Wie funktioniert der OpenID Connect Logout?

Tolleiv Nietsch - 2024-05-15 - OpenID Connect

OpenID Connect Logout

Bare.ID ermöglicht als Single Sign-On-Lösung eine zentrale und damit sichere und komfortable Anmeldung für die verbundenen Applikationen. Doch wie verhält es sich mit dem Logout? Wie wird sichergestellt, dass Nutzer von allen Applikationen auf verschiedenen Geräten und Browsern wieder ordentlich abgemeldet werden? Der folgende Artikel erklärt die verschiedenen Mechanismen des OpenID Connect-Standards, die Bare.ID nutzt, um einen geordneten Logout-Prozess sicherzustellen.

Single Sign-out oder Single Log-out

Den Gegenpart zum Single Sign-on (SSO) bildet der Single Sign-out oder Single Logout (SLO). Was SSO so praktisch und sicher macht, gilt auch für SLO. Nutzer beenden mit dem Klick auf einen Abmelden-Button alle aktiven Sessions der mit Bare.ID verbundenen Applikationen, ohne dass sie sich aktiv um den Logout von allen einzelnen Applikationen kümmern müssen. Sie werden zentral abgemeldet, so wie sie auch zentral angemeldet wurden. Auf diese Weise wird sichergestellt, dass keine einzelnen aktiven Sessions verbleiben, die als Einfallstor für Datendiebstahl oder andere Angriffe genutzt werden.

Logout in einer Applikation (RP-Initiated Logout)

Was geschieht also im Hintergrund, wenn ein Nutzer auf den "Logout-Button" in einer mit Bare.ID verbundenen OIDC-Applikation klickt? Zunächst stellt die Applikation selbst sicher, dass die aktive Nutzersession beendet und der lokale Sicherheitskontext des Benutzers aufgehoben wird. Der Nutzer wird aus der Applikation ausgeloggt.

Als nächstes erfolgt die Kommunikation an den OpenID-Provider. Bare.ID stellt einen Logout-Endpunkt namens `end_session_endpoint` zur Verfügung. Die Applikation stellt eine Anfrage (Request) zum Logout an die hinterlegte URL. Ein solcher Logout-Request sieht dann z.B. so aus:

Dieser Logout-Request beinhaltet folgende Parameter:

- **id_token_hint** (empfohlen): Der zuvor von Bare.ID bereitgestellte ID-Token, um der Applikation die Identität des Nutzers und der aktiven Session mitzuteilen.
- **post_logout_redirect_uri** (optional): Es kann eine URL mitgeliefert werden, zu der umgeleitet werden soll, nachdem die Abmeldung durchgeführt wurde. Die URL muss zuvor bei Bare.ID hinterlegt worden sein.

- **state** (optional): Ein Status kann in Form eines zufälligen Strings mitgeliefert werden. Dieser wird dann durch Bare.ID in den Callback zu `post_logout_redirect_uri` aufgenommen. Auf diese Weise kann die Applikation den Status zwischen der Abmeldeanfrage und dem Callback überwachen.

  Weitere Informationen zum OpenID Connect RP-Initiated Logout finden sich in der Spezifikation unter https://openid.net/specs/openid-connect-rpinitiated-1_0.html.

Bare.ID ist jetzt informiert, dass der Nutzer sich abmelden möchte und kann darauf reagieren. Ist der `id_token_hint` Nach dem Redirect zum Logout-Endpunkt wird dem Nutzer eine Abmeldeseite angezeigt und gefragt, ob er sich von Bare.ID und somit zentral von allen verbundenen Applikationen abmelden möchte.

Klickt der Nutzer auf den "ABMELDEN"-Button, müssen alle verbundenen Applikationen mit aktiven Sessions für diesen Nutzer informiert werden, dass die Sessions beendet und der Nutzer abgemeldet werden muss. Hierfür bietet der OpenID Connect-Standard verschiedene Mechanismen an, die von Bare.ID unterstützt werden:

Session Management

Das OpenID Connect Session Management definiert einen Mechanismus, der es einer verbundenen Applikation ermöglicht, den Login-Status eines Nutzers bei Bare.ID ständig zu überwachen. Auf diese Weise können Nutzer von der Applikation abgemeldet werden, sobald sie sich zentral bei Bare.ID ausloggen. Dabei wird, wie auch beim Front-Channel-Logout, eine reine Front-Channel-Kommunikation verwendet.

Meldet sich ein Nutzer über Bare.ID bei einer Applikation an, wird in der Authentication Response auch der `session_state`-Parameter an die Applikation übermittelt. Der darin enthaltene String wurde zuvor serverseitig errechnet und repräsentiert den Login-Status des Nutzers auf der Seite von Bare.ID. Derselbe String wird nun auch von der Applikation berechnet und kann regelmäßig mit dem Wert aus dem erhaltenen `session_state`-Parameter verglichen werden.

Um die Menge an übertragenen Daten dabei gering zu halten, rendern sowohl Bare.ID, als auch Applikationen, die Session Management unterstützen, einen versteckten iframe. Der applikationsseitige iframe muss die eindeutige ID des Bare.ID-seitigen iframes kennen und kann dann die Daten, bestehend aus Client ID und Session State, von dort per postMessage pollen. Dabei dürfen ausschließlich Daten, die gesichert aus dem Bare.ID-iframe stammen, verarbeitet werden, um Cross-Site-Scripting-Attacken zu verhindern.

Ändert sich der Session State also auf Seiten von Bare.ID, weil der Nutzer sich zentral

abgemeldet hat oder die Session abgelaufen ist, können alle Applikationen darauf reagieren und den Nutzer auf ihrer Seite ausloggen.

> *Weitere Informationen zum OpenID Connect Session Management finden sich in der Spezifikation unter https://openid.net/specs/openid-connect-session-1_0.html.*

Front-Channel-Logout

Auch der Front-Channel-Logout-Mechanismus wird vom Browser durchgeführt. Im Gegensatz zum Session Management wird dabei aber kein iframe auf Seiten der Applikationen benötigt. Es muss lediglich der Endpunkt `frontchannel_logout_uri` für eine Logout-URI registriert werden, die vom Browser aufgerufen werden kann, um einen Logout des Nutzers für die Applikation durchzuführen.

Bare.ID rendert für jede Applikation, die eine aktive Session für einen Nutzer von Bare.ID hat und die den Front-Channel-Logout-Mechanismus unterstützt, einen separaten iframe mit der entsprechenden Logout-URI. Meldet sich ein Nutzer von Bare.ID ab, kann er durch parallelen Aufruf aller Logout-URIs per HTTP-Request von allen verbundenen Applikationen gleichzeitig abgemeldet werden.

Die Applikation löscht bei Aufruf der Logout-URI alle relevanten Cookies und leert den lokalen Speicher. Wird die Applikation jetzt vom Nutzer aufgerufen, wird im Browser die erneute Anmeldung des Nutzers verlangt.

> *Weitere Informationen zum OpenID Connect Session Management finden sich in der Spezifikation unter https://openid.net/specs/openid-connect-frontchannel-1_0.html.*

Back-Channel-Logout

Front-Channel-Logout und Sessions-Management nutzen Front-Channel-Kommunikation und benötigen eine aktive Session der verbundenen Applikation im geöffneten Browser des Nutzers. Um einen Nutzer von allen Applikationen abmelden zu können, ohne dass dieser einen Browser öffnen muss, stellt der OpenID Connect-Standard den Back-Channel-Logout zur Verfügung. Er spezifiziert eine Server-zu-Server-Kommunikation für den Logout-Request und ist der zuverlässigste Logout-Mechanismus, da er die Unsicherheiten durch einen Nutzer-Browser umgeht.

Loggt der Nutzer sich bei Bare.ID aus, wird ein Logout-Request an den Logout-Endpunkt aller Applikation mit einer aktiven Session gesendet. Dieser Request beinhaltet einen Logout-Token (JSON Web Token JWT), ähnlich dem ID-Token. Die Parameter des Logout-Tokens beinhalten u.a. eine eindeutige Session-ID, um die zu beendende Session eindeutig identifizieren zu können. Die Applikation validiert den Token, führt den Logout aus und liefert im Erfolgsfall einen HTTP-Status 200 zurück. Refresh-Token der aktiven Sessions

verlieren bei Bare.ID ihre Gültigkeit.

> *Weitere Informationen zum OpenID Connect Session Management finden sich in der Spezifikation unter https://openid.net/specs/openid-connect-backchannel-1_0.html.*