Python et Unix : TP 5

Préambule :

L'objectif de ce TP est d'implémenter des outils basiques testant la sécurité.
La lecture des pages (pydoc.org) se rapportant à crypt, getpass et pwd est fortement recommandée

Exercice 1 (6 points)

A l'aide de getpass.getpass(), pwd.getpwnam(), et crypt.crypt() écrivez une application qui demande le login, le mot de passe (sans echo à l'écran) et affiche "Succès" si le mot de passe est bien le mot de passe UNIX de l'utilisateur.

Exercice 2 (10 points)

A l'aide de pwd.getpwall(), trouvez :

- La liste de tous les sels (deux premiers caractères du mot de passe crypté) utilisés. On éliminera les mots de passe protégés (x ou *)
- Construisez un hash associant à chaque password crypté le nom de l'utilisateur qui lui correspond

Sur http://www.unil.ch/ling/frgut.html téléchargez un dictionnaire (dicos/*.dico).

Les expressions régulières sont un outil d'analyse puissant que nous n'avons pas eu le temps de traiter. Elles permmettent de reconnaître des *pattern* dans des chaines de caractères.
La ligne suivante :

```
mot = re.sub("\/\w+","",lignedico)
```

permet de nettoyer les annotations dans les fichiers .dico. Elle substitue à la chaine "/" suivie d'un certain nombre non nul de caractères de mot [a-zA-Z_0-9] par la chaine vide "".

Pour chaque ligne du dictionnaire (utilisez f.xreadlines() ou f est obtenu par open()), si le mot après nettoyage à une longueur au moins égale au minimum de longueur du système, crypter le mot en utilisant tous les sels répertoriés, tester la présence du mot crypté dans le hash des password cryptés connus.

Afficher le login et le password des comptes ainsi détectés.

Executez votre script avec *time* (man 1) pour mesurer ses performances. Essayez de changer votre mot de passe pour voir combien de temps il faut pour vous craquer.

Exercice 3 (4 points)

Ecrivez un script prenant en argument un nom de machine (ou son IP), qui essaye d'ouvrir une connection TCP sur tous les ports de 0 à 1024 sur la machine cible, et qui produit en sortie la liste des numéros de ports de services activés sur la machine cible.

Utilisez un try/except pour rattraper les erreurs produites par masocket.connect() quand le service n'est pas disponible.

Le script suivant lit le fichier /etc/services :

```
services = {}
fserv = open("/etc/services")
for line in fserv.xreadlines() :
	if re.match("^\#",line) :
		continue
	else :
		match = re.match("(\w+)\s+(\d+)\/(\w+)\s+(.+)",line)
		if not match :
			continue
		if (match.group(3) == "tcp") :
		   services[match.group(2)] = match.group(1) + " : " + match.group(4)
for k in services :
	print k, ":", services[k]
```

Utilisez ce script pour attribuer des noms aux ports que vous avez découvert.