# Rapport V&V

## Fuzzing Swagger

2016-2017

Amel MESSADENE - Amina BOUSSALIA

# Introduction

Pour le TP de l'UE vérification et validation, nous avons choisi de travailler sur le fuzzing swagger. Donc, swagger est un metamodel de service REST. Pour programmer ça, nous avons pris pour exemple à tester l'api swagger "petstore" présente sur le site en json.

# Solution

Afin de tester une API REST définies à l'aide de swagger, nous avons développé un fuzzer pour valider les propriétés génériques de l'API développée.

La première étape du projet est de parser le fichier swagger.json de l'api petstore de swagger. Avec la fonction init(), nous avons chargé ce fichier puis parcouru tous les paths ou nous avons récupéré la clé et le path. Le clé correspond au modèle dans le fichier json, on en retrouve trois dans notre exemple (pet, store, user). Le path représente url de chaque méthode.

La deuxième étape du parsing est de récupérer la liste des opérations de chaque path, et donc récupérer plus concrètement les méthodes rest. Chaque méthode est représentée par un nom, des paramètres et des réponses. et c'est ces derniers qu'ils faut vérifier.

Pour tester, nous envoyons plusieurs requêtes à l'aide de la fonction fuzzer afin de faire apparaître des erreurs. La méthode checkThat de la classe ErrorCollector permet de comparer la méthode et le code retourné par fuzzer avec le code attendu du parsing. La comparaison se fait en ajoutant un échec si les deux valeurs ne se correspondent pas. Nous avons choisi d'utiliser la classe ErrorCollector car elle permet à l'exécution d'un test de se poursuivre même après avoir trouvé une erreur dans une méthode. Dans notre cas, on peut avoir plusieurs codes attendus par le parsing, et donc la comparaison se fait avec tous les codes.

La fonction fuzzer est celle qui envoie des requêtes paramétrées et retourne le résultat à tester. Elle prend en paramètres le path, la méthode et sa liste de paramètres. L'objectif de la fonction est d'envoyer à chaque fois un random d'entier ou de chaîne de caractères sur les trois parties de la méthode (id, name, status).

L'envoie des requêtes se fait à l'aide d'une connexion http. Le résultat retourné est sauvegardé dans un buffeur et renvoyé ensuite pour être vérifié par la méthode checkThat.

# Evaluation

Le projet est codé avec java, et contient une classe test “TestSwagger” en JUnit pour pouvoir utiliser Errorcollector et réaliser et exécuter des tests.

# Discussion

Le parsing fonctionne parfaitement, nous avons réussi à tester certaines méthodes, d’autres ne retourne un code d’erreur 405.

LA difficulté était de comprendre le fonctionnement et savoir envoyer des requêtes HTTP avec des données randoms, savoir choisir la meilleure solution parmis plusieurs proposées dans la documentation sur le net.

# Conclusion

Nous avons choisi de travailler sur un sujet pour tester les API REST. ce sujet est intéressant car nous utilisons souvent ces API mais nous avons encore pas su comment les tester. Pour améliorer notre projet, nous pouvons proposer de tester plus en détails le body d’une méthode par exemple.