Bonjour, je vais vous présenter mon fuzzer et les différentes techniques de testing utilisées tout au long du projet.

Tout d’abord, voici comment se déroulera ma présentation. Je vais commencer par une brève explication de ce en quoi le problem set consiste en un premier temps, je vais ensuite vous montrer une partie de mon code et vous expliquer comment mon fuzzer fonctionne, puis je vais vous montrer la façon dont j’ai testé le programme et enfin énumérer les différents problèmes que j’ai rencontré tout au long du cours.

Pour commencer, ce problem set consiste en la création d’un fuzzer. Un fuzzer c’est un programme qui a pour objectif de tester un autre programme. D’abord nous mettons un fichier en entrée, ce fichier sera ensuite corrompu et on met ce fichier en entrée du programme dont on souhaite tester le comportement. De ce que j’ai compris, le testing devrait se faire à partir des diverses techniques vues dans le cours.

Je vais maintenant vous parler de mon code. Il s’agit tout simplement du code présent dans la vidéo du problem set que j’ai adapté à divers programmes que j’ai installé dans mon PC. Dans la vidéo, les fichiers en entrée sont des pdf mais j’ai choisi de créer un fuzzer qui accepte toute sorte de fichier en entrée. De plus, le fait d’avoir mis des fichiers tels que des images ou des vidéos est intéressant pour voir le rendu du fuzzing.

Au départ, il y a 2 modes d’exécution. Le premier mode consiste en l’ouverture d’un fichier dont l’adresse est donnée en entrée, l’autre mode consistant au fuzzing d’un fichier au hasard parmi le seed de 10 fichiers que vous pouvez voir dans la variable test\_files. En fonction de l’extension du fichier, le programme choisira ensuite un programme au hasard pour l’ouverture qui aura lieu après le fuzzing. Le fuzzing correspond tout simplement au code de Charlie Miller, c’est le même que celui qui a été présenté dans la vidéo. Le fichier de sortie sera ensuite ouvert dans le programme qui a été choisi au hasard puis devrait se fermer normalement. Il est souvent arrivé que le programme ne se referme pas ou qu’il y ait eu crash.

Je vais maintenant vous montrer comment j’ai testé le fuzzer. Vous avez pu constater que j’ai stocké toutes les phases du code dans un fichier de log et que ces phases s’affichent également dans le terminal. Le fichier de log permet de garder une trace de chaque événement qui a eu lieu. Je suis aussi passé par du random testing, le fuzzer est lui-même une sorte de random testeur mais pour tester le fuzzer j’ai décidé de faire du random testing sur ce dernier également. Cela permet d’effectuer un plus grand nombre de tests de manière plus ou moins impartiale (du sens ou moi qui suis le programmeur ne puisse pas décider que mon programme fonctionne à tous les coups). Pendant le testing j’avais dans un premier temps songé à passer par d’autres types de testings mais ils se sont révélés non pertinents dans le cas ici présent. Le code coverage est impossible du fait de la randomisation du fuzzer, les black-box et white-box testing n’est pas faisable étant donné que la notion de pré et post condition n’a pas de sens ici, j’ai fait tester le code par d’autres étudiants qui m’ont donné des conseils pour l’améliorer. J’avais dans un premier temps fait du stress testing mais les conséquences étaient mauvaises pour mon PC.

Pour finir, je vais vous parler des divers problèmes que j’ai rencontrés pendant le cours. J’ai d’abord voulu créer des statistiques pour voir le taux de crash durant une expérience de tests mais compte tenu de l’implémentation de mon code ça a été très difficile et je n’y suis pas parvenu finalement. Un autre problème majeur était la fois où j’ai effectué un stress testing, ce qui a rendu l’explorateur de fichiers inutilisable (même après redémarrage du PC), j’ai eu de grosses difficultés dans l’implémentation du logging (certains messages apparaissaient bizarrement car c’était pas encodé en UTF-8 et je n’ai pas trouvé le moyen de le faire), pendant le fuzzing, des programmes ne se fermaient pas et je me retrouvais souvent avec une vingtaine d’instances du même programme ouverts simultanément et enfin, j’ai du me restreindre à ne faire que du random testing du fait de l’implémentation de mon code, ce qui fait que je ne suis même pas certain que le fuzzer fonctionne correctement.