Deception Component Generator

Riccardo Barbieri

15 dicembre 2023

Indice

1 Introduzione **3**

2 Requisiti **3**

3 Identity provider **4**
3.1 Concetti chiave 4
3.1.1 Assegnamento ruoli 4
3.1.2 Client 5

4 Progettazione **5**
4.1 Architettura 5
4.2 Definizione componente 5
4.3 Business logic 6
4.3.1 Registrazione entità 7
4.4 Generazione dati 7
4.4.1 Mockaroo 7
4.4.2 Generation controller 8
4.5 Deployment 8

5 Client linea di comando **8**

6 Conclusioni **9**
6.1 Sviluppi futuri 9

1 Introduzione

In questo documento verrà discusso lo sviluppo di `deception-kit`, un sistema software in grado di generare servizi e componenti fasulli allo scopo di esporre bersagli a valore nullo a eventuali attori malevoli.

Nel mondo della sicurezza informatica è da tempo diffuso l'utilizzo di misure di difesa basate sull'inganno, queste soluzioni mirano a esporre risorse e servizi che possono essere considerate di particolare interesse per possibili attaccanti; l'approccio principale si basa sulla creazione di un componente *honeypot* come punto di ingresso fasullo al perimetro di sicurezza di una organizzazione.

Un honeypot dovrebbe essere progettato per sembrare ricco di informazioni agli occhi di un potenziale attaccante senza però lasciar trasparire la sua natura di esca esponendo servizi non sufficientemente protetti che possono apparire sospetti ad attori malevoli: l'obiettivo principale è quello di bilanciare questi attributi per indurre un attaccante a **sprecare** più risorse possibili su un obiettivo futile.

Questo progetto mira a fornire una suite di strumenti atti a semplificare la creazione di suddetti honeypot, offrendo una interfaccia di facile utilizzo che considera tutte le fasi richieste nella generazione di un componente di questo tipo.

2 Requisiti

Il generatore deve essere in grado di creare una immagine compatibile con OCI Image Format, standard open source industriale formulato dalla Open Container Initiative.

Il formato immagine generato deve contenere tutte le informazioni fittizie necessarie al corretto funzionamento del componente, deve inoltre essere possibile istanziare un container partendo dalla definizione generata, che dovrà quindi comprendere tutte le configurazioni rilevanti.

Questa prima versione del progetto si concentrerà sulla generazione di un identity provider, un componente che permette ad altri servizi di delegare ad esso l'autenticazione e l'autorizzazione dell'accesso da parte di utenti o sistemi automatici.

Nel particolare caso di un identity provider dovranno essere:

- generati utenti fittizi;
- configurati gruppi e ruoli di sicurezza;
- abilitati e configurati flow di autenticazione oAuth per proteggere altri servizi.

Un ulteriore requisito che il progetto rispetterà consiste nell'usabilità dello strumento: si mira a bilanciare il livello di dettaglio delle possibili configurazioni e la rapidità di deployment del componente.

3 Identity provider

Come primo passo nello sviluppo di questo sistema è stato necessario scegliere l'identity provider sul quale si baserà le generazione e studiarne il funzionamento.

Si è deciso di utilizzare Keycloak, un identity provider open source che offre un varietà molto ampia di configurazioni, adattandosi a svariati casi di utilizzo. Offre tutte le feature che alternative closed source come Okta o Microsoft Azure Active Directory offrono a scapito di una barriera di più alta, ampiamente colmata da una documentazione molto dettagliata e una community altrettanto attiva.

Tuttavia il vantaggio principale che ha guidato questa scelta è la possibilità di effettuare self-hosting del servizio, opzione non disponibile con soluzioni closed source, in particolare Keycloak è predisposto all'utilizzo tramite container Docker, offre anche uno strumento per generare una immagine ottimizzata in base alle configurazioni utilizzate.

3.1 Concetti chiave

Le feature che Keycloak offre sono svariate, si elencano di seguito quelle rilevanti ai fini del progetto:

- supporto per OpenID Connect
- supporto per OAuth2.0
- Single Sign On/Out per applicazioni browser
- Two/Multi Factor Authentication

Keycloak suddivide tutte le configurazioni relative a utenti, client, ruoli, gruppi e flow in realm, ogni realm è isolato dagli altri esistenti e può gestire e autenticare soltanto utenti che controlla per client che controlla.

3.1.1 Assegnamento ruoli

I ruoli possono essere assegnati a singoli utenti, client e gruppi.

Ruoli assegnati a utenti possono essere sfruttati dai client per implementare strategie di Role Based Access Control, è possibile anche assegnare ruoli a un particolare gruppo per associare un set di ruoli a un utente.

3.1.2 Client

Un client è una entità che può richiedere a Keycloak di autenticare un utente o un altro servizio e ottenere le informazioni di autenticazione relative una volta autenticato.

Keycloak supporta il concetto di *direct grant* permettendo ai client di richiedere accesso ad altri servizi tramite Keycloak, entrano in gioco i ruoli assegnati a singoli client.

4 Progettazione

4.1 Architettura

Allo scopo di creare un sistema software estensibile e flessibile, si è deciso di incapsulare il core della business logic dietro a una HTTP API, questo approccio permetterà di:

- creare metodologie di interazione diverse
- distribuire i componenti del sistema
- semplificare integrazioni con altri sistemi

La metodologia di invocazione principale per il servizio sarà realizzata tramite un tool da linea di comando.

4.2 Definizione componente

Alla base di `deception-kit` si trova la definizione del componente: per permettere all'utente di specificare le caratterische che il componente generato dovrà avere è stato predisposto deception-def, un formato di file basato su yaml.

Il formato di questa specifica si suddivide in due parti:

- tipologia di componente da generare, chiave `component`
- definizione delle parti che costituiscono il componente

Nel caso del componente "id-provider" la specifica accetta la definizione di `groups`, `users`, `clients` e `roles`; ognuna di queste si suddivide in due sezioni:

- parametri generali della specifica
- definizione di una lista di users, groups...

Questo formato di definizione è stato pensato per rendere possibile la specifica di un sottoinsieme rilevante di caratteristiche per ogni categoria di risorse necessarie alla generazione del componente, bilanciando la granularità della configurazione e semplicità di utilizzo; le opzioni che non è possibile definire tramite questo formato sono impostate tramite valori di default fornite da Keycloak.

L'infrastruttura che opera la conversione di questa definizione testuale in entità è stata progettata per essere facilmente estensibile.

4.3 Business logic

L'API che implementa la business logic di generazione è stata realizzata tramite il framework Java Spring, questa decisione è stata presa per il fatto che questo framework semplifica la creazione della configurazione degli endpoint e la gestione delle dipendenze, permettendo di concentrarsi sulla business logic.

Da un punto di vista di alto livello la generazione del componente id-provider avviene come segue:

- creazione di una istanza temporanea di Keycloak
- registrazione di client, utenti, gruppi e ruoli
- associazione dei ruoli a gruppi, utenti e client
- estrazione della configurazione risultante dall'istanza di Keycloak
- generazione della definizione dell'immagine

Per registrare le varie entità sull'istanza di Keycloak si è deciso di creare una istanza temporanea del server Keycloak, in questo modo è possibile sfruttare il runtime del server per:

- configurare i parametri non specificati con valori di default;
- assicurare la consistenza delle configurazioni.

Keycloak server offre strumenti a supporto della configurazione del server, come ad esempio la possibilità di esportare e importare configurazioni di singoli o multipli realm; vengono usati questi strumenti per esportare la configurazione creata sull'istanza temporanea in formato json.

La definizione dell'immagine specifica comandi da eseguire alla creazione della stessa che importano la configurazione generata nella nuova istanza del server.

4.3.1 Registrazione entità

L'API espone i seguenti endpoint:

- `registerClients`
- `registerUsers`
- `registerGroups`
- `registerRoles`
- `assignRoles`

questi endpoint accettano richieste HTTP POST che trasportano liste delle entità da registrare.

Il server Keycloak espone una API per effettuare tutte le operazioni necessarie alla configurazione e gestione del servizio, per effettuare la registrazione delle entità il core del sistema sfrutta questa API stabilendo una connessione alla ricezione di una richiesta HTTP a uno degli endpoint dedicati.

4.4 Generazione dati

deception-kit permette anche di delegare la generazione di dati fittizi tramite il file di definizione yaml: la specifica delle entità `user` e `group` accetta un parametro `total` intero, questo numero definisce quanti utenti o gruppi generare in totale compresi quelli definiti nel file di defizinone.

4.4.1 Mockaroo

Per la generazione dei dati si è deciso di utilizzare Mockaroo, un servizio che permette di generare dati mock realistici basandosi su uno schema definito tramite json; questa decisione è guidata dal fatto che il servizio offre una vastissima gamma di tipi di dato, come ad esempio il tipo `Username` o `Department` che sono stati utilizzati per generare alcuni dei dati utilizzati da deception-kit.

Mockaroo espone una API HTTP per generare i dati, nel progetto è stata utilizzata la versione non self-hosted, è tuttavia possibile effettuare un deployment self-hosted tramite l'acquisto di una licenza enterprise.

Sono state create apposite classi per definire il modello di dati che Mockaroo deve generare che, usate in combinazione con l'astrazione `MockarooApi` e `MockFactory`, permettono di ottenere istanze delle entità del modello utilizzato per registrare le risorse sul server Keycloak.

4.4.2 Generation controller

Il `GenerationController` è il componente che coordina la generazione dei dati mock a partire dalla definizione del servizio e restituisce tutte le risorse generate.

Questo componente riceve la definizione come corpo della richiesta post che lo invoca, un primo livello di accertamento avviene grazie al parser yaml che verifica la consistenza della definizione con il modello, seguono poi accertamenti sulla correttezza della semantica, ad esempio:

- riferimenti a gruppi o ruoli esistenti
- consistenza delle specifiche del numero di gruppi per utente

Si è deciso di non generare mock per i client in quanto devono essere definiti correttamente per poter proteggere vere "finte API" e sarebbe difficile ottenere configurazioni esistenti con una generazione casuale; è stato scelto di lasciare la definizion dei ruoli all'utente in quanto sono di semplice definizione e devono essere consistenti con i client e gli utenti generati.

4.5 Deployment

`deception-core` può essere avviato come normale applicazione java, tuttavia il deploymento consigliato è effettuato tramite docker: sono state predisposte apposite task Gradle che al rilascio di una nuova versione generano l'immagine docker, questa immagine viene poi pubblicata sul registry Docker di default; questo approccio permetterà a `deception-cli` di istanziare il demone dell'API se non ancora in esecuzione.

5 Client linea di comando

Come indicato in precedenza il metodo principale per interagire con l'API `deception-core` è una applicazione da linea di comando.

Alla versione corrente l'interfaccia `deception-cli` supporta il comando `generate` che accetta i parametri:

- `--component`: del componente da generare
- `--definition`: nome del file che contiene la definizione

Un esempio:

```
deception-cli generate --component "id-provider" ---definition definition.yaml
```

Una volta lanciato il comando l'applicazione si occuperà di:

- istanziare il demone `deception-core` se non ancora in esecuzione
- creare l'istanza temporanea di Keycloak

Una volta avviato il server Keycloak vengono effettuate le necessarie richieste HTTP al demone.

Una volta registrate tutte le risorse necessarie, il tool sfrutta l'applicativo `certbot` per generare i certificati TLS, questi verranno inclusi nella definizione dell'immagine.

Completata la generazione delle risorse viene estratta la configurazione Keycloak dall'istanza temporanea e generato il Dockerfile finale con i parametri di interesse correttamente configurati.

Una volta generato il file Dockerfile contenente la definizione dell'immagine viene stampato a video il comando che può essere utilizzato per istanziare un container a partire dalla definizione generata.

6 Conclusioni

In questo documento è stato presentato lo sviluppo di una prima versione di `deception-core` e `deception-cli`, un sistema software per generare componenti fittizi per sviare impegno e risorse di attori malevoli su target futili.

6.1 Sviluppi futuri

L'obiettivo a lungo termine consiste nel fornire una suite completa di strumenti per creare interi sistemi di deception components, aggiungendo supporto ad altre tipologie di componenti basandosi sull'infrastruttura creata in questa versione iniziale.

A supporto di questo obiettivo si intende aggiungere supporto per la creazione di piani di *infrastructure provisioning* tramite Terraform che definisce il deployment dei componenti generati.

Verrà inoltre sviluppata una logging facility dedicata ai visualizzare lo stato di tutti i componenti generati e di cui è stato effettuato il deployment.