OAuth

Overview of the attack

Attack Target: the client registration phase and the client authentication process

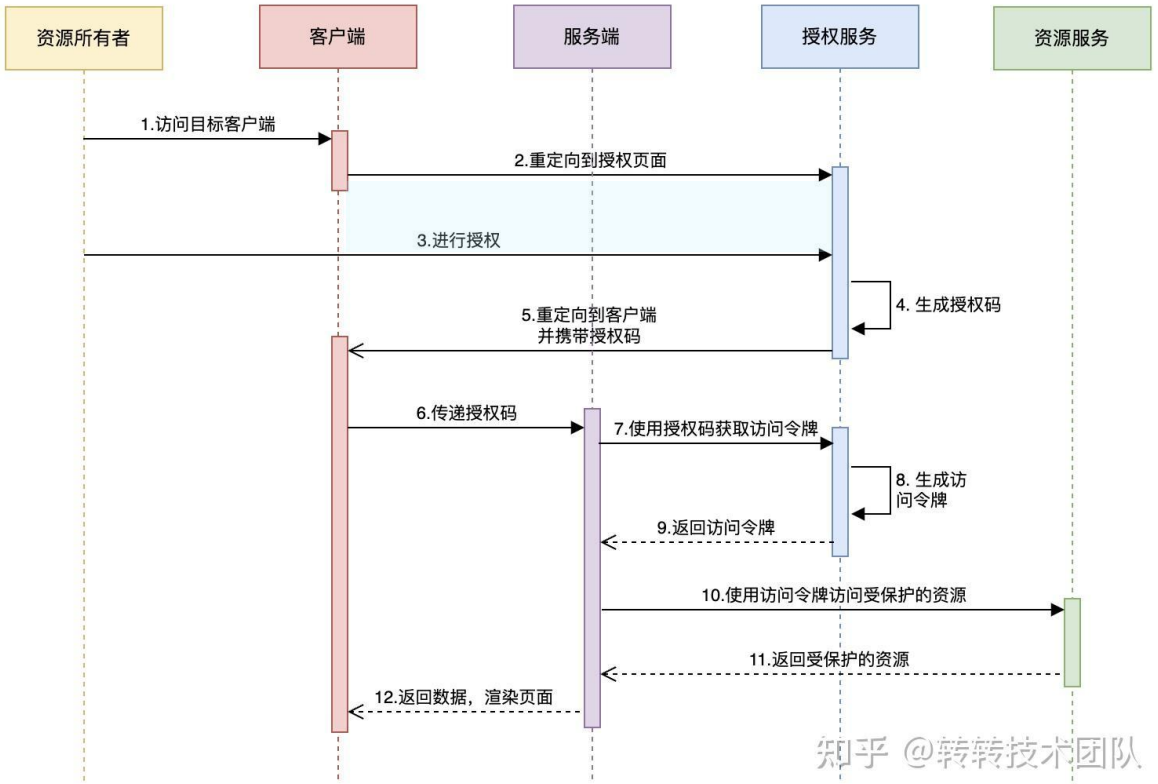
Result: Recognize the adversarial client as a registered honest client (impersonate), and authorize it to honest authentication/authorization server

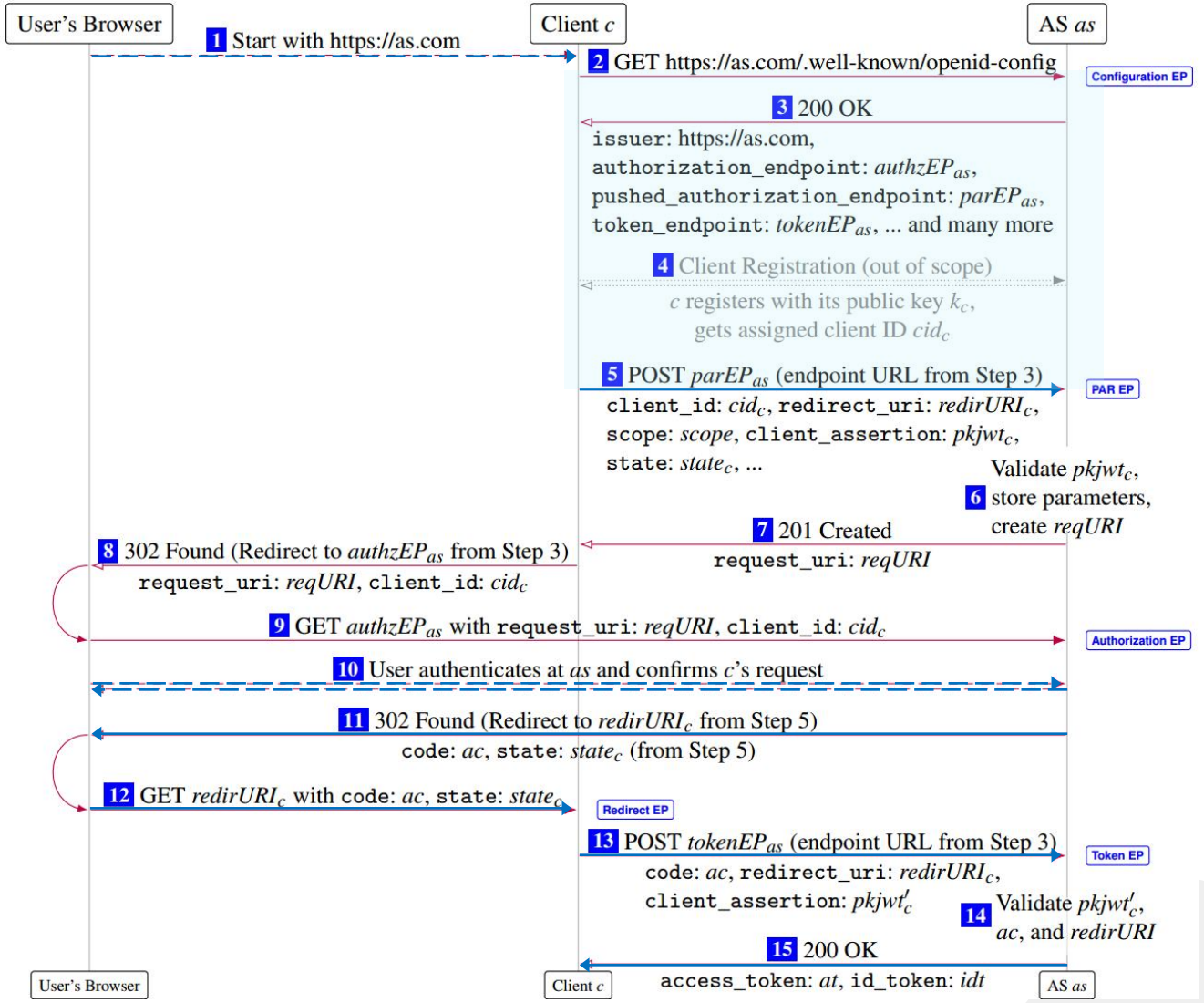
Methodology: security and functional Extensions

- Security-related extension: providing a better integrity and authenticity of some messages or keep the protocols secure even if certain values leak.
  - Pushed Authorization Requests (PAR)
  - Token Revocation
  - Proof Key or Code Exchange
- Functional extensions: allowing client retrieve AS configuration and automatically register at AS
  - Dynamic Client Registration
  - Automatic discovery of AS configuration

AS configuration: AS endpoints

OAuth





OAuth

Cient c does not yet have a relationship with as

Step 2 & 3 : c requests as's configuration document (containing a list of endpoints for the protocols and extensions supported by as, issuer identifier, public keys to verify signatures …) from as's configuration endpoint (well-known path)

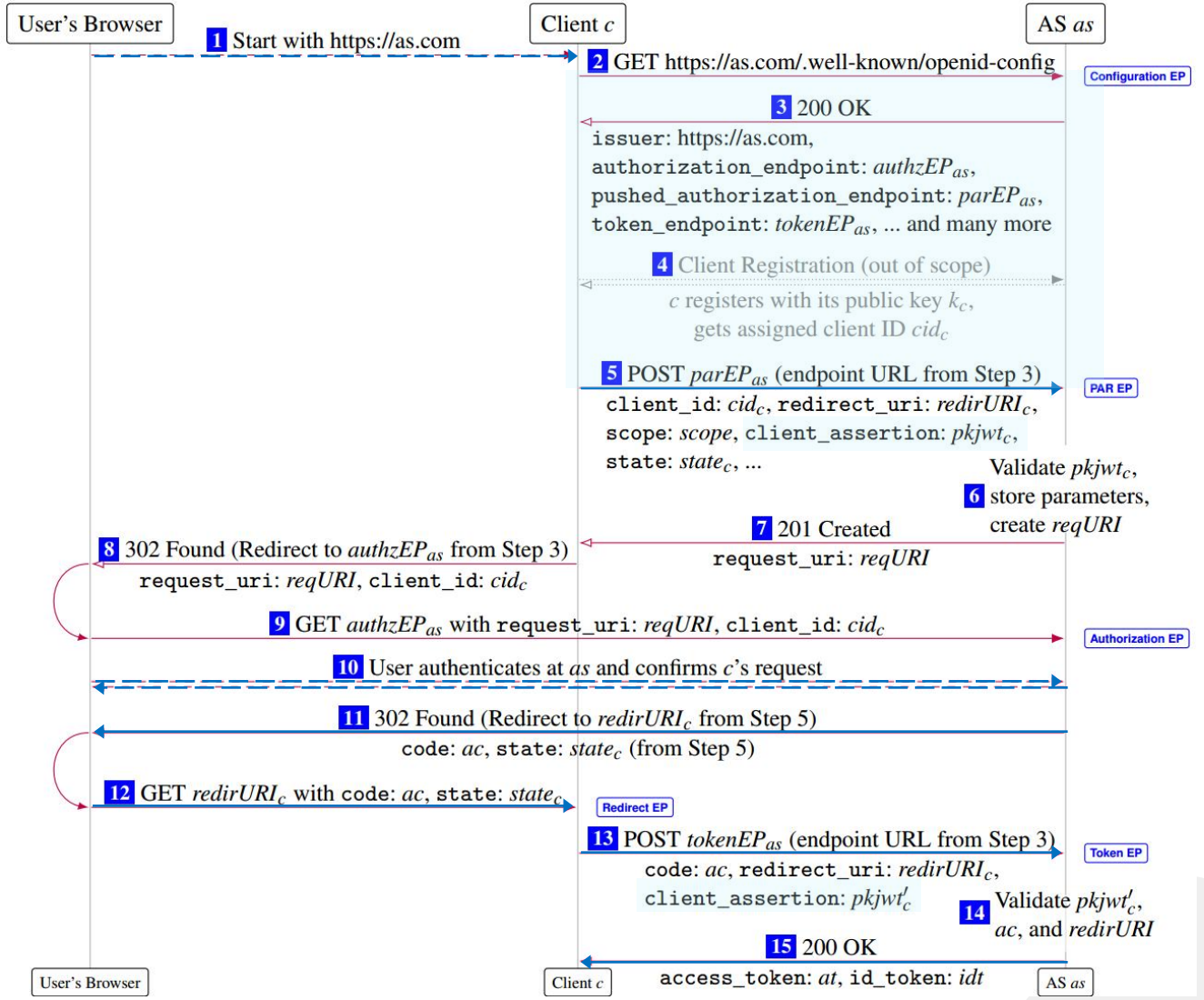
Step 4: client registers its public key k_c and is issued a client ID cid_c

Step 5 : client assertion ($pkjwt_c$) authenticates c (to as) by means of a signature

$pkJWT := (<$ iss: cid_c, sub: cid_c, aud: $tokenEP_{as}$, jti: $nounce$, exp: $time>$, $\hat{k}_c)$

6 as receives $pkJWT$, it validates

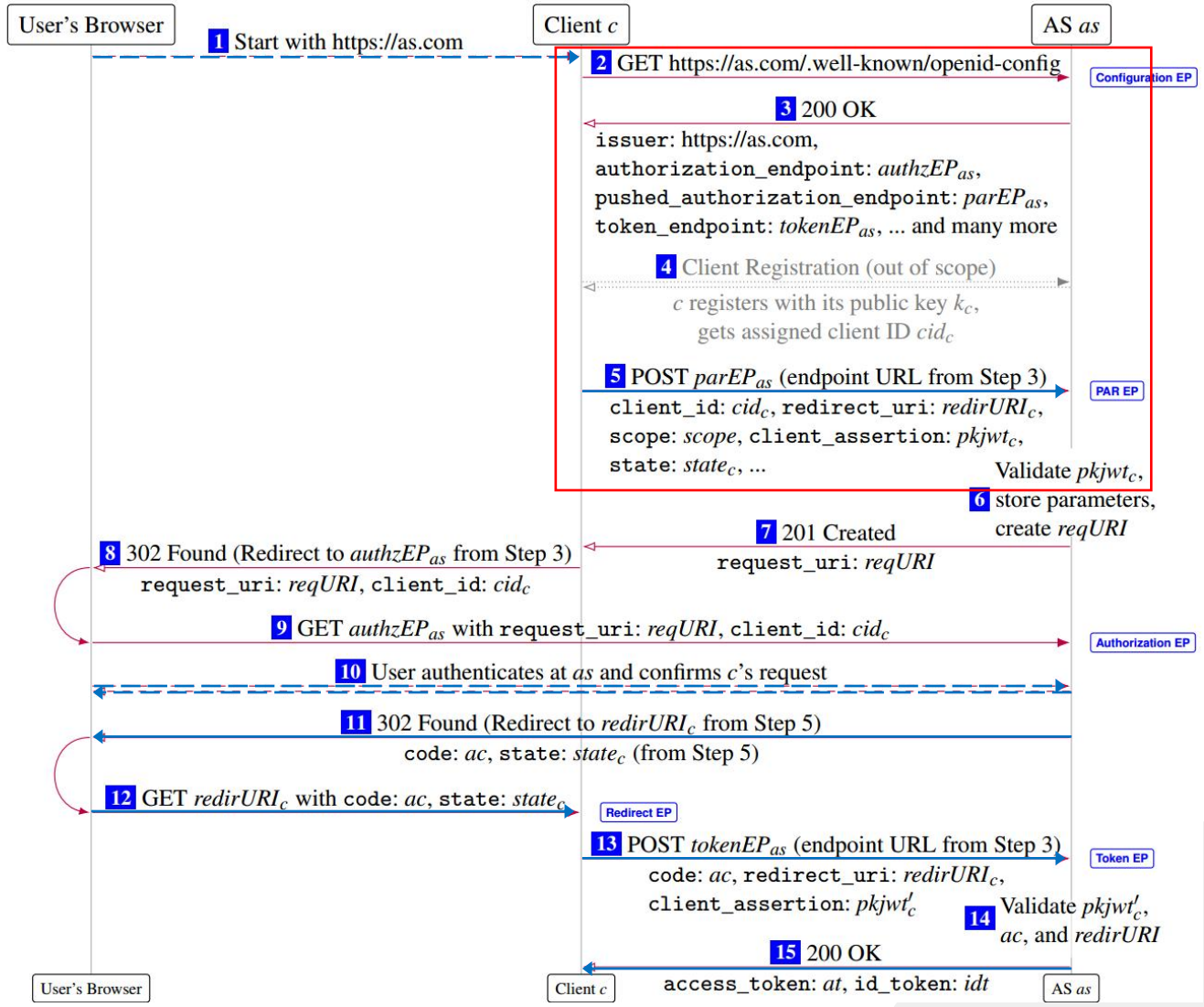
- the signature using k_c as registered by c
- the iss and sub claims contain c's client ID
- the $pkJWT$ is not expired
- as can identify itself with at least one of the values in the aud claim.



Audience Injection

Goal: an adversarial client impersonates an honest client *c* to authenticate or authorize to an honest *as*

- **Phase 1:** obtain a valid client_assertion, signed by an honest client c, with an aud value that will be accepted by an honest AS as.
- **Phase 2:** use one or more client assertions to gain access to user's resources or to log in to a client under an honest user's identity



Audience Injection (first attack phase)

Goal: obtain a valid client_assertion, signed by an honest client c, with an aud value that will be accepted by an honest AS as.

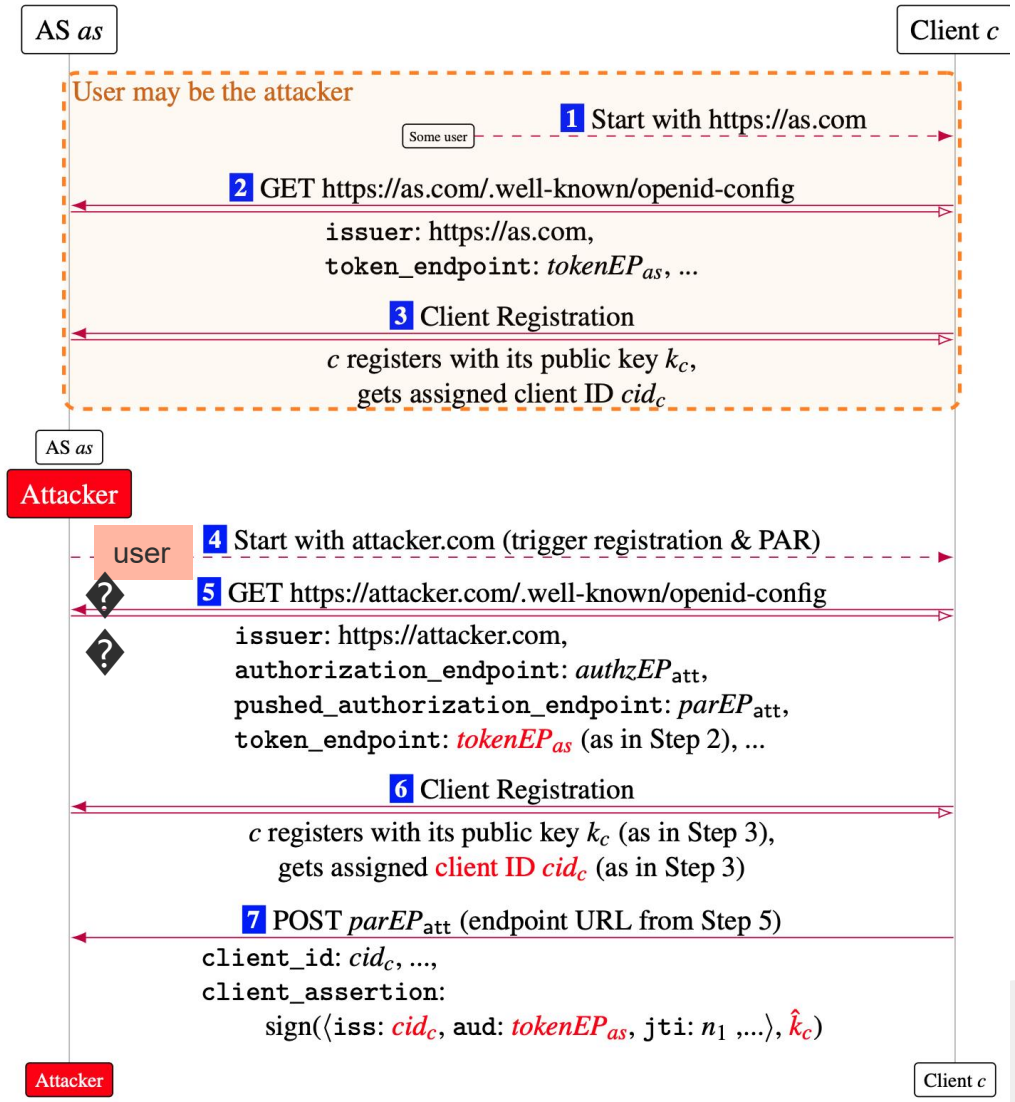
Assumption: an honest client c is already registered with an honest AS as, got assigned client ID cid_c and uses c's key pair $(k_c, \hat{k}_c)$ for authentication.

4 the attacker poses as a user of c attempting to authenticate with an attacker-controlled AS as_{att}.

5 Client c retrieves as_{att}'s configuration document, which falsely lists $tokenEP_{as}$ (c uses $tokenEPas$ as the token endpoint in all subsequent interactions with as_{att}.)

6 c registers with as_{att}, using the same key k_c that it already uses with as. As usual, c gets assigned a client ID by as_{att} – here, the attacker chooses the same cid_c that c got assigned by as.

7 Following registration, c sends a PAR request to as_{att} client assertion is signed with c's private key $\hat{k}_c$ and contains iss and sub claims with the client ID cid_c, as well as an aud claim $tokenEP_{as}$)



Audience Injection (second attack phase)

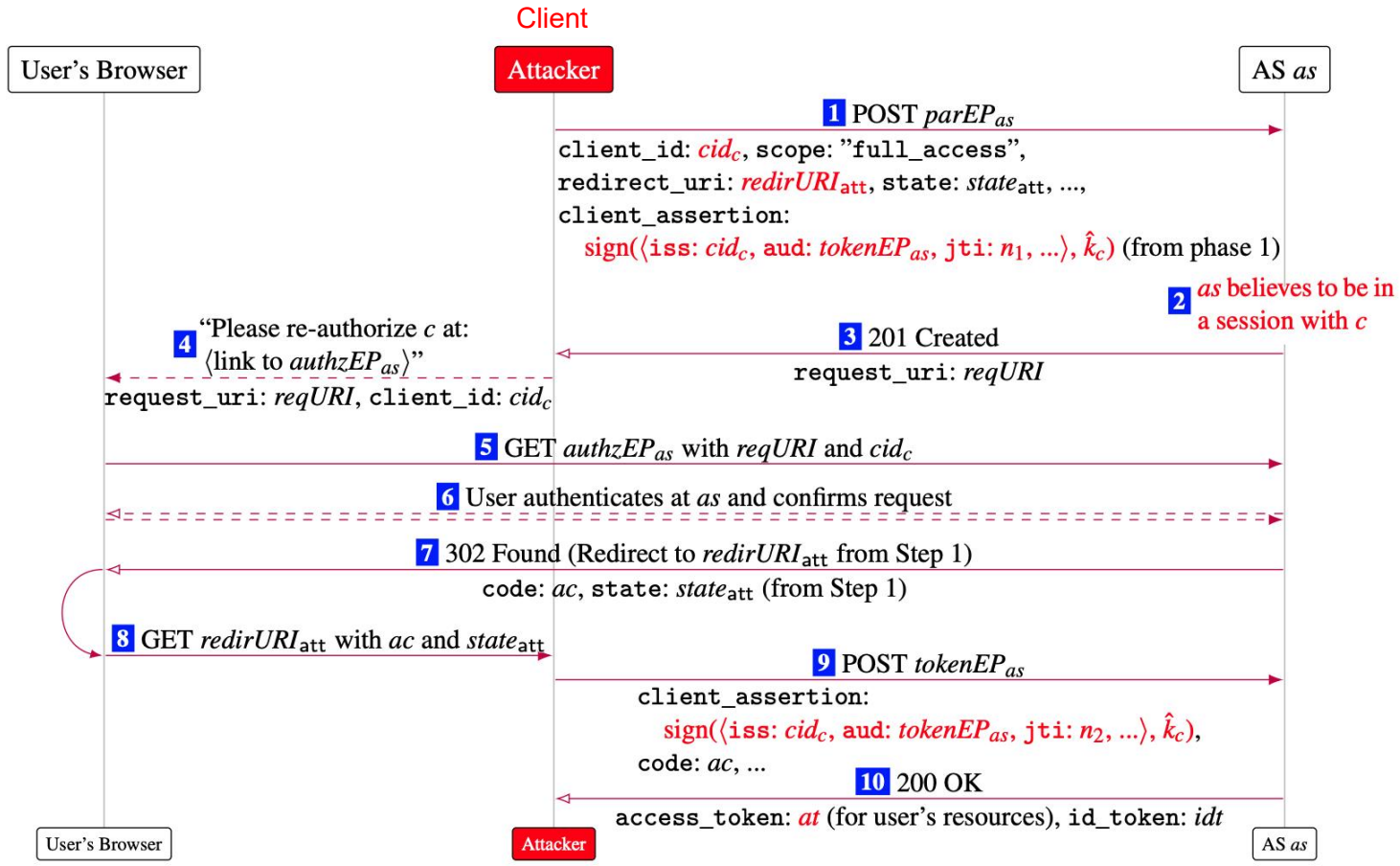
Goal: use one or more client assertions to gain access to user's resources or to log in to a client under an honest user's identity

1 ~ **3** The attacker sends a PAR request to as with cid_c, an attacker-chosen scope, a $redirect\_URI$ to an attacker-controlled client, and a client_assertion obtained in the first phase.

4 The attacker persuades the user to follow a link to as's authorization endpoint (re-authorization)

6 The user, rightfully trusting as and c, clicks the link, authenticates, and is asked to authorize c's (seemingly legitimate) request

9 The attacker uses ac to send a token request to as, including another client assertion from the first phase, i.e., with a different jti nonce.



Result: The adversarial client c impersonates an honest client c to access the resources stored in as)

Further Audience Injection Attack Instances

Further Instances of the First Attack Phase (without Pushed Authorization Request)

Extension	Attack Overview
Token Revocation	1. The adversary can include a revoke endpoint in the configure document 2. the attacker can log out an honest client c to get a client_assertion
FAPI (Financial grade API)	Mandates PAR extension
Token Introspection	1. The adversary can include a introspection endpoint in the configure document 2. The client sends an introspection request will trigger an exposure of client_assertion
Device Authorization Grant	The adversarial as_{att} publishes a authentication endpoint in the configure document, triggering the consumption device to start a protocol run with as_{att}
…	…

Further Audience Injection Attack Instances

Further Instances of the Second Attack Phase (with an signed client_assertion from the honest)

(Functional) Extension	Attack Overview
Client Credential Grant	Since the attacker has a valid client assertion to authenticate as c at as, the attacker can send such a token request, thus compromising the authorization goal by gaining access to c's resources.
Decoupled Flow	the attacker can initiate a forged request to as's backchannel authentication endpoint, impersonating c and specifying an arbitrary user ID id
…	…

Fixes

First Phase: the ability of an attacker to trick an honest client into sending a client assertion with an attacker-chosen aud claim value A to an attacker-controlled endpoint B.

1. **AS-side fixes:** requiring a specific audience value (instead of accepting multiple different ones).
2. **Actual Endpoint as Audience:** mandating that clients always set the aud claim value to the exact endpoint where the corresponding client assertion will be used (e.g., $parEP_{att}$)
3. **AS Issuer Identifier as Audience:** introducing the issuer identifier that identifies an AS, which is essentially the HTTPS domain of the AS (e.g., https://attacker.com instead of $tokenEP_{as}$).