Opgave SMP-Python 20-21 tentamen

BELANGRIJK

Voor deze opgave krijg je 8 uur de tijd om deze te maken. Je mag gebruik maken van het internet om zaken om te zoeken. Je moet wel zelf (alleen) de opgave maken. Als je de opgave inlevert, verklaar je ook dat dit je **eigen** werk is. Bij twijfel zullen we een nader onderzoek doen en zul je je programma mondeling moeten toelichten. We stellen dan vast of je de stof zelf beheerst. Dit kan dan alsnog tot een onvoldoende leiden. Mochten we constateren dat je (delen) hebt overgenomen van andere bronnen, dan doen we een melding bij de examencommissie voor het plegen van plagiaat, met alle gevolgen van dien.

Opdracht

In deze opgave analyseer je een system-logfile [1].

Context

Zodra je een computer verbindt met het internet zul je zien dat al snel pogingen worden gedaan om verbinding te maken met je computer via SSH. Kwaadwillige partijen zullen proberen je wachtwoord te raden om zo in te loggen op je computer. Daarom is het verstandig om je ssh-logfiles te analyseren en maatregelen te nemen. In dit tentamen moet je de logfile analyseren en een aantal zaken opmerken.

Uitleg

In de log-file zien we een aantal inlogpogingen met ssh. Sommige zijn geslaagd, andere falen.

We zijn geïnteresseerd in een aantal zaken.

Voor het basis cijfer (5.5) moeten de volgende zaken worden gedaan:

1. We willen graag weten hoeveel inlog-pogingen per dag slagen en falen.
2. We willen graag weten hoeveel inlog-pogingen per ip-nr slagen en falen.
3. We willen weten welke sessies langer dan 10 uur hebben geduurd.

Voor een hoger cijfer (+1.5) moeten de volgende zaken gevonden worden:

1. Vind de aanvallen van de culprits en vermeld deze.

Er zijn ook nog extra testgevallen voor de eerst drie opgaven. Deze werken met grotere systemlog-files en met ingewikkeldere gevallen. Hiermee kun je extra punten (+0.5) verdienen. Verder wordt het cijfer (+1.0) bepaald door de flake8-linter van CodeGrade. Het skeleton-programma is zo geschreven dat voor die skeleton-code geen punt aftrek plaats vind.

En de laatste punten (+1.5) worden door de docent gegeven op basis van de gekozen oplossingen en programmeer stijl.

De basis vormt een logfile `system.log`. Deze heeft de vorm:

```
dt;prog;pid;ip;user;state
2020-07-13 14:10:02.332880;sshd;584;192.168.179.3;root;started
2020-07-13 15:11:03.332880;sshd;584;192.168.179.3;root;finished
2020-07-13 14:09:06.332880;sshd;140;192.168.179.9;frans;failed
```

Het programma SSH maakt twee soorten log regels.

1. Geslaagde inlog-poging
   Als het juiste wachtwoord gebruikt wordt, wordt een ssh verbinding gemaakt, er worden dan twee logregels gemaakt. Een als de sessie begonnen wordt met status `started` en een als de sessie beëindigd is met status `finished`. Er zijn dan twee log-regels zoals:

```
2020-07-13 14:10:02.332880;sshd;584;192.168.179.3;root;started
2020-07-13 15:11:03.332880;sshd;584;192.168.179.3;root;finished
```

1. Gefaalde inlog-poging
   Als niet het juiste wachtwoord gebruikt wordt, wordt geen ssh verbinding gemaakt, wel wordt deze poging gelogd. De status is dan `failed`. Er is dan een log-regel zoals:

```
2020-07-13 14:09:06.332880;sshd;140;192.168.179.9;frans;failed
```

readLog

Om de file in te lezen is een methode `readlog` alvast gegeven. Deze leest de logfile in en geeft een lijst van dicts terug. Deze lijkt kwa return-waarde dus erg op de `scan` functie van `opg2a`.

def readLog(fname):

De log-file is een semicolon-separated (puntkomma-gescheiden) file met een header. Python heeft een standaard library dit het makkelijk maakt om een dergelijke file in te lezen `csv`. (https://docs.python.org/3/library/csv.html) Je kunt de file inlezen met behulp van `csv.DictReader`. De reeds geïmplementeerde functie ziet er als volgt uit:

```
import csv  # gebruik de python-library csv

def readLog(fname):
    with open('system.log', 'r') as fp:
        lst = list(csv.DictReader(fp, delimter=';'))
    return lst
```

Deze methode leest de file `fname` in en retourneert een lijst van dicts. Dit kun je testen door op de command-line de volgende actie

```
$ python3
>>> from opgt import readLog

>>> # Lees de lijst of dicts uit 'system_mini.log'
>>> lst = readLog('system_mini.log')
>>> len(lst)
3

>>> # Eerste element van de list
>>> lst[0]
{'dt': '2020-07-13 14:10:02.332880', 'prog': 'sshd', 'pid': '584',
 'ip': '192.168.179.3', 'user': 'root', 'state': 'started'}
```

Op deze wijze wordt een lijst van dicts verkregen.

Datetime

Je moet die SSH-verbindingen vinden die langer dan 10 uur duren. Gezien de normale werkdag 8 uur duurt, is het raar dat deze verbindingen zolang duren. Vandaar dat we geïnteresseerd zijn in deze verbindingen.

Python heeft een module om te rekenen aan tijdstippen (datetime https://docs.python.org/3/library/datetime.html) De belangrijkste objecten zijn:

datetime.datetime

Dit object representeert een tijdstip. De belangrijke routines zijn `strftime` om een tijdstip naar een string te formatteren. En `strptime` om een string naar een `datetime` object te converteren Het object `datetime` heeft attributen zoals `hour` , `minute` , `second` .

datetime.timedelta

Dit object representeert een tijdsduur. De attributen zijn `days` en `seconds` . Je krijgt een object `timedelta` door twee `datetime` objecten van elkaar af te trekken. op deze manier kun je het aantal

seconden en dagen tussen twee tijdstippen berekenen.

Voorbeeld van het gebruik van `datetime` en `timedelta`

```
>>> import datetime

>>> # Maak van een string een `datetime` object
>>> strDtS = '2020-07-13 14:10:02.332880'
>>> dtS = datetime.datetime.strptime(strDtS, '%Y-%m-%d %H:%M:%S.%f)
>>> print(dtS, type(dtS))
2020-07-13 14:10:02 <class 'datetime.datetime'>

>>> # Maak van een string een datetime-object
>>> strDtF = '2020-07-13 15:11:03.332880'
>>> dtF = datetime.datetime.strptime(strDtF, '%Y-%m-%d %H:%M:%S.%f')
>>> print(dtF, type(dtF))
2020-07-13 15:11:03 <class 'datetime.datetime'>

>>> # Het verschil timedelta tussen de twee timestamps
>>> td = dtF-dtS
>>> print(td, type(td))
1:01:01 <class 'datetime.timedelta'>
>>> print(td.seconds, td.days)
3661 0
```

Mogelijk is '%f' niet geïmplementeerd in jou versie van python3.
Gebruik dan:

```
>>> dtS = datetime.datetime.strptime(strDtS[:19], '%Y-%m-%d %H:%M:%S')
```

findCulprit

De **culprit** is een persoon die probeert in te breken in het systeem. Deze culprit zal dus acties achter laten in de logfile. Deze acties vallen op in vergelijking met normale inlog acties. Probeer alvast te bedenken wat het patroon is van een normale inlog actie. Hoe zouden de acties van de culprit hiervan afwijken?

Methoden

Het doel van het programma is om verschillende informatie uit deze lijst van log-regels te halen, die interessant zijn voor een Cyber Security Specialist. Je moet vier methoden# implementeren:

- dayCount
- ipCount
- longSessions

- findCulprit

In de tentamen-opgave is dit nader uitgewerkt. Deze krijg je op de tentamendag

Beoordeling:

Het programma wordt (gedeeltelijk automatisch) beoordeeld op de volgende punten:

- 1.00: Voor een programma dat uitvoerbaar is. Dwz `python3 opgt -f system.log` geeft geen python-foutmeldingen.
- 1.50: countByDay: de juiste output wordt gegeven
- 1.50: countByIp: de juiste output wordt gegeven.
- 1.50: longSessions: de juiste output wordt gegeven.
- 1.50: findCulprit: de juiste output wordt gegeven.
- 0.50: Lastig: Extra tests voor de eerste drie gevallen.
- 1.00: Beoordeling door linter.
- 1.50: Beoordeling door de docent mbt de gekozen oplossing en codeerstijl.

De beoordeling op CodeGrade gebruikt grotere system.log files om een en ander te testen. Het kan dus zijn dat je programma lokaal met de kleine system log-files werkt, maar niet met de test-set op CodeGrade. Kijk dus goed wat voor foutmelding CodeGrade geeft bij uploaden van het programma en de foutmeldingn bij de test.

Inleveren Voltijd

Lever uiterlijk ma 14 jun 2020 18:00 in op CodeGrade (opgt). Alleen het programma `opgt.py` moet worden ingeleverd. De tests voor de eerste 5.5 punten worden direct in CodeGrade uitgevoerd en geven aan of je een voldoende hebt. Deze 5.5 punten krijg je alleen als alle test voor deze methoden slagen. Een fout (hoe klein ook) betekent geen punten. Je kunt tussen 14:00 en 18:00 veelvuldig je programma uploaden om een en ander te testen. Doe het niet op het laatste moment, dan kan het druk zijn en krijg je niet direct resultaat, dus test je programma lokaal en upload het na 14:00 uur.

De overige punten worden voor ma 21 juni bekend gemaakt. Je weet echter direct al of je geslaagd bent of niet (5.5 of meer).

Je kunt van 14:00 eens per 10 minuten een programma inleveren. Het laatste programma wordt beoordeeld. Programma's na 18:00 ingeleverd kunnen we alleen in uitzonderlijke omstandigheden alsnog beoordelen.

Inleveren Deeltijd

Lever uiterlijk ma 14 jun 2020 24:00 in op CodeGrade (opgt). Alleen het programma `opgt.py` moet worden ingeleverd. De tests voor de eerste 5.5 punten worden direct in CodeGrade uitgevoerd en geven aan of je een voldoende hebt. Deze 5.5 punten krijg je alleen als alle test voor deze methoden slagen. Een fout (hoe klein ook) betekent geen punten. Je kunt tussen 20:00 en 24:00 veelvuldig je programma uploaden om een en ander te testen. Doe het niet op het laatste moment, dan kan het druk zijn en krijg je niet direct resultaat, dus test je programma lokaal en upload het na 20:00 uur.

De overige punten worden voor ma 21 juni bekend gemaakt. Je weet echter direct al of je geslaagd bent of niet (5.5 of meer).

Je kunt van 20:00 eens per 10 minuten een programma inleveren. Het laatste programma wordt beoordeeld. Programma's na 24:00 ingeleverd kunnen we alleen in uitzonderlijke omstandigheden alsnog beoordelen.

Hertentamen

Ik hoop dat een ieder slaagt voor het tentamen, maar er is een tweede mogelijkheid op ma 28 juni 10:00-18:00 (voltijd) 16:00-24:00 (deelltijd).

Veel Succes

Frans Schippers f.h.schippers@hva.nl ModuleCoordinator SMP

---

1. Dit is een vereenvoudigde versie van een systeem-logfile. ↩