Authentik e il social login

Ci sono dei casi in cui un identity provider esiste già, magari con fatto con `keycloak`, ma i suoi manutetori sono sufficientemente bestie da non voler apportare customizzazioni per le tue applicazioni: per esempio potrebbero avere la bella idea di usare il `preferred_username` come chiave identificativa e averla identica al *codice fiscale* o *vat id* se vogliamo dirlo alla esterofila, o ancora potrebbero decidere che si passa il name, il family_name, ma non entrambi insieme. Questo comportamento opinabile è abbastanza terribile: come fare? Facile, si prende il nostro Authentik, si fa federare con l'identity provider gestito che pure il mio cane sa fare meglio e, a quel punto, si agisce direttamente sui campi che il **nostro** IDP (politically correct) fornisce. Lo schema altamente tecnico è il seguente:

```
subCaneCloak [preferred_username, name, family_name]--> IdP_PC --> [preferred_username_manipolato, full_name (composto da manipolazione di name_family_name)]
```

Le manipolazioni sono estremamente semplici in Authentik perché sono ottenibili andando ad intervenire con linguaggio `Python` sui dati ottenuti.

Passo 1: San Francesco parla con gli animali

Dopo aver indossato il mantello di San Francesco, è possibile interagire con la sottospecie addetta alla gestione del *subCaneCloak* richiedendo un accesso di tipo `Confidential`.

Pro:

- c'è uno sbatty iniziale pari a zero
- da questo momento le applicazioni che usano il vostro Authentik PC, ottengono credenziali valide direttamente dall'upstream IdP

Contro:

- appena ne trovo uno ve lo dico

A questo punto andiamo sul nostro IdP

Passo 2: Configurazione del social login

Una volta entrati sull'*interfaccia amministrativa* guardiamo intensamente il menù di sinistra: *Cartella --> Federazione e social login* e creiamo una nuova entry di tipo *Sorgente Oauth di OpenID* e compiliamo a modo:

- Nome: subCaneCloak
- Slug: subcanecloak
- Modalità di corrispondenza utente: Collega gli utenti su un identificatore univoco
- Modalità di abbinamento di gruppo: Collega gli utenti su un identificatore univico
- Percorso utente %(slug)s <-- Questo è dove va a creare l'alberatura. Troverete gli utenti sotto *Cartella --> Utenti -->* sezione dedicata (nel nostro caso *subCaneCloak*)
- Impostazioni del protcollo: qui ci schiantiamo il client_id e il secret che ci è stato fornito dalle veshtie
- Ambiti: profile openid email
- Impostazioni URL: questi parametri vanno valorizzati con quello che vi viene fonito dai dementi di cui sopra
- PKCE Method: None
- Modello di autenticazione di codice di autorizzazione: HTTP_BASIC

Ora c'è una questione: se lasciamo così, il tutto funziona, MA non stiamo effettuando nessuna manipolazione dei campi. Se volessimo attuarla dovremmo proseguire con... **Attenzione**: lasciamo in standby questa finestra, oppure concludiamola: ma al passo 4 dovremo tornare su questa per modificarla.

Passo 3: customizzazione dei campi: il property mapping

Vogliamo un mapping tale che, quando un utente si logga via OAuth, il suo profilo su Authentik sia creato con un nome completo formattato correttamente (es. "Mario Rossi"), indipendentemente da come i dati arrivano dalla sorgente. Senza questo mapping, Authentik potrebbe importare i nomi esattamente come passati dal provider, col rischio di avere una lista utenti disordinata (es. alcuni tutti in minuscolo, altri con solo il nome e senza cognome). Come fare?

Guardiamo molto, molto intensamente, alla Chuck Norris e oltre (senza sfasciare tutto a calci), il menù a sinistra: *Personalizzazione --> Mappatura proprietà --> Crea* e creiamo una nuova *Mapping delle proprietà sorgente OAuth* e chiamiamola come meglio ci aggrada:

- Nome: subCaneCloal-mapping
- Espressione:

```
given_name = info.get("given_name", "")
family_name = info.get("family_name", "")

return {
    "username": info.get("preferred_username"),
    "email": info.get("email"),
    # .title() converte "MARIO" -> "Mario" e "ROSSI" -> "Rossi"
    "name": f"{given_name.title()} {family_name.title()}".strip()
}
```

`info.get()` è quella funzione che dice ad Authentik di estrarre i campi da quanto ricevuto dall'upstream

Passo 4: modifica del social login

Riprendiamo da dove abbiamo lasciato al passo 3 (o se l'avessimo conclusa, torniamo su *Cartella --> Federazione e social login* e modifichiamola). Scorriamo in fondo fino a *Mappatura degli attributi Oauth* e sia sulla *Mappatura proprietà Utenti* che su *Mappatura proprietà Gruppo* scegliamo il mapping che abbiamo appena creato. Fine.

Passo 5: abilitazione del social login

Abbiamo creato il social login ma ancora non è attivo. Per far sì che abbia effetto, deve essere associato alla fase di login. Per cui tanto per cominciare capiamo dove avviene.

Dal menù a sinistra: *Sistema --> Brands*. Modifichiamo *authentik-default* e scendiamo nella sezione *Flussi predefiniti* e guardiamo che cosa è valorizzato come *Flusso di autenticazione*: se non avete apportato modifiche sarà *default-authentication-flow (Welcome to authentik!)*. Abbandoniamo questa schermata e andiamo su *Flussi e Fasi --> Fasi* e modifichiamo quella che abbiamo appena visto, ovvero

default-authentication-flow (Welcome to authentik!). Nella sezione *Impostazioni sorgenti* spostiamo il social login creato prima nella parte di *Fonti selezionate* e aggiorniamo. Adesso il social login sarà attivo.

[Che il talpone sia con voi]

[EOF]

Fluxer Icon

Canale di discussione e supporto

Canale di discussione e supporto. Per confrontarti su configurazioni o altro entra su fluxer.dreadful.work (https://fluxer.dreadful.work/invite/dreadful).