~~IETF 107, Vancouver~~
Virtual Interim bis
May 2020

OAUTH WG
draft-ietf-oauth-dpop

DPoP

OAuth 2.0 Demonstration of Proof-of-Possession at the Application Layer

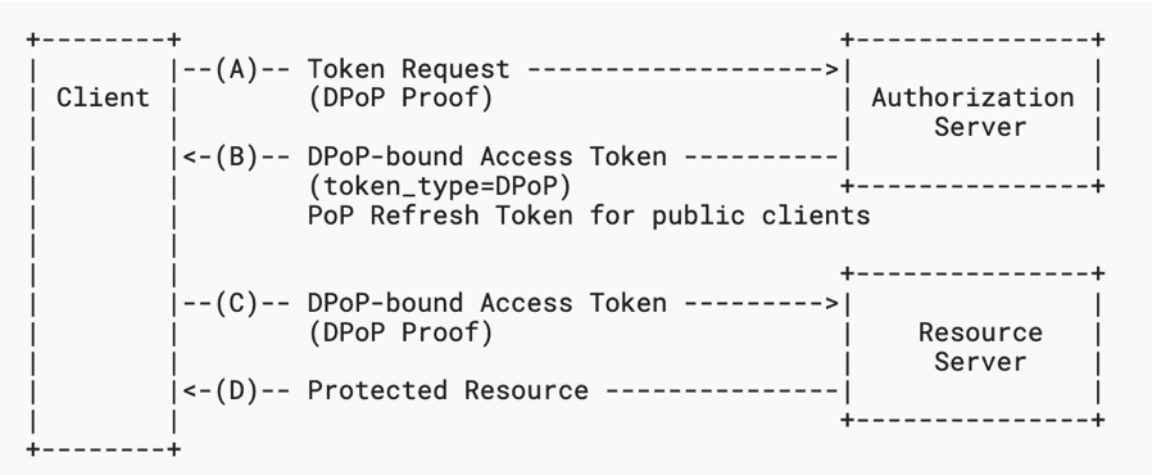
Daniel Fett
Brian Campbell
John Bradley

Torsten Lodderstedt
Michael Jones
David Waite

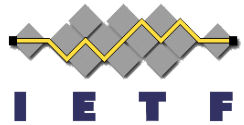
What D'Heck is DPoP?

Application-level proof-of-possession protections for access and refresh tokens

- DPoP Proof JWT sent as an HTTP header
  - Demonstrates a reasonable level of proof-of-possession in the context of the request
  - Sent the same way with the same syntax and semantics for both token requests to the AS and protected resource requests
  - AS uses the proof to bind tokens
  - RS uses the proof to verify bound tokens



Anatomy of a DPoP Proof JWT



Explicitly typed

Asymmetric signature algorithms only

The public key for which proof-of-possession is being demonstrated

Minimal info about the HTTP request

Unique identifier for replay checking

Only valid for a limited time window relative to creation time

Other stuff *could* go here

```
{
 "typ":"dpop+jwt",
 "alg":"ES256",
 "jwk":
  {
   "kty":"EC", "crv":"P-256"
    "x":"l8tFrhx-34tV3hRICRDY9zCkDlpBhF42UQUfWVAWBFs",
    "y":"9VE4jf_Ok_o64zbTTlcuNJajHmt6v9TDVrU0CdvGRDA"
  }
}
{
 "jti":"-BwC3ESc6acc2lTc",
 "htm":"POST",
 "htu":"https://server.example.com/token",
 "iat":1562262616
}
```

Access Token Request

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6Ik
 VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUklDUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR
 nMiLCJ5IjoiOVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R1JE
 QSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiItQndDM0VTYzZhY2MybFRjIiwiaHRtIj
 oiUE9TVCIsImh0dSI6Imh0dHBzOi8vc2VydmVyLmV4YW1wbGUuY29tL3Rva2VuIiwia
 WF0IjoxNTYyMjYyNjE2fQ.2-GxA6T8lP4vfrg8v-FdWP0A0zdrj8igiMLvqRMUvwnQg
 4PtFLbdLXiOSsX0x7NVY-FNyJK70nfbV37xRZT3Lg
grant_type=authorization_code
&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
&code_verifier=bEaL42izcC-o-xBk0K2vuJ6U-y1p9r_wW2dFWIWgjz-
```

DPoP proof JWT in HTTP header

Access Token Response

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-cache, no-store

{
   "access_token":"eyJhbGciOiJFUzI1NiIsImtpZCI6IkJlQUxrYiJ9.eyJzdWIiOi
     Jzb21lb25lQGV4YW1wbGUuY29tIiwiaXNzIjoiaHR0cHM6Ly9zZXJ2ZXIuZXhhbXB
     sZS5jb20iLCJhdWQiOiJodHRwczovL3Jlc291cmNlLmV4YW1wbGUub3JnIiwibmJm
     IjoxNTYyMjYyNjExLCJleHAiOjE1NjIyNjYyMTYsImNuZiI6eyJqa3QiOiIwWmNPQ
     09SWk5ZeS1EV3BxcTMwalp5SkdIVE4wZDJIZ2xCVjN1aWd1QTRJIn19.vsFiVqHCy
     IkBYu50c69bmPJsj8qYlsXfuC6nZcLl8YYRNOhqMuRXu6oSZHe2dGZY0ODNaGg1cg
     -kVigzYhF1MQ",
   "token_type":"DPoP",
   "expires_in":3600,
   "refresh_token":"4LTC8lb0acc6Oy4esc1Nk9BWC0imAwH7kic16BDC2"
}
```

Token type indicates that the **access token** is bound to the DPoP public key

Access Token Response Alt.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-cache, no-store

{
   "access_token":"x9C-_-laeb4ioiHicffsIxtpZC36IkJ7qUdrRiv2",
   "token_type":"DPoP",
   "expires_in":3600,
   "refresh_token":"4LTC8lb0acc6Oy4esc1Nk9BWC0imAwH7kic16BDC2"
}
```

DPoP Bound Access Token

JWT & Introspection Response

```
{
   ... other claims / members ...

   "cnf":
   {
     "jkt":"0ZcOCORZNYy-DWpqq30jZyJGHTN0d2HglBV3uiguA4I"
   }
 }
```

Confirmation claim carries the SHA-256 JWK Thumbprint of the DPoP public key to which the access token is bound

Protected Resource Request

```
GET /protectedresource HTTP/1.1
Host: resource.example.org
Authorization: DPoP eyJhbGciOiJFUzI1NiIsImtpZCI6IkJlQUxrYiJ9.eyJzdWI
 iOiJzb21lb25lQGV4YW1wbGUuY29tIiwiaXNzIjoiaHR0cHM6Ly9zZXJ2ZXIuZXhhbX
 BsZS5jb20iLCJhdWQiOiJodHRwczovL3Jlc291cmNlLmV4YW1wbGUub3JnIiwibmJmI
 joxNTYyMjYyNjExLCJleHAiOjE1NjIyNjYyMTYsImNuZiI6eyJqa3QiOiIwWmNPQ09S
 Wk5ZeS1EV3BxcTMwalp5SkdIVE4wZDJIZ2xCVjN1aWd1QTRJIn19.vsFiVqHCyIkBYu
 50c69bmPJsj8qYlsXfuC6nZcLl8YYRNOhqMuRXu6oSZHe2dGZY0ODNaGg1cg-kVigzY
 hF1MQ
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6Ik
 VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUklDUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR
 nMiLCJ5IjoiOVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R1JE
 QSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiJlMWozVl9iS2ljOC1MQUVCIiwiaHRtIj
 oiR0VUIiwiaHR1IjoiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlLm9yZy9wcm90ZWN0Z
 WRyZXNvdXJjZSIsImlhdCI6MTU2MjI2MjYxOH0.lNhmpAX1WwmpBvwhok4E74kWCiGB
 NdavjLAeevGy32H3dbF0Jbri69Nm2ukkwb-uyUI4AUg1JSskfWIyo4UCbQ
```

DPoP public key bound JWT access token

DPoP proof

Protected Resource Request Alt.


```
GET /protectedresource HTTP/1.1
Host: resource.example.org
Authorization: DPoP x9C-_-laeb4ioiHicffsIxtpZC36IkJ7qUdrRiv2
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6Ik
 VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUklDUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR
 nMiLCJ5IjoiOVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R1JE
 QSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiJlMWozVl9iS2ljOC1MQUVCIiwiaHRtIj
 oiR0VUIiwiaHR1IjoiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlLm9yZy9wcm90ZWN0Z
 WRyZXNvdXJjZSIsImlhdCI6MTU2MjI2MjYxOH0.lNhmpAX1WwmpBvwhok4E74kWCiGB
 NdavjLAeevGy32H3dbF0Jbri69Nm2ukkwb-uyUI4AUg1JSskfWIyo4UCbQ
```

DPoP public key bound reference style access token

DPoP proof

Recent Current Status and Updates

Traveled through Frankfurt retuning from the 4th OAuth Security Workshop where DPoP was largely conceived thereby justifying the use of this photo

draft-ietf-oauth-dpop

- -00 WG draft published on April 1st (no joke)
- -01 published on May 1st
  - (not insignificant) Editorial updates
  - More formally define the DPoP Authorization header scheme
  - Define the 401/WWW-Authenticate challenge for the scheme
    - With an algs param
  - Added "dpop_signing_alg_values_supported" authorization server metadata
  - Added "invalid_dpop_proof" error code for DPoP errors in a token request
  - Fixed up and added to the IANA section
  - Moved the Acknowledgements into an Appendix and added a bunch of names (best effort looking back at emails)
- IIW session ~ April 28th (so I'm told)
- Discussed during post 107 WG interim on May 5th
- Some on-list feedback around the same time

[some] Open Questions

Currently pandemic fighting by self-isolating at home in Denver thereby justifying the use of this photo

Threat Model & Objectives

- Lots of opportunity for improvement and clarification
- Honestly, I'm hoping Dr. Daniel Fett can help writing / rewriting these parts of the document
- In the meantime I've 'borrowed' some of his content…
  - https://danielfett.github.io/notes/oauth/DPoP%20Attacker%20Model.html

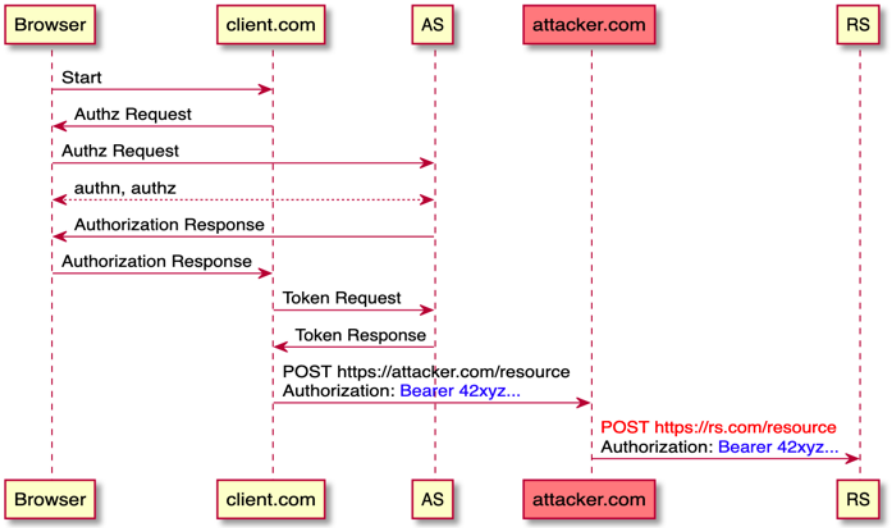
Attacker Model

Misconfigured Resource Endpoint

A resource endpoint is misconfigured. For example, if OAuth Metadata is used, the following configuration can lead to the userinfo endpoint being under the control of the attacker:

```
{
    "issuer": "https://attacker.com",
    "authorization_endpoint": "https://honest.com/authorize",
    "token_endpoint": "https://honest.com/token",
    "userinfo_endpoint": "https://attacker.com/userinfo"  #  ← attacker
}
```

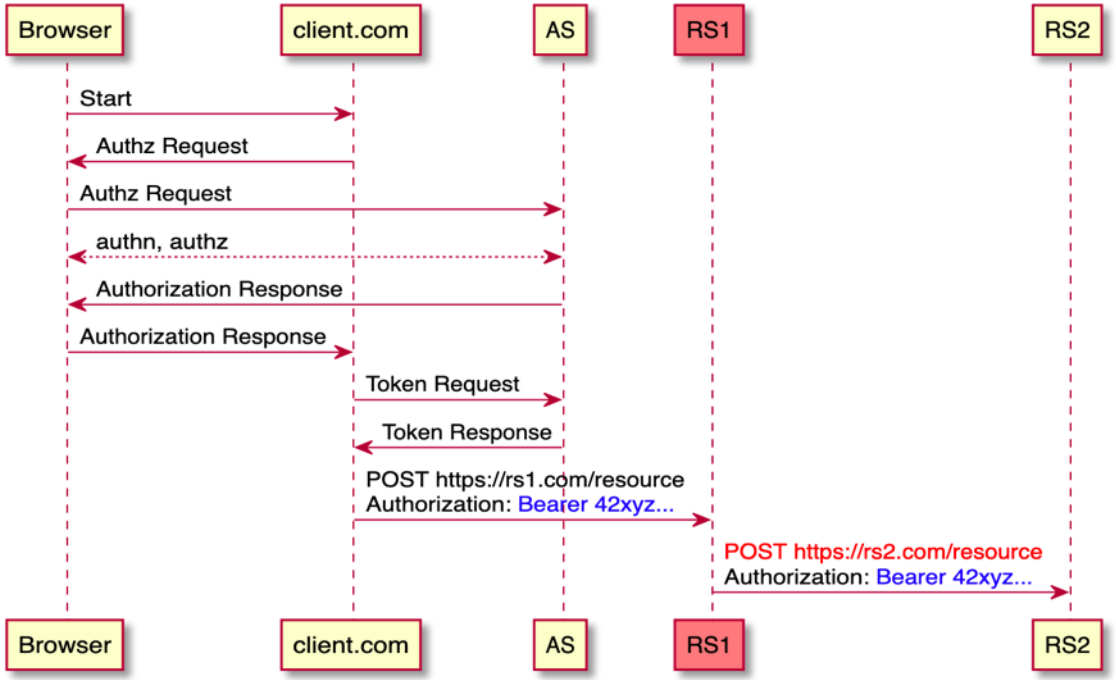
Attack:



Attacker Model Cont.

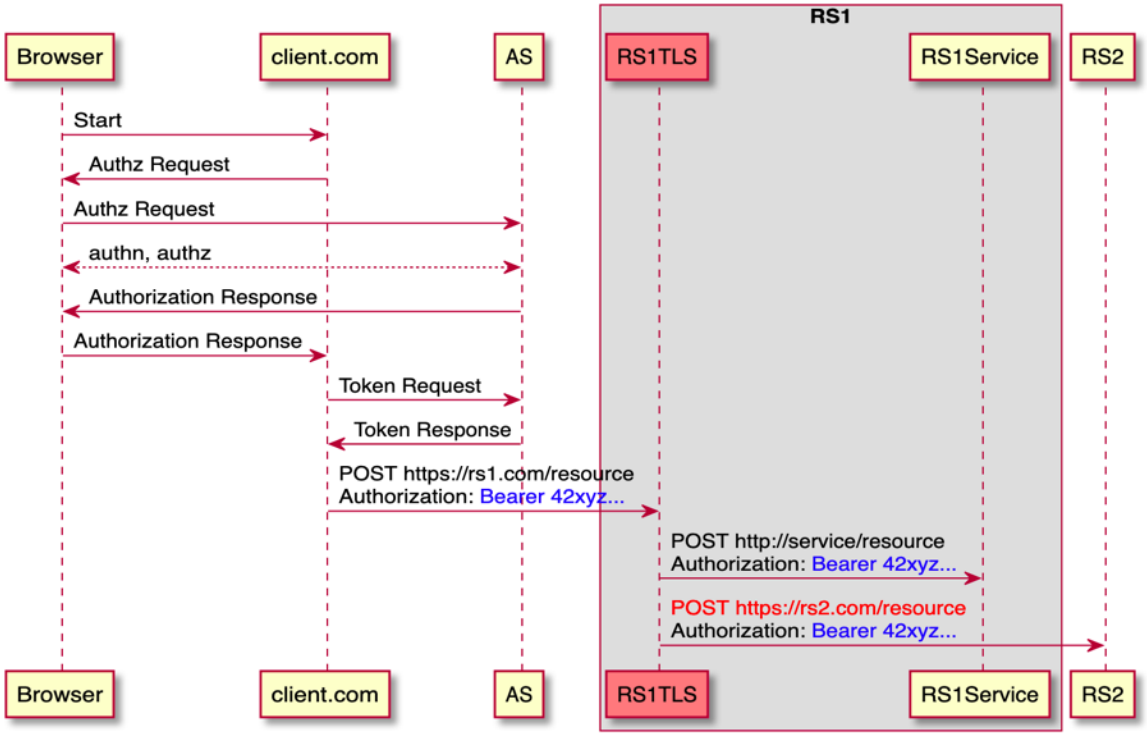
Compromised/Malicious Resource Server

One of multiple resource servers can become compromised or act maliciously for other reasons.



Attacker Model Cont.

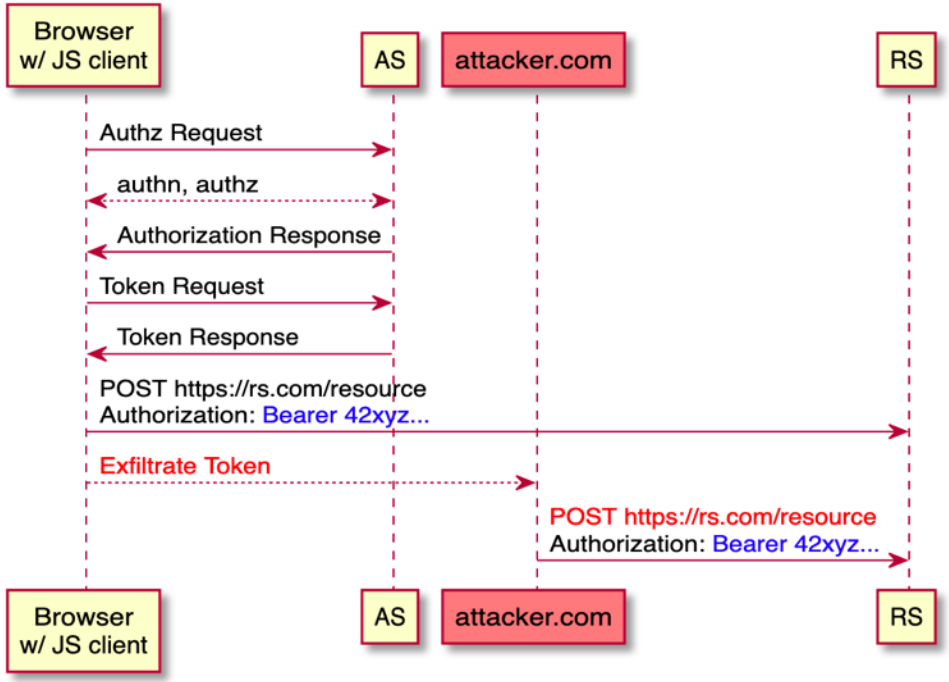
If TLS termination is done at a separate component at the resource server, that component can become compromised, for example by exploting a buffer overflow attack in the reverse proxy:



Attacker Model Cont.

Stolen Token (Offline XSS)

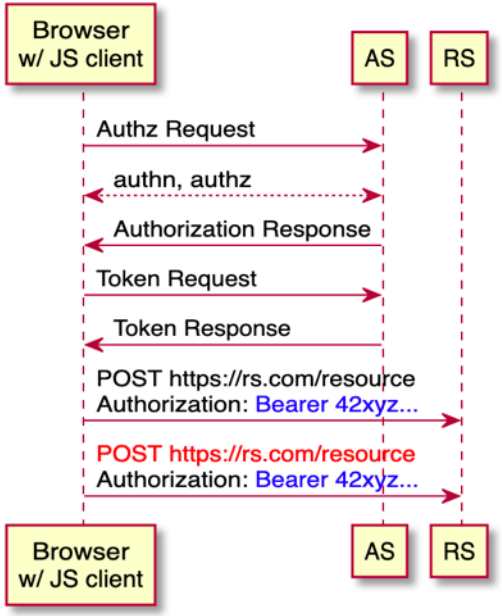
An attacker can leverage an XSS attack to exfiltrate the access token from a single page application.



Attacker Model Cont.

Online XSS (out of Scope?)

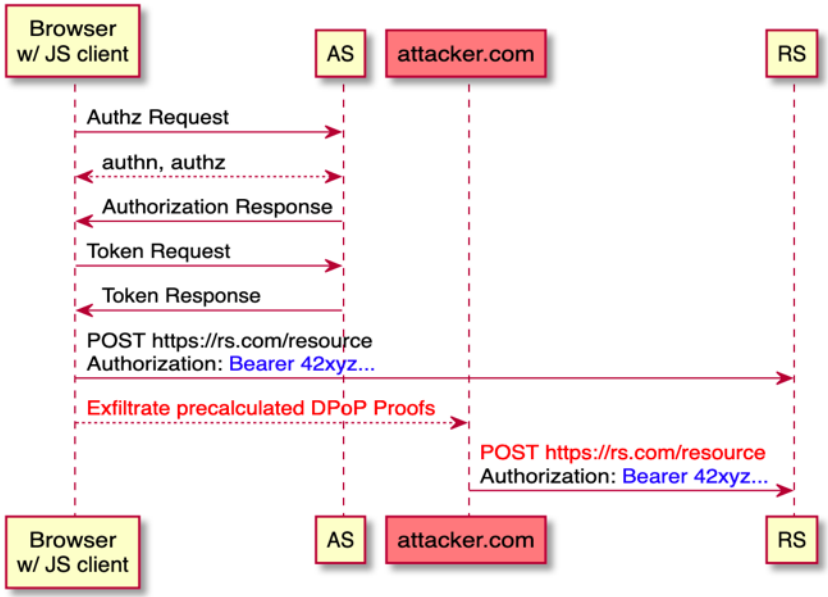
If a user's browser is online and an attacker has injected JavaScript code into the client's SPA, the attacker can use the token without exfiltrating it first. There is most likely no defense against this threat except preventing XSS.



Attacker Model Cont.

Precomputed Proofs

If the attacker is able to precompute DPoP tokens, or is able to exfiltrate the secret key needed for generating DPoP proofs along with the access token, DPoP does not protect the access token if no server-generated nonce is used in the proof.

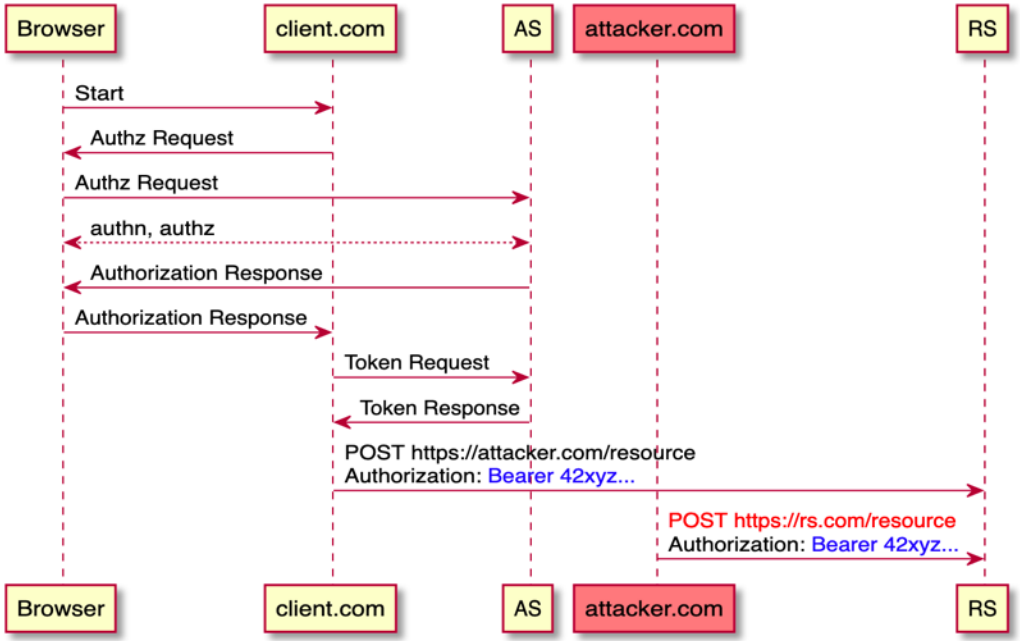


> We should think about requiring/allowing for a server-sent nonce via the WWW-Authenticate header.

Attacker Model Cont.

Exfiltration from Otherwise Secure Channel

Attacks on TLS might allow for the recovery of strings in encrypted messages that are repeated in multiple messages. One example would be the BREACH attacks against HTTP compression.



Symmetric crypto is significantly more efficient than asymmetric

- This is absolutely true but there are other costs/complexities
- Real world implications mostly unquantified
- A couple different potential approaches (at least)
  - Key distribution
  - Key agreement
- Consider this closed (for now anyway) coming out of the pre #107 interim meeting and WG adoption

Difficulties with `jti`

- Issues:
  - Detecting/preventing replay via `jti` can be very problematic for large-scale deployments (also exacerbating inefficiencies asymmetric crypto)
- Current situation:
  - `iat` can also limit replay window
  - Need is unclear
  - replay check on `jti` is only a SHOULD and also qualified “within a reasonable consideration of accuracy and resource utilization, a JWT with the same jti value has not been received previously”
- Some options/ideas … ?
  - Explicitly mention that the replay space is qualified by the URI and method thus reducing the scope of data replication needed
    - There was a mention of splitting path out from htu
  - Further loosen/qualify (like perhaps a MAY)
  - Drop the tracking requirement all together
  - Something else…

Signal that the RT is bound?

- Issue:
  - “useful to be able to have DPoP refresh tokens and Bearer access tokens as a transition step” but “It seems like the spec requires the same token_type for both access tokens and refresh tokens” - IIW summary
    - Note that token_type applies to the access token per RFC 6749
- Current situation:
  - **RTs are only bound for public clients** (this needs apparently needs better treatment in the draft)
  - DPoP access tokens are (most likely) useable as Bearer access tokens
  - Does the client need a signal?
- An option/idea … ?
  - A new token endpoint response parameter could be introduced
    - i.e. “the_refresh_token_in_this_here_response_is_dpopped_so_now_you_know”: true

Client Metadata?

- “were supportive of defining … [Client] Registration Metadata to declare support for DPoP ... [which] might [be] supported token_type values.” – IIW summary
- But the utility of client metadata isn’t entirely clear (at best)
- Short of a legitimate need/use being articulated this one should just be closed out

Downgrades, Transitional Rollout & Mixed Token Type Deployments

- Issue:
  - Topic needs some treatment
- Current situation:
  - Pretty much silent on it so assumptions florish
- An option/idea …
  - An RS supporting both Bearer and DPoP schemes simultaneously needs to update its Bearer token evaluation functionality to reject tokens that are DPoP bound
  - A DPoP only RS is only DPoP
  - A bearer only RS will most likely accept a DPoP bound AT, which helps support mixed/transitional deployments (without a client requesting more granular tokens)

Freshness & Scope of Signature

- Issue:
  - “[no] guarantees that the DPoP signature was freshly generated, as there is no nonce from the server incorporated into the signature.”
- Current Situation:
  - `iat` doesn’t keep it fresh with respect to pre-computation by an adversary who somehow (XSS?!) can use the private key but not steal it
  - No challenge/response was an intentional design choice
- Some options/ideas … ?
  - It’s sufficiently okay as is
  - “People agreed that having a server nonce would add additional security” and “[someone is] already… providing the nonce as a WWW-Authenticate challenge” value– IIW summary
    - *Really* want to avoid adding a challenge/response round trip to every call
    - No challenge available at token endpoint
  - Incorporate a hash of the authorization code, refresh token, access token, other artifact into the DPoP proof
  - Other…

Why did you do it that way?

- Issue:
  - Some variation of the question has come up for many aspects
- Current Situation:
  - DPoP proof JWT header on all requests
  - `Authorization: DPoP <token>` for protected resource access
- Some options/ideas … ?
  - The symmetry and consistency is nice
  - Could alternatively be (and maybe starts to make more sense, if additional context is introduced into the proof):
    - DPoP proof JWT sent to AS as protocol parameter
    - `Authorization: DPoP at=<token>, proof=<proof JWT>` for protected resource access

The flow that shall not be named

- OAuth 2.0 DPoP for the **Implicit** Flow
  - https://tools.ietf.org/html/draft-jones-oauth-dpop-implicit-00 (maybe -01)
- Soliciting reviews and next steps

Gratuitous closing slide featuring the city where will meet together next *

* *Maybe* Bangkok in the fall