UE121 – Introduction à la programmation

HAUTE ÉCOLE DE NAMUR-LIÈGE-LUXEMBOURG

Technologie de l'informatique – bloc 1

Sécurité des systèmes – bloc 1

Examen

# Consignes

Au plus tard à la fin de l’heure prévue d’examen, **vous devez avoir déposé un dossier contenant votre projet complet (examen compris) correctement nommé (voir consigne partie 1) dans le dossier correspondant à votre session-examen** dans le dossier correspondant à votre local dans le dossier Exa\_HomeDir.  
Par exemple pour Bob Smith en IR1A passant l’examen au L001 sur la session 42, un dossier « TIR121\_IR1A\_SmithBob » se trouvera dans « Exa\_HomeDir > L001 > IE-L001-42 ».  
Sans un dossier correctement nommé au bon endroit et dans les temps, l’examen sera considéré comme non présenté.

Lors de l’examen, vous pouvez vous munir d’une clé USB contenant votre projet de base (et rien d’autre), d’une (et une seule) feuille A4 de notes manuscrites (recto-verso) ainsi que de feuilles de brouillon (vierges).

# Grille de cotation

1. Pytest projet de base :

* KO : note finale = 0/20 (le reste n’est pas corrigé)
* OK : note finale = 3/20 + points examen

1. Pytest examen :
   * + - Q1 OK : +8
       - Q2 OK : +5

* Q3 OK : +3

1. Pylint tout :
   * Fautes graves : -10 la première, -15 si plus d’une (le reste n’est pas corrigé)

* Fautes moyennes : -1/ faute
* Fautes légères : de -0 à -1.5/ type de faute en fonction du nombre
* Minimum 2/20
* Aucune faute : +1

# Fichier exam.py

Dans le dossier de votre projet, à la racine, créez un fichier « exam.py ». Il se trouvera donc au même niveau que « main.py ». **Répondez aux questions de l’examen dans ce fichier « exam.py ».**

Concernant les tests unitaires de l’examen (qui testent donc « exam.py »), ils se trouvent dans le dossier « fichiers utiles ».

Pour ce qui est de pylint, vous disposez (dans « fichiers utiles ») d’un fichier de config qui permet, avec la commande « pylint ./ », de tester tous vos fichiers .py et d’ignorer les fautes que nous ne pénaliserons pas.

# Syslog et auth.log

Pour le projet de base, nous utilisions un fichier de log appelé « syslog ». Pour l’examen, nous allons en utiliser un second : « auth.log ». Sa structure est identique à syslog.

Pour rappel, voici à quoi peut ressembler une ligne de ce fichier :

Oct 28 10:31:27 kali systemd[1]: Starting Login Service...

Plusieurs informations sont présentes sur cette ligne :

* La date (avec le mois écrit en 3 lettres (en anglais) et le jour en 2 chiffres)
* L’heure,
* Le hostname,
* Le programme,
* L’id du processus (pas toujours présent)
* Un message explicatif (peut être composé de sous messages séparés par des « : »)

Pour la suite, nous parlerons parfois de ligne de log « bien formée », il s’agit d’une ligne respectant cette syntaxe.

# Questions

## Les id des programmes qui ouvrent une session

Le fichier « auth.log » prend entre autres note des sessions qui sont ouvertes via le message « session opened for user XXX » (pour un utilisateur XXX).

Ecrivez la fonction process\_id\_which\_open\_sessions qui permet de retourner la liste (sans doublon) des processus qui on ouverts une session (quel que soit l’utilisateur) dans un laps de temps défini..

def process\_id\_which\_open\_sessions(logs, date\_min, date\_max):

    """ Pre :

    - logs est une liste où chaque élément est une ligne de log bien formée

    - date\_min et \_max sont des str représentant une date (et heure) au format "MM:JJ HH:MM:SS".

- date\_min est plus petit que date\_max

        Post :

    - retourne une liste (sans doublon) des process id qui ouvrent une session dans le temps défini par date\_min et date\_max.

    """

## Les processus d’un programme générant une erreur

Écrivez la fonction error\_process qui prend en paramètre une liste de logs bien formés, un programme, et une date. La fonction retourne la liste (sans doublon) des processus de ce programme qui ont générés un message d’erreur pendant la date passée en paramètre.

def error\_process(logs, program, date):

    """ Pre :

       - logs est une liste où chaque élément est une ligne de log bien formée

- program est le programme recherché

- date est une date au format MM:JJ (exemple: 24 Octobre = 10:24)

      Post :

       - retourne une liste (sans doublon) des processus qui ont émis au moins un message d’erreur.

    """

## 3. Les sessions de kali

Dans le fichier « auth.log », le programme « systemd-logind » prend note des nouvelles sessions des différents utilisateurs.

D’après le fichier « auth.log », combien de session DIFFERENTES a eu l'utilisateur kali ?

Aidez-vous des différentes fonctions que vous avez écrites pour le projet pour répondre et écrivez la fonction « bonus » qui retourne simplement cette réponse.

def sessions\_kali(logs):

    """

Pre:

- logs est une liste où chaque élément est une ligne de log bien formée

Post:

- retourne le nombre de sessions différentes ouvertes par l'utilisateur kali

"""