RAPPORT PROJET V&V

Fuzzing API REST

Trinôme :

Annaig Butterworth

Chouaib Hentabli

Irifi Faicel

Master 2 Genie Logiciel

ISTIC - Université Rennes 1

Année universitaire 2016-2017

Introduction :

L'objectif de notre travail est la réalisation d'un outil qui permet de tester des Api Rest, définie à l'aide du framework « Swagger ».

Afin de tester une Api Rest et de vérifier si elle se comporte comme c'est définie dans la spécification de cette dernière. Pour cela, on a utiliser le principe du Fuzzing .

Ce principe réside dans l'injection des données aléatoires dans les entrées d'un programme.

Solution proposé :

Dans cette partie, nous nous intéressons à la solution que nous proposons.

Dans un premier lieu, nous allons citer les objectifs viser par notre outils. Ensuite, nous allons décrie les différentes étapes suivi afin d'implémenter notre outil.

Parmi les objectifs de notre solution :

- Notre outil doit fonctionner pour n'importe quel Api Rest définie à l'aide de swagger.
- Générer des Url aléatoirement et de vérifier si l'Api en question retourne toujours l'erreur 404.
- Vérifier si les données retournées par l'opération GET conforme bien à leurs spécification.
- Récupérer pour chaque opération, tous les codes des réponses possibles. Ensuite, vérifier si la requête générer par notre outil pour cette opération retourne un code qui existe dans les codes récupérés.

Pour commencer, chaque Api Rest définie à l'aide de swagger contient un fichier nommé « swagger,json», ce fichier permet de décrire cette Api.

Comme première étape, nous avons besoin d'un outil qui permet de parcourir ce fichier et de le lire dans des POJO Java. Pour cela, nous avons utilisé un outil qui s'appelle « swagger parser ».

Ensuite, à travers cet outil on peut identifier tous les url possibles, leurs opération (GET, POST…) ainsi leurs paramètres.

La deuxième étape consiste à implémenter le principe de Fuzzing. Dans cette étape, on va générer des données aléatoirement en fonction des entrées dont on a besoin (exemple : générer des chaînes de caractères, des entiers…, ou aussi générer un objet JSON qui sert comme entrée pour une requête de type POST ou PUT. Comme ils y a plusieurs types de données, afin de simplifier le processus de création de ces dernières. On a utilisé un patron de conception créationnel qui permet d'instancier des objets dont le type est dérivé d'un type abstrait.

La troisième partie consiste à implémenter la fonction permettant de faire des requêtes. Elle prend en paramètres les données générés. Ensuite, selon le type de l'opération elle appelle une fonction qui permet de formater l'url dans un format qui correspond au type de l'opération.

Finalement, cette partie consiste à la création d'une suite de tests qui permet de vérifier chaque type d'opération (GET, POST...) dans le but de détecter des bugs.

Difficultés rencontrées :

Parmi les difficultés rencontrés :

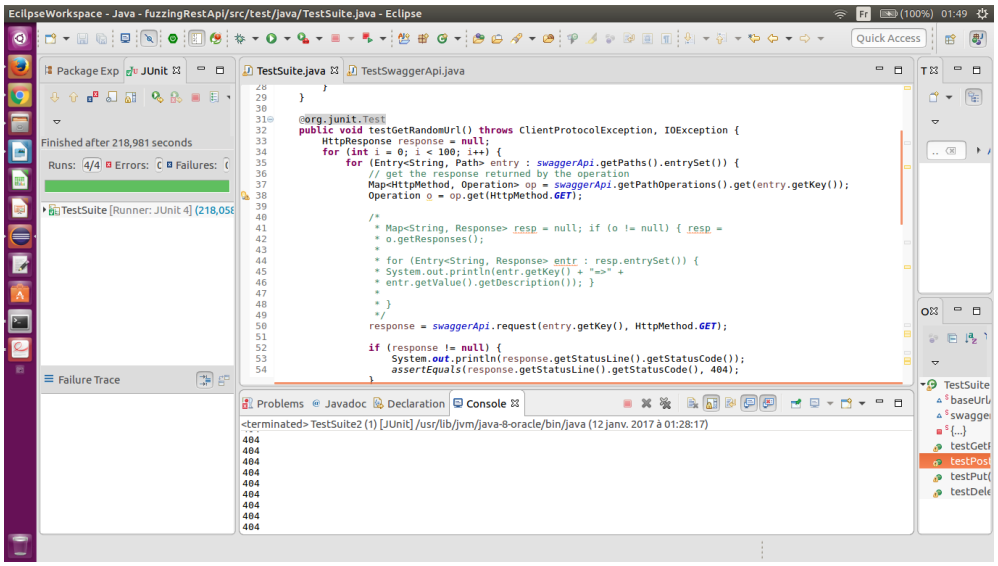
- Toujours faire une correspondance entre un objet « property » et un objet « paramètre », dans le but de détecter le type du paramètre, vu que ce dernier ne contient pas un champs qui permet de le détecter.
- Faire plusieurs opérations sur des chaînes de caractères, dans le but de bien formater un url.
- Le non trouvaille d'un oracle junit qui permet de tester si une valeur existe dans un intervalle de valeurs. Par exemple un oracle AssertIn(valeur, <List> ListValeurs).

Résultats :

Notre objectif principale, c'était de récupérer tous les codes de réponses possibles pour chaque opération et de tester le code retourner par notre requête s'il existe parmi les codes récupérés.

Vu les difficultés mentionnées juste en haut, on a pas pu atteindre tous les objectifs fixer. Pour cela, dans notre test on a vérifier seulement le code d'erreur 404, qui signifie que l'url généré n'est pas supporter par l'Api.

Dans ce qui suit, nous allons illustré quelques résultats obtenu par notre suite de tests :



Conclusion :

Nous avons conçu et réalisé un outil de test et de vérification des Api Rest toute en implémentant le principe de Fuzzing. Pour ce faire, nous avons divisé le processus de développement de notre outil en qutres parties. La première représente l'utilisation de "swagger parser" afin de lire la spécification de l'Api à tester. La deuxième à travers le patron de conception "Factory", nous a permis de générer des données de manière aléatoire. La troisième, elle était consacrée à la mise en place du système de requetage. Quand à la quatrième, elle était consacrée aux tests en vu de détecter des bugs.