one CAAuth header (and Schal). I'd be happy to provide new text for this section if you want.

Need comment from crypto folks on sha1 vs. md5.

10. Message Framing

The message transported by this protocol are highly likely to be encrypted or signed. Because of this they will be inside of some sort of multipart mime body. Any endpoint sending and receiving these is going to have to deal with all the issues of using unique boundaries to frame messages. There is no way to avoid doing this. It is trivial to do this at wire speed. It is not complicated to implement.

I am proposing that we add the following text. "If the MSRP body contains a multipart mime body, the length of the message may be set to 0." The advantage of this is that we can send messages where the length of the messages is not known when the endpoint starts sending it. This makes very little difference to the complexity of the code on the endpoints. It makes the relay slightly more complicated. Given the speed that boundaries can be scanned, it is very unlikely to reduce the capacity of a relay.

This will allow clients to start sending a message before they have generated the complete message. This has turned out to be very useful in many web applications. It is so simple for us to get this right now and so hard to fix later.

11. Relay Discovery

Did not see a mention of how to find a relay to send the BIND to. I am fine with configuration being the only way to do this but a SRV lookup of the service `_msrp` in the default domain also seem reasonable.

12. Syntax or URL

Change the MSRP URL from the form
msrtp:host.example.com:1234/asfd34
to
Msrp:asfd34@host.example.com:1234

This reduces the ambiguity about if /asf34/123 somehow matches /asf34/* and it also make the syntax nicer when a password is included in the URL. I understand this is just syntax but this syntax is a better mapping of this information content into how URL are generally thought of.

13. *Limit the size of the message*

(Section 6.4) Make it clear that length must be less than 2^{64} . If we don't clarify this some implementation might put a limit at 2^{32} which will cause no end of interoperability issues.

14. *Default Port Number*

Do an IANA registration of a port number and recommend that relays use this port. This is only a configuration recommendation and does not change the protocol. The ports must be specified in all places it is currently specified in the protocol. Lack of a port in a MSRP URI means you must do a SRV lookup to find the port and does not mean the port is assumed to be the IANA registered port number.

The reason for requiring the port in the messages is that it makes the code simpler and has less interoperability bugs. The reason for recommending a common port is to make it easier for firewall configuration and network monitoring tools as well as setup in situations where DNS is changeable.

15. *URI Tags*

Do we need a transport=tls tag in the URI? Specifically, how are we going to upgrade to both TCP and SCTP either of which could use TLS or not.

16. *SRV for MSRPS*

Needs to do _tls instead of _tcp when the URI scheme is MSRPS. Several places this needs a bit of updating.

17. *TLS hop by hop VS proxy*

I see a ton of problems with trying the proxy approach. Hop by hop seems fine for what we want here. Keep in mind we want a way for the client to authenticate the proxy and TLS provides that.

18. *Renegotiation after failure.*

I like the stuff on closing connection. Consider the idea of adding "The endpoint that was hosting the previous session MAY try and negotiate a new session with a new SDP offer/answer but the other endpoint SHOULD NOT try to create a new session."

19. *Timers*

Are there any timers lurking about and if so can we define non normative recommendation on their values. Recommendation on soft state times?

20. Firewall Security Issue

I'm not sure the firewall people would call this "firewall friendly" - Add some text to the security section along the lines of Some systems try to use firewalls to limit who users inside the firewall can communicate with. Relay systems such as this render this useless. People allowing MSRP through their firewalls should be aware the difficulties of controlling where the relay is connecting too.

21. CPIM Compatibility

Section 9.4 might be all correct but I did not understand it enough to go implement it. An example and some more text might be good.

22. Trivial NITS

Section 1. I would use "page-mode" instead of "pager-mode"

Section 2. The efficiency argument is pretty bogus – in most ways it is similar to SIP over TCP. Sig-comp would eliminate the compression argument. I do buy into the session security efficiency argument. The best argument is treating this like other media streams – emphasize that and not the bogus stuff.

Many places TCP is used when really I think this document needs to be defined across TCP and TLS. (Yes – I know TLS runs on TCP :-)-

Section 4. It is trivial for us to change names now. The Unix socket operation of BIND is not the right analogy for what is going on here – LISTEN would be a better name.

Section 5.1 – what is the protocol field when TLS is being used

Section 5.1 – Do we really need the "*" or could we just always have that be the semantics. So you can always send anything you want and get a 415 if it does not work.

Section 5.6 In example make clear that the port in the m lines is not used by setting it to something like 666. Add text that says even though these aren't used, they SHOULD be set to the same value as the URI. This will aid tools that process SDP for debugging and network monitoring reasons. So they are write only fields – no one should read them but we will define what gets written into them anyways.

Section 6.2 – how are tags in URI compared. We don't have any yet but might want to define in case a later version do.

Section 6.3 – make it clear that v4 and v6 IP address formats MUST be supported.

Section 6.4 – the BNF defines Reason as a token. I don't think this is what you want.

BNF - Change "S-URL" to just "URL" and "Cauth" to "Auth" – at this point the first leader of the header will uniquely identify the header. This is good.

Make clear that Cauth and Auth can occur more than once while others can't.

23. Updates

If you would like, I would be happy to provide new text for all the bits of the draft that involve security. Actually I would be glad to take a pass at editing the source with

tracking turned on in word then you can accept or reject my proposed changes as you see fit.

Cullen