IETF Security Architecture Update

IETF 84

Eric Rescorla

ekr@rtfm.com

Overview

- Draft update
- Identity origin indication
- Consent freshness and ICE

Reviews

- Three reviews from Dan Druta, Richard Ejzack, Martin Thomson (on -01/-02)
  - Your name here?
  - Thanks!
- A few substantive issues and a lot of editorial ones
- Believe I have folded in all their comments

Changes since -02

- Forbid persistent HTTP permissions.
- Clarified that IP Location Privacy is intended as protection against the other sides, not the site (S 5.4)
- Fold in the IETF portion of `draft-rescorla-rtcweb-generic-idp`
- Retarget the continuing consent section to assume Binding Requests (more on this shortly)

IP Location Privacy

"A side effect of the default ICE behavior is that the peer learns one's IP address, which leaks large amounts of location information, especially for mobile devices. This has negative privacy consequences in some circumstances. The following two API requirements in this section are intended to mitigate this issue: issue. Note that these requirements are NOT intended to protect the user's IP address from a malicious site. In general, the site will learn at least a user's server reflexive address from any HTTP transaction. Rather, these requirements are intended to allow a site to cooperate with the user to hide the user's IP address from the other side of the call. Hiding the user's IP address from the server requires some sort of explicit privacy preserving mechanism on the client (e.g., Torbutton [https://www.torproject.org/torbutton/]) and is out of scope for this specification." (S 5.4)

Identity Origin Detection

- Issue raised in discussions with W3C people
- A correct IdP identity assertion needs to be bound to the PeerConnection
  - Otherwise malicious JS could instantiate the IdP proxy and get their own assertion
- Currently this requirement is levied on the IdP
  - `postMessage()` messages must come from an origin starting with `rtcweb://`
  - Which can't be reproduced by ordinary content
- This works but is not very flexible
  - And is a pain for the IdP

Proposed change

- Move verification from IdP to the `PeerConnection` (RP)
- Require assertions to carry an `requesting_origin` field
  - Contains the origin of the `postMessage()` message
  - In this case would be `rtcweb://peerconnection` [TBD]
- This would be passed to the receiving PeerConnection by the IdP
- Checked by the receiving `PeerConnection`
  - Which would enforce the right value

Example

```
{
  "type":"SUCCESS",
  "id":1,
  "message": {
    "idp":{
      "domain": "example.org"
      "protocol": "bogus"
    },
    "requesting_origin" : "rtcweb://peerconnection", // NEW
    "assertion": "..." // opaque
  }
}
```

Communications Consent Overview

- Consensus to use ICE for initial consent
  - Sufficient for prevention of cross-protocol attack
  - Not so great protection against packet-based DoS
- Open issues
  - Binding Request versus new STUN method for continuing consent
  - Timer settings
  - Should we limit sender rate?
  - What about simulated forking?

Consent Freshness

- As specified, ICE only checks at start of session
- Keepalives just keep the NAT binding open
  - But aren't confirmed
  - Or authenticated
- What if I no longer wish to receive traffic?
  - General agreement, some sort of keepalive
  - Check every X seconds
  - If I don't receive a keepalive after Y seconds must stop transmitting
    - Can re-start ICE if needed

What method to use?

- `draft-muthu-01` Defines a new ICE method
  - Simple request/response
  - No username/password or message integrity
- Why not just use STUN binding Request?
  - Binding requests require integrity checks

  "One of the reasons for ICE choosing STUN Binding indications for keepalives is because Binding indication allows integrity to be disabled, allowing for better performance. This is useful for large- scale endpoints, such as PSTN gateways and SBCs as described in Appendix B section B.10 of the ICE specification."

Is a MAC needed?

- ICE Binding Requests are authenticated with a MAC
  - Based on ufrag and password
- `Consent` as currently specified does not have a MAC
  - All security is from the 96-bit STUN transaction ID
  - ... must be pseudorandomly generated
  - This is plenty of security against an off-path attacker
- Without MAC, on-path attacker can simulate consent even if the victim is not responding
  - MAC requires attacker to have username and password as well
- Not clear if there is a concrete attack that requires MAC

... But

- ICE implementations already need to implement Binding Requests

  "Though Binding Indications are used for keepalives, an agent MUST be prepared to receive a connectivity check as well. If a connectivity check is received, a response is generated as discussed in [RFC5389], but there is no impact on ICE processing otherwise." [RFC 5245; Section 10]

- So Binding Requests will work better with non-RTCWEB equipment

Duration of Unwanted Traffic/Timer Settings

- Assume we have initial consent and we re-check every T_c seconds
  - On average next check will be at $T_c/2$ (worst case T_c)
- With RFC 5389 parameters, a transaction fails after 39500 ms
  - This seems awful long
- Consensus at interim to shorten these parameters
  - This is just a profile of ICE parameters

Shorter STUN timers

- RFC 5389 has configurable values Rc, Rm
- Proposal: $Rc = 5$, $Rm = 4$. Use measured RTO, minimum $200ms$
- Example: Packets transmitted at $0, 200, 600, 1400, 3000$; transaction fails at $4600ms$
- Rationale
  - If our RTT is $> 5s$, that's not going to be a very good user experience

A related problem: liveness testing

- Applications want to detect connection failure
  - This needs to happen on a shorter time scale than consent
  - How much dead air will people tolerate? ($< 5seconds$)
- Proposal: configurable minimal *received* traffic spacing
  - If no packet is received in that time, send a Binding Request
  - Application failure signaled on Binding Request failure

Why not RTCP for liveness?

- We want to do liveness testing for non-RTCWEB peers
  - Regrettably some of these do not do RTCP
- Design goal: get liveness without help of other side
  - So *just* RTCP isn't enough
  - But is counted as part of traffic spacing

Combined Consent/Liveness
Both checks done via binding requests

- Consent timer: T_c (default $= 20s$, no more than $30s$)
- Packet receipt timer: T_r (no less than $500ms$); configurable
- When either timer expires start a STUN transaction
- When a STUN transaction succeeds, re-start both timers
- When a STUN transaction fails
  - If transaction was started by T_c, stop sending, abort
  - ... else, notify application of failure, but keep sending
- T_r is also reset by receiving any packet from the other side
- This is what is in current `draft-muthu-01`

What API points do we need?

- Ability to set keepalive frequency (individually on each stream?)
- A consent transaction has failed and so I am not transmitting on stream M
- A liveness check has failed on stream M

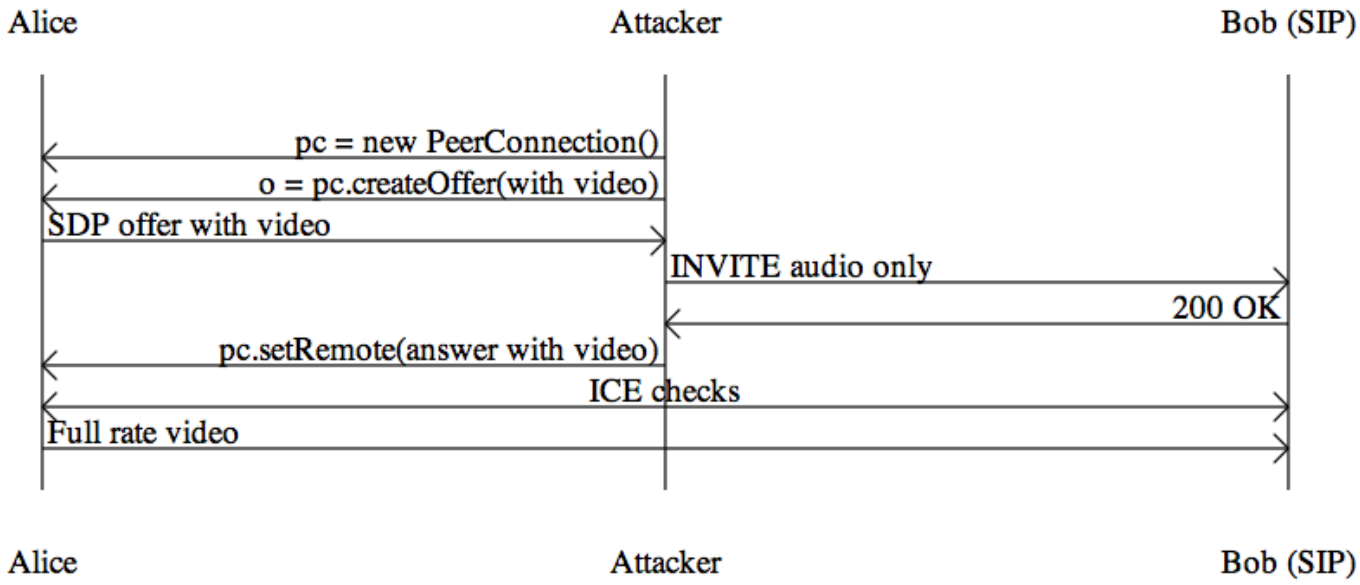
Proposed API

```
// Do a liveness check every 500ms and call callback if it fails
pc.setLivenessCheck(500, m, function(m) {
   // media stream m has apparently failed
});


//
pc.onstreamfailed = function(m) {
   // media steam consent check has failed
}
```

- Would a constraint + event combination be better?

DoS via Excessive Traffic [Thomson and others]



Why does this work?

- ICE only verifies connectivity
  - But anything can be sent over that channel
- SDP parameters are under the control of the attacker
  - And that is what controls bit rate

No user consent required (?)

- We just need a source of high bandwidth data
  - We (probably) can't use a datachannel because it's congestion controlled
  - And the sequence numbers are unpredictable (allegedly)
- But media probably is not
- It's not video that requires user consent
  - ... but access to the camera
- Set up a bogus MediaStream blob that generates continuous patterns
  - Use it to source the data
  - This shouldn't trigger consent dialogs

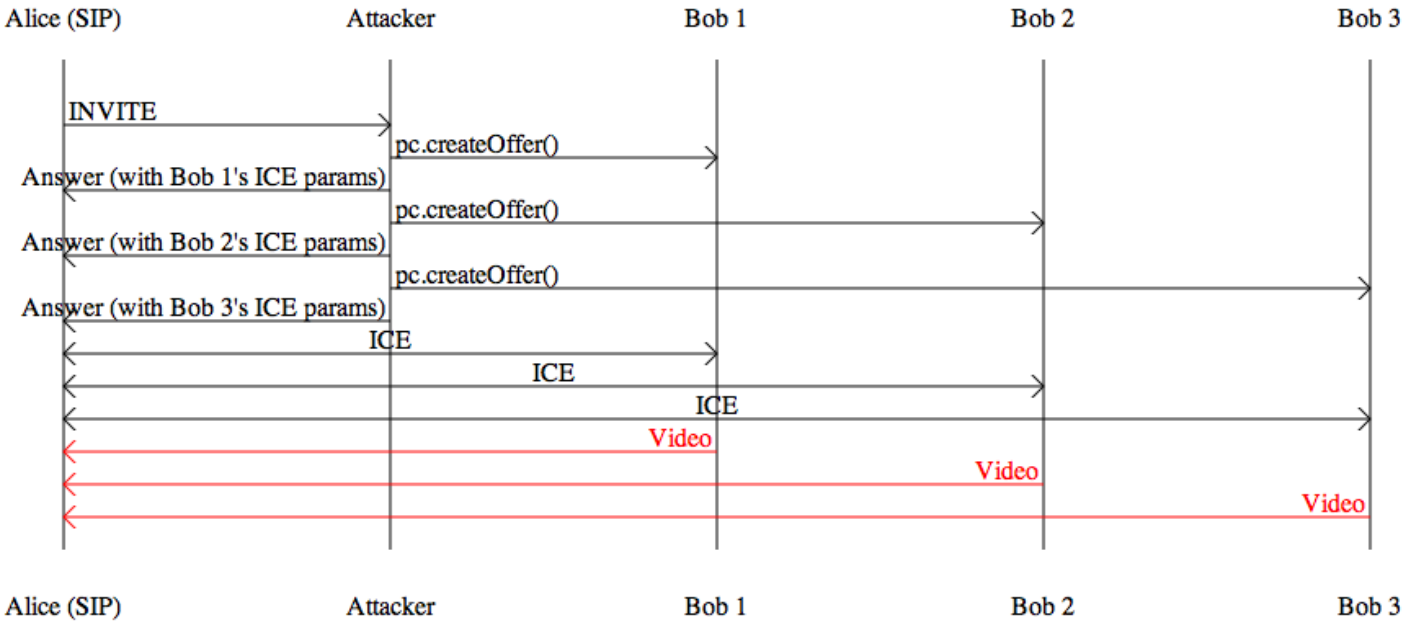
How serious is this issue?

- Basically another version of voice hammer
- Short duration
  - If we have consent freshness then < 1 minute
- But very scalable
  - I can mount this using an ad network or any other traffic fishing system
  - No user consent required
  - Not self-throttling (unlike HTTP-based attacks)

Defenses against bandwidth attacks

- Bandwidth indications in the SDP
  - This won't work
  - Attacker controls the SDP
- Bandwidth indications in ICE checks
  - This will work
  - How do we deal with non-RTCWEB peers?
    - Assume infinite bandwidth?
    - Assume finite bandwidth? If so, what?
  - Arguably this is overkill
- Proposal: Live with it between consent checks

Simulated Forking [Westerlund]



Proposed Solutions

- Rate limit the number of outstanding ICE connections you respond to?
  - Based on unique ufrag/passwords
- If using DTLS you can rate limit the number of DTLS associations
  - Not clear that this is better than ICE
- Proposal: guidance to servers on rate limiting