Formal Security Analysis of the OpenID Federation Specification: Private Key JWT Replay

Pedram Hosseyni Ralf Küsters Tim Würtele
{pedram.hosseyni, ralf.kuesters, tim.wuertele}@sec.uni-stuttgart.de
Institute of Information Security – University of Stuttgart, Germany

September 21, 2024

Overview.
For automatic registration, the Federation specification uses Pushed Authorization Requests. Furthermore, Federation lists `private_key_jwt` as a client authentication method without any further restrictions, therefore, private key JWTs can contain the token endpoint of an OP as the audience value. However, a malicious OP can declare arbitrary endpoints in its metadata, in particular, token endpoints of other OPs. As a result, the attacker can obtain valid private key JWTs for an RP which can be used at honest OPs, thus, impersonate the RP.

Audience of Private Key JWTs.
According to OIDC, for the `private_key_jwt` authentication method, the "Audience SHOULD be the URL of the Authorization Server's Token Endpoint" [OIDC, Section 9]. This is also permitted by PAR, which states: "In order to facilitate interoperability, the authorization server MUST accept its issuer identifier, token endpoint URL, or pushed authorization request endpoint URL as values that identify it as an intended audience." [RFC9126, Section 2].

We note that [Fed38, Section 5.1.3] may (depending on how one reads that section, we think that there are several possible interpretations) require the audience value to be either OP's authorization endpoint or OP's Entity Identifier. The attack described in the following still works if the authorization endpoint is used instead of the token endpoint.

Assumptions.

- PAR endpoint can also be used for all regular OIDC flows: [Fed38, Section 12.1.1.2] specifies the usage of the PAR endpoint for automatic registration. We assume that the OP allows the PAR endpoint for all subsequent OIDC flows (i.e., without re-registration).
- Redirect URIs can be chosen by the RP for each authenticated pushed authentication request (regardless of whether that URI was registered before). This is in line with both the Federation specification and with PAR: "The exact matching requirement MAY be relaxed when using PAR for clients that have established authentication credentials with the authorization server. This is possible since, in contrast to a conventional authorization request, the authorization server authenticates the client before the authorization process starts and thus ensures it is interacting with the legitimate client. The authorization server MAY allow such clients to

specify redirect_uri values that were not previously registered with the authorization server". [RFC9126, Section 2.4]

Involved Parties, Identifiers, and Relevant Endpoints.

- Honest OpenID Provider: Let *op* be an honest OpenID Provider with the Entity Identifier $entityID_{\text{op}}$ and token endpoint $tokenEP_{\text{op}}$.
- Honest Relying Party: Let *rp* be an honest Relying Party with the Entity Identifier $entityID_{\text{rp}}$. Let $privKey_{\text{rp}}$ be the private signature key of $entityID_{\text{rp}}$ (as used for OIDC, i.e., not the Federation key) and let $pubKey_{\text{rp}}$ be the corresponding public verification key.
- Attacker: The attacker can act both as an RP and OP, which we denote by att_{rp} and att_{op}, with the Entity Identifier $entityID_{\text{att}}$. In its OP metadata, the attacker chooses the token endpoint of the honest OP as its token endpoint, i.e., $tokenEP_{\text{op}}$. Furthermore, let $redirectURI_{\text{att}}$ be an endpoint of att_{op}.

We assume that *rp* already obtained the metadata for *op* and att_{op}, and describe the attack in the following.

Part 1: RP registers at OP.

First, *rp* performs automatic registration at *op* using the PAR endpoint as specified in [Fed38, Section 12.1.1.2], which, amongst others, allows client authentication using Request Objects and the `private_key_jwt` method. In line with Federation PR#83, we assume that Request Objects are required.[1] For this first step, for the attack to work, it would be sufficient if *rp* uses a Request Object without any other client authentication mechanism. However, the attack does not rely on this, i.e., *rp* may also use an additional client authentication mechanism for this request.[2]

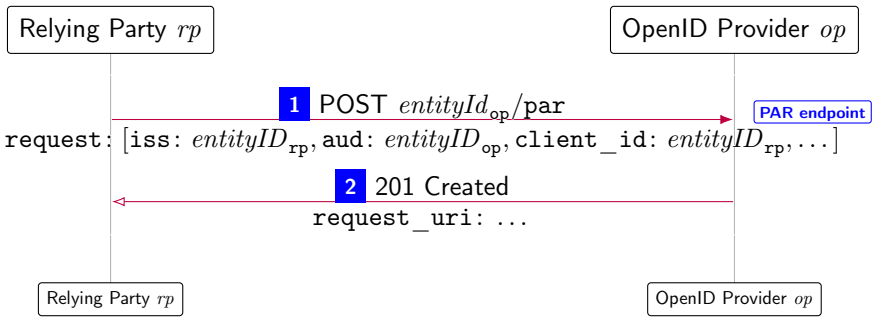

Figure 1: Private Key JWT Replay Attack Part 1: Honest RP registers at honest OP.

Figure 1 shows the relevant values of the pushed authorization request. The OP processes the request and registers $entityID_{\text{rp}}$ using the Federation protocol; in particular, the OP registers $pubKey_{\text{rp}}$ as the signature verification key. The remaining steps of the protocol flow are not relevant and omitted.

[1] We note that it remains somewhat unclear whether one of the other three authentication mechanisms listed in [Fed38, Section 12.1.1.2] must be used during registration (in addition to the signed Request Object).

[2] Note that according to [RFC6749, Section 2.3], clients must not use more than one authentication method in each request – however, request objects are usually not considered as a client authentication method (see [RFC6749, Section 2], [OIDC, Section 9], and [Fed38, Section 5.1.3]). On the other hand, as noted before, [Fed38, Section 12.1.1.2] may allow using request objects for client authentication.

Part 2: RP registers at Attacker.

The *rp* performs automatic registration at att_{op}, and the RP performs client authentication using private key JWT authentication.[3]

As noted above, the RP creates the private key JWT such that the audience value is the token endpoint from att_{op}'s metadata (i.e., the token endpoint of *op*). More precisely, the private key JWT looks as follows:

$$pkJWT = \mathsf{sig}([\texttt{iss}\colon entityID_{\text{rp}}, \texttt{sub}\colon entityID_{\text{rp}}, \texttt{aud}\colon tokenEP_{\text{op}}, \texttt{jti}\colon \ldots, \texttt{exp}\colon \ldots], privKey_{\text{rp}}),$$

with $privKey_{\text{rp}}$ being the private signature key of $entityID_{\text{rp}}$.

This step is done twice, i.e., the attacker obtains two private key JWTs $pkJWT_1$ and $pkJWT_2$. To do so, the attacker (posing as a user of *rp*) could either trigger *rp*'s registration at two different attacker-controlled OPs (both listing $tokenEP_{\text{op}}$ as their token endpoint), or start the same flow twice at the same attacker-controlled OP to obtain the private key JWTs at the PAR endpoint.

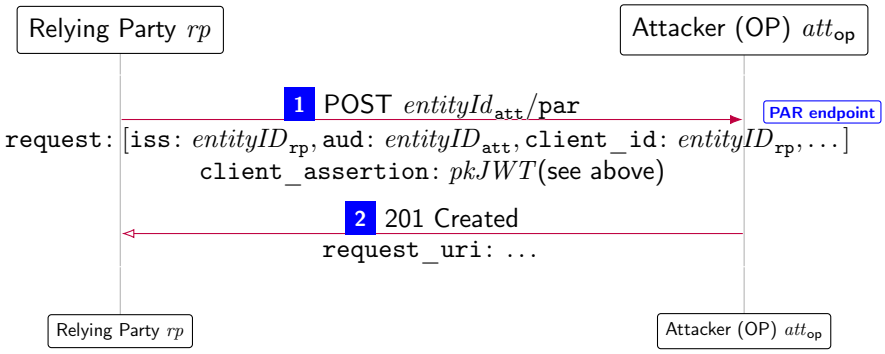

Figure 2: Private Key JWT Replay Attack Part 2: Honest RP registers at malicious OP. att_{op}'s OP metadata lists $tokenEP_{\text{op}}$ as token endpoint (i.e., the honest OP's token endpoint).

Part 3: Attacker Impersonates Honest RP.

The attacker can use the private key JWTs to impersonate the honest RP.[4] We show a possible attack in Figure 3: In Step 2, the attacker starts a flow and uses one of the previously obtained private key JWTs, thus, impersonating *rp*. The attacker also specifies an endpoint that he controls as the redirection URI. After receiving the `request_uri`, the attacker redirects the honest user to *op* in Step 4.

Note that the user will be asked to authorize *rp* in Step 6 (from OP's perspective, the honest user performed an OIDC authorization code flow with *rp* – not with att_{rp}). We assume that the user will accept that request – however, consider the following scenario: The attacker sends an email to the user, copying the corporate design of *rp*, and asks the user to "confirm" their registration with *rp* in order to continue using *rp*'s services. Conveniently, that email contains a link that the user can click to do so; this link points to an attacker website that resembles *rp*'s website, where the user now starts the OIDC flow (cf. Step 1). Hence, the user *expects* to authorize *rp* at *op* (note that we do NOT assume the user to enter any credentials on an attacker site).

After the OP redirects the user back to att_{rp} in Steps 7 and 8, the attacker can send a token request to *op*, again using one of the previously obtained private key JWTs.

[3] Note: Even if Federation PR#83 in combination with [RFC6749, Section 2.3] would prohibit the use of authentication mechanisms besides request objects for automatic registration, then such a private key JWT will be created and sent to the attacker-controlled att_{op} in all subsequent OIDC protocol flows between *rp* and att_{op}.

[4] As described in [RFC6749, Section 10.2], preregistered redirect URIs would prevent such an impersonation, which, however, is not mandatory when using PAR with client authentication at the PAR endpoint.

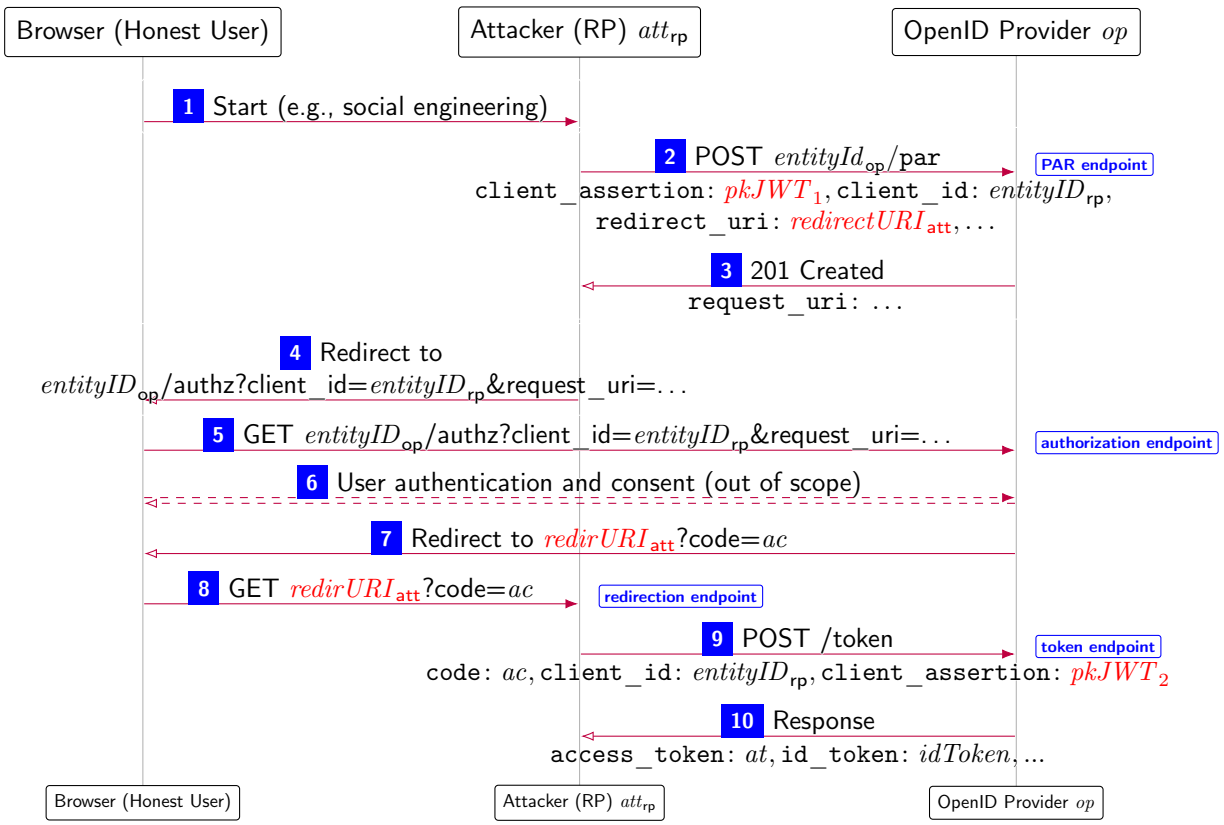

Figure 3: Private Key JWT Replay Attack Part 3: The attacker uses the private key JWTs obtained before to impersonate the honest *rp* towards the honest *op*.

Attack on Authorization. The attacker can use the access token obtained in Step 10 to gain access to resources of an honest user (that are managed by the honest *rp*, i.e., the token can contain any scope that the honest *rp* may request).

Attack on Authentication. Instead of redeeming the authorization code at the token endpoint in Step 9, the attacker could also start another flow at the honest *rp* and inject the authorization code, thus logging in at *rp* as the honest user.

Note that state/nonce do not prevent this attack: Before luring the honest user into starting a flow, the attacker would first start its own flow at *rp* to obtain the nonce value in the authentication request. The attacker can then include this value in the authentication request in Step 2. Likewise, in "its" flow, the attacker can take the state value from the authentication request and include it in the authentication response.

References

[RFC6749] D. Hardt. *The OAuth 2.0 Authorization Framework*. RFC 6749. Oct. 2012. DOI: 10.17487/RFC6749. URL: https://www.rfc-editor.org/info/rfc6749.

[RFC9126] T. Lodderstedt, B. Campbell, N. Sakimura, D. Tonge, and F. Skokan. *OAuth 2.0 Pushed Authorization Requests*. RFC 9126. Sept. 2021. DOI: 10.17487/RFC9126. URL: https://www.rfc-editor.org/info/rfc9126.

[Fed38] M. B. Jones, A. Å. Solberg, J. Bradley, G. De Marco, and V. Dzhuvinov. *OpenID Federation 1.0. Draft 38*. Ed. by R. Hedberg. OpenID Foundation, Aug. 19, 2024. URL: https://openid.net/specs/openid-federation-1_0-38.html.

[OIDC] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore. *OpenID Connect Core 1.0 incorporating errata set 2*. OpenID Foundation, Dec. 15, 2023. URL: http://openid.net/specs/openid-connect-core-1_0.html.