PROBLEM SET 4: FUZZER

JONAH VERLY

PLAN DE PRÉSENTATION

Introduction	• Brève explication de ce en quoi consiste le problem set.
Code	• Explication du fonctionnement de mon code.
Testing	• Explication de ma stratégie de testing
Problèmes	• Les différents problèmes rencontrés lors du codage/testing

INTRODUCTION

- En quoi consiste ce problem – set ?
- → Créer un fuzzer (programme qui permet de tester un logiciel par l'ouverture de fichiers corrompus)
- → Effectuer des tests sur le fuzzer à travers différentes stratégies vues dans le cours

CODE

- Modifier le code source d'un fichier lambda
- Ouvrir ce dernier avec un programme choisi au hasard
- N'importe quel type de fichier est accepté

```
#INITIALISATION

print('Debut de l\'étape de fuzzing ...')
logging.debug('Debut de l\'etape de fuzzing ...')

if not manual: #seed contenant tous les fichiers dans le cas ou l'utilisateur ne rentre pas l'adresse
    test_files = [
        'Test_1.pdf', 'Test_2.pdf',
        'Test_1.jpg', 'Test_2.jpg',
        'Test_1.png', 'Test_2.png',
        'Test_1.mp4', 'Test_2.mp4',
        'Test_1.txt', 'Test_2.txt']
    selected_file = random.choice(test_files)
    selected_file_address = 'problem_set_4' + '\\' + 'fichiers_test'+'\\' + random.choice(test_files)
else:
    selected_file_address = str(input('Donner l\'adresse du fichier à fuzz:')) #adresse du fichier donnée par l'utilisateur


file_name = selected_file_address.split(".")
extension = file_name[(len(file_name) - 1)] #extension pour fichier de sortie


#choix du programme adequat pour ovrir le fichier de sortie
if extension in ['mp4', 'mov', 'avi', 'wmv', 'mp3']:
    selected_program = random.choice(['C:\Program Files\VideoLAN\VLC\\vlc.exe', 'C:\Program Files (x86)\GRETECH\GOMPlayer\GOM.exe'])
elif extension in ['jpg', 'jpeg', 'png', 'gif']:
    selected_program = random.choice(['C:\Program Files (x86)\Windows Live\Photo Gallery\WLXPhotoGallery.exe', 'C:\WINDOWS\system32\mspaint.exe'])
elif extension == 'pdf':
    selected_program = random.choice(['C:\Program Files (x86)\PDFsam Basic\pdfsam.exe', 'C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\AcroRd32.exe'])
elif extension == 'txt':
    selected_program = 'C:\Program Files (x86)\AbiSuite2\AbiWord\\bin\AbiWord.exe'
else:
    print('Impossible de fuzzer ce fichier')
    logging.debug('Impossible de fuzzer ce fichier')
    return

program_name = selected_program.split("\\")
selected_program_name = program_name[(len(program_name) - 1)]

print('Fuzzing du fichier %s avec le logiciel %s en cours ...'%(selected_file, selected_program_name))
logging.debug('Fuzzing du fichier %s avec le logiciel %s en cours ...'%(selected_file, selected_program_name))
```

PHASE DE FUZZING

```
#phase de fuzzing (Code de Charlie Miller)

for i in range(numwrites):
    rbytes = random.randrange(256) #byte aleatoire
    readnum = random.randrange(len(buffer)) #indice aléatoire dans le buffer
    buffer[readnum] = rbytes #remplacement du byte present

print('Fin de l\'étape de fuzzing ...')
logging.debug('Fin de l\'etape de fuzzing ...')
```

FICHIER OUTPUT

```
#fichier de sortie
output_file = "problem_set_4\\fuzz_output." + extension #fichier de sortie

print('Ouverture du fichier de sortie en cours ...')
logging.debug('Ouverture du fichier de sortie en cours ...')
try:
    open(output_file, 'wb').write(buffer)
except PermissionError:
    return

#lire le fichier de sortie
process = subprocess.Popen([selected_program, output_file])

#mettre fin au processus de lecture du fichier

print('Fermeture du fichier de sortie en cours ...')
logging.debug('Fermeture du fichier de sortie en cours ...')

time.sleep(1)
crashed = process.poll()
if not crashed:
    process.terminate
else:
    print('Un crash a eu lieu ...')
    logging.debug('Un crash a eu lieu ...')
```

TESTING

- Utilisation d'un fichier de log pour garder une trace de l'exécution du programme
- Random testing

```
def random_tester(max_num):
    """Random testing for the fuzzer

    Parameter:
    ----------
    max_num: max number of tests (should be > 0) (int)

    """
    assert max_num > 0

    num_test = random.randrange(max_num) #nombre total de tests durant le testing

    print('Il y aura un total de %d tests ...\n'%(num_test))
    logging.debug('Il y aura un total de %d tests ...\n'%(num_test))

    for i in range(num_test): #tests de fuzzing
        print('________Début du Fuzzing n°%d________'%(i + 1))
        logging.debug('________Debut du Fuzzing numero %d________'%(i + 1))
        fuzz_mode = str(input('Voulez vous fuzzer un fichier en particulier? (y/n)')) #avec fichier en particulier
        if fuzz_mode in['y', 'yes', 'o', 'oui']:
            logging.debug('Fuzz d\'un fichier choisi par l\'utilisateur')
            file_fuzzer(True)

        else: #avec fichier random
            logging.debug('Fuzz d\'un fichier aleatoire')
            file_fuzzer(False)

        print('_________Fin du Fuzzing n°%d_________\n\n'%(i + 1))
        logging.debug('_________Fin du Fuzzing numero %d_________\n\n'%(i + 1))

    print('Tous les tests ont été fait ...\n')
    logging.debug('Tous les tests ont ete fait ...\n')
```

TESTING

- Utilisation d'un fichier de log pour garder une trace de l'exécution du programme
- Random testing

```
from os import remove

#suppression des fichiers générés si possible

print('Suppression des fichiers générés en cours ...\n')
logging.debug('Suppression des fichiers generes en cours ...\n')

if exists('problem_set_4\\fuzz_output.pdf'):
    try:
        remove('problem_set_4\\fuzz_output.pdf')
        print('Le fichier de sortie pdf a été supprimé ...')
        logging.debug('Le fichier de sortie pdf a ete supprime ...')

    except PermissionError:
        print('Erreur lors de la suppression du fichier de sortie pdf ...')
        logging.debug('Erreur lors de la suppression du fichier de sortie pdf ...')

if exists('problem_set_4\\fuzz_output.jpg'):
    try:
        remove('problem_set_4\\fuzz_output.jpg')
        print('Le fichier de sortie jpg a été supprimé ...')
        logging.debug('Le fichier de sortie jpg a ete supprime ...')

    except PermissionError:
        print('Erreur lors de la suppression du fichier de sortie jpg ...')
        logging.debug('Erreur lors de la suppression du fichier de sortie jpg ...')

if exists('problem_set_4\\fuzz_output.png'):
    try:
        remove('problem_set_4\\fuzz_output.png')
        print('Le fichier de sortie png a été supprimé ...')
        logging.debug('Le fichier de sortie png a ete supprime ...')

    except PermissionError:
        print('Erreur lors de la suppression du fichier de sortie png ...')
        logging.debug('Erreur lors de la suppression du fichier de sortie png ...')
```

PROBLÈMES RENCONTRÉS

- Le code présent dans la vidéo ne fonctionnait pas.
- Certains programmes ne s’ouvraient pas.
- Encodage du fichier de log.
- Choix de la stratégie de testing.
- Ouverture des programmes pendant exécution du fuzzer.

MERCI POUR VOTRE ATTENTION