5.7 Discovered Weaknesses

5.7.1 Incorrect implementation of at_hash

In the TARA Technical Specification [32], the identity token has at_hash value that is not according to the OIDC spec [33].

When looking at the id token response, it has a property at_hash with the value of X0MVjwrmMQs/IBzfU2osvw==. This value is supposed to be base64url encoded. Instead, it is a regular base64 string. The Demo REST Client example provided by the same authors [71] correctly converts the base64 value into the base64url encoded value, which leads us to believe that there is a mistake in the documentation and or implementation. If it was following the specifications of the TARA documentation and not OIDC spec, it should have no reason to do so.

The eeID service follows the TARA documentation, and because of that, the at_hash uses base64. An issue arises when the using OpenID Connect libraries (see listing 4).

```
IDX21348: Validating the 'at_hash' failed, see inner exception.
IDX21300: The hash claim: 'UtsKV8+hA/bB0EE/xR9cCQ==' in the
    id_token did not validate with against: 'AT-95-
    VU6Y2LZjrNrVCdhlEaCxG6Gpzt0RsE-Z', algorithm: 'RS256'.
```

Listing 4. Microsoft.IdentityModel.Protocols.OpenIdConnect fails to validate at_hash

If we compute the hash manually, we see precisely why the verification failure happens (see listing 5). The implementation expects a different string than was provided. The same transcoding behavior is seen on the TARA Demo Client [71].

```
user@localhost:~$ access_token="AT-95-
    VU6Y2LZjrNrVCdhlEaCxG6Gpzt0RsE-Z"
user@localhost:~$ echo -n $access_token | openssl dgst -binary -
    sha256 | head -c 16 | base64
UtsKV8+hA/bB0EE/xR9cCQ==
user@localhost:~$ echo -n $access_token | openssl dgst -binary -
    sha256 | head -c 16 | base64 | tr '/+' '_-' | tr -d '='
UtsKV8-hA_bB0EE_xR9cCQ
```

Listing 5. Verifying at_hash manually

The implication of this discovery means that all working clients who use eeID have incorrect OpenID connect implementation. This issue affects only those who use the correct OpenID Connect implementation libraries.

The reason for the incorrect implementation stems from backward compatibility [65].

5.7.2 Confusing state and nonce behavior

Nowhere in the OpenID Connect specification mentions that state should be transferred over to the id_token. TARA confused the purposes of state and nonce properties, extended

the behavior to cover each other, and, by extension, made one of the properties obsolete.

The state property is part of the underlying OAuth2.0 specification, where it is an "opaque value used by the client to maintain state between the request and callback" [58]. The primary security feature is to prevent CSRF attacks [58, 66].

The nonce property is part of the OpenID Connect specification, and its primary purpose is to prevent replay attacks when using implicit or hybrid flows [33]. Later, researchers discovered that it could also protect against authorization code injection attacks with the code flow. The disadvantage of using nonce as a state parameter is that it directly influences the size of the id_token, which should be kept as small as possible. The reason for keeping this token as small as possible is so that developers could later send them in request headers to their resource servers [72]. The only issue is that these tokens expire after 40 seconds and cannot be refreshed [32], making this approach impractical.

If we look at the data flow diagram (see figure 15), we see that both state and nonce have each other's properties. After the user agent returns to the callback URL, both nonce and state are returned when only state is required. After the company's authentication server establishes a backchannel and redeems the code for an id_token, this token again contains both nonce and state when only nonce is required.

In the security analysis performed last year, a researcher suggested removing the nonce parameter from the protocol [73]. We disagree with this approach and would suggest removing the state parameter from the id_token response. Removing the nonce parameter and having no support for PKCE would break OIDC compliant libraries' ability to mitigate authorization code injection attacks.

5.7.3 Wrong claims in the OpenID Connect discovery endpoint

The discovery endpoint provides all information about possible requests and responses. The data listed there does not match the documentation. For example, in the discovery endpoint for eeID (`https://auth.eeid.ee/oidc/.well-known/openid-configuration`), claims like gender are present, even though this claim can never appear inside of the identity token. On the flip side, the claim profile_attributes, as described in the documentation, is missing from the discovery document.

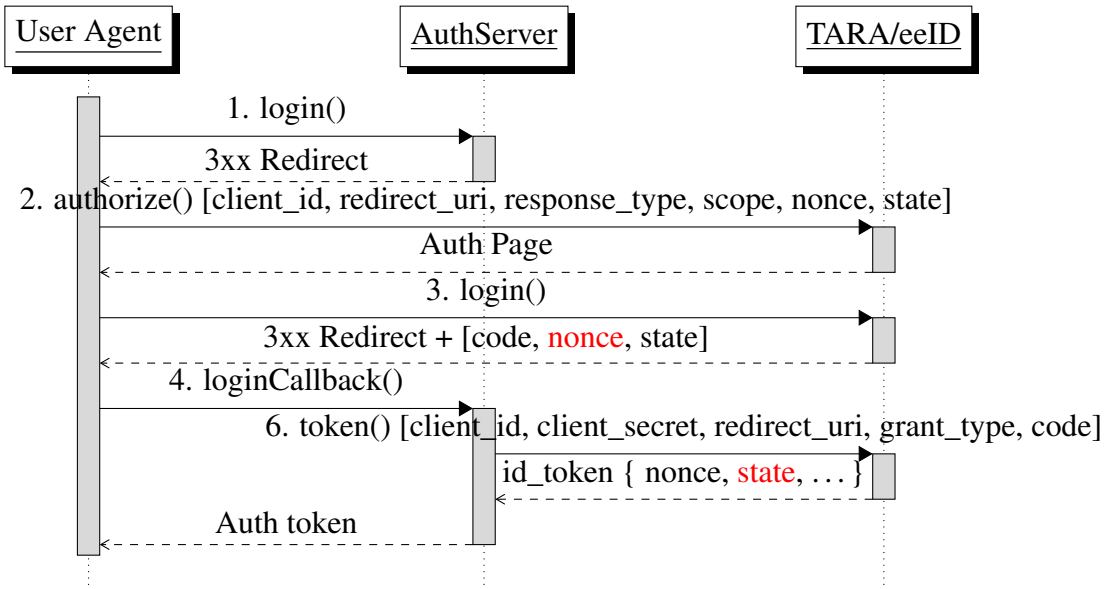

Figure 15. The incorrect OIDC code flow used in TARA/eeID