Test practic Bitdefender - Cloud Security Analyst

Contex

Soluțiile de tip cloud (IAS – *Infrastructure As Service*) sunt folosite de către tot mai multe companii în detrimentul echipamentelor fizice: hard-disk-urile se înlocuiesc cu spațiu de stocare în cloud, procesoarele și memoria RAM se transformă în mașini virtuale etc...

Infrastructurile găzduite în cloud ajung adeseori vaste și complexe, devenind dificil de administrat. Ca urmare a acestei complexități, pot să apară probleme de configurare a diverselor resurse, probleme care conduc la nevoie de produse (programe) care să-i poată asigura pe administratori că infrastructura este sigură și respectă prevederile unor standarde definite în acest scop.

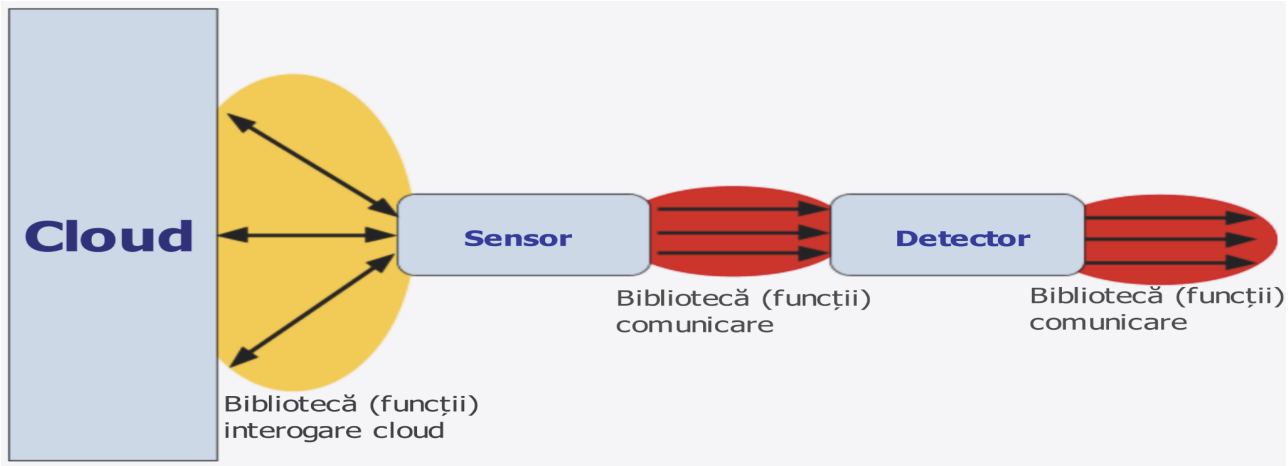
Problema propusă va încerca să răspundă acestei nevoi prin colectarea datelor (prin intermediul unui *senzor*) care descriu modul în care sunt configurate resursele dintr-un cloud și apoi prelucrarea acestor date, de către un *detector*, în scopul de a găsi potențiale greșeli de configurare (cazuri de nerespectare ale unor cerințe).

Componenta *senzor* are la dispoziție o bibliotecă de funcții (adică un modul Python), funcții prin intermediul cărora se poate efectua colectarea datelor iar componenta *detector* va efectua procesarea efectivă a acestora. Separarea colectării de codul care face prelucrarea ne garantează o arhitectura decuplată unde cele două funcționalități distincte pot fi implementate separat și fără interdependențe.

Senzorul și detectorul au la dispoziție și o bibliotecă de comunicare prin intermediul căreia pot trimite și primi date. Funcțiile din această bibliotecă vor fi folosite atât de către senzor cât și de către detector:

- senzorul trimite datele culese din cloud înspre detector
- detectorul face apeluri atat pentru a prelua datele de la senzor cât și pentru a trimite mai departe rezultatele sale finale (către orice alt program ar fi interesat să le folosească).

Contextul de execuție este ilustrat în figura următoare:



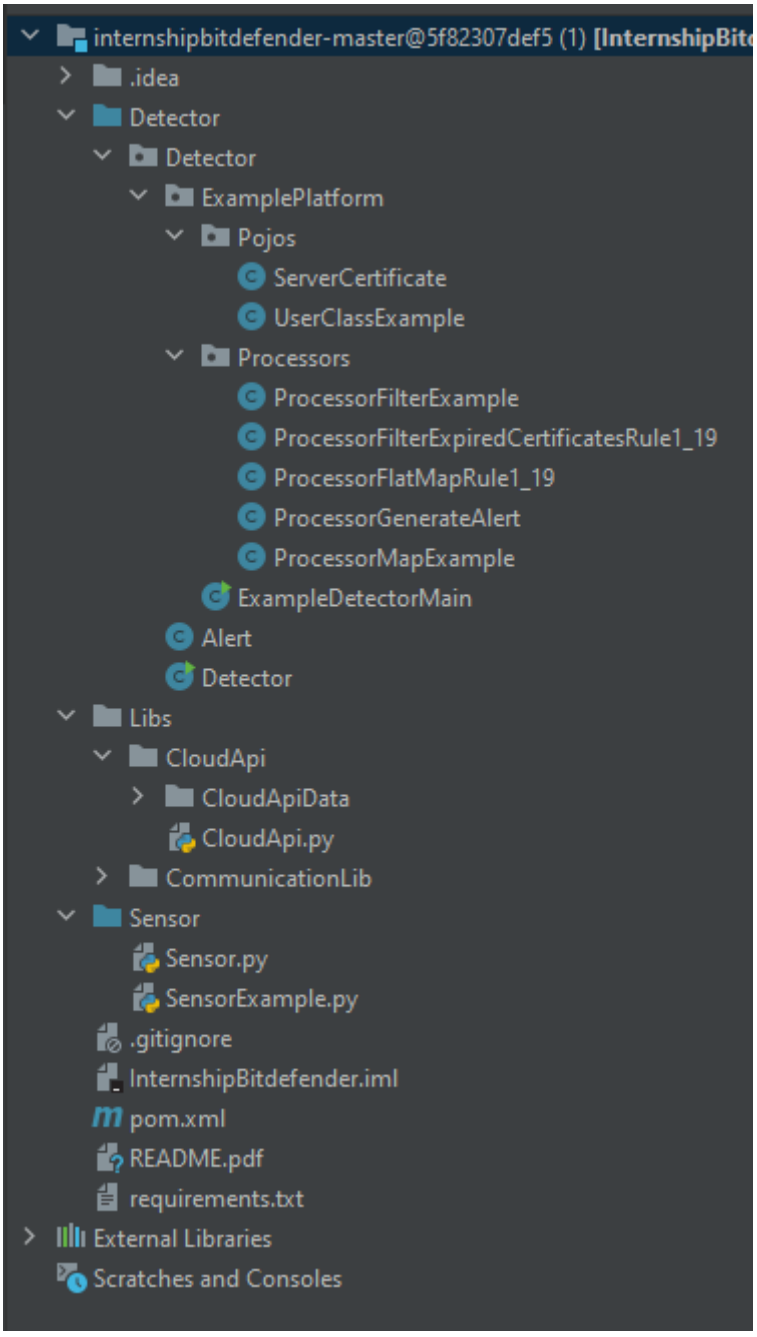
Notă: biblioteca de interogare a cloud-ului (Cloud API), cea de comunicare (CommunicationLib) și

cloud-ul în sine reprezintă în resursele de lucru o simulare a unor astfel de servicii existente în lumea reală: cloud = AWS, interogare cloud = Boto3 și comunicare = Kafka.

Resurse

Pentru început îți recomandăm să folosești IDE-ul **IntelliJ** versiunea Community. În continuare, explicațiile legate de modul de configurare și rulare sunt exclusive acestui IDE.

În arhiva atașată la acest email se regăsește proiectul structurat astfel:



README.pdf: Este recomandat să începi dezvoltarea prin a citi acest document. Aici gasești explicați pașii standard de configurare, pluginurile recomandate și modul de rulare a exemplului.

Sensor: În acest director se află un model de lucru și explicațiile aferente scrierii unui Sensor. Tot în acest director va trebui să implementezi cerințele care țin de componenta Sensor.

Citeste și ruleaza **SensorExample.py**

În privința rezolvării problemei, poți alege să extinzi exemplul curent sau să vii cu o implementare proprie.

Detector: În acest director se află un model de lucru și explicațiile aferente scrierii unui Detector. Tot în acest director va trebui să implementezi cerințele care țin de componenta Detector.

Poți alege să păstrezi structura exemplificata în subdirectorul **ExamplePlatform** sau să o înlocuiești cu o structură proprie.

Citește și rulează **ExamplePlatform > ExampleDetectorMain.py**

Notă**:** Este important ca rezultatele detectorului să fie de forma **Alert.java**, din acest motiv clasa se află în exteriorul folderului ExamplePlatform**.**

Libs: Acest director contine bibliotecile folosite în scrierea Detector-ilor si Sensor-ilor. Este recomandat să citești descrierile API-urilor din fișierul **CloudApi.py**

Cerințe

a.) Ensure IAM password policy requires minimum length of 14 or greater (Rule 1.8)

- gather the information about the password policy (Sensor)
- check for minimum password length and generate an alert in case of non-conformity (Detector)

Example:

```
"PasswordPolicy": {
    …
    "MinimumPasswordLength": 19,
    …
}
```

The above example should NOT generate any alerts as it respects the rule, *MinimumPasswordLength* being greater than 14.

b.) Ensure IAM Users Receive Permissions Only Through Groups (Rule 1.15)

- gather the list of ALL users (there should be 6) (Sensor)
- gather the list of attached policies per user, if ANY found generate an alert (Sensor, Detector)
- gather the list of user policies per user, if ANY found generate an alert (Sensor, Detector)

c.) Ensure multi-factor authentication (MFA) is enabled for all IAM users that have a console password (Rule 1.10)

- gather the FULL credential report (Sensor)
- check for every user that the password usage and the MFA are enabled, if one of them is not, generate an alert (Detector)

d.) Ensure there is only one active access key available for any single IAM user (Rule 1.13)

- gather the list of ALL users (there should be 6) (Sensor)
- gather the list of access keys (Sensor)
- check that for every user there is ONLY ONE access key that is ACTIVE, generate an alert otherwise (Detector)

Example:

```
"AccessKeyMetadata": [
    …
    {
        "UserName": "littleboy",
        "AccessKeyId": "AAIFRAAATLXAAABWM11N",
        "Status": "Active",
        "CreateDate": "2021-01-21 00:05:06+00:00"
    },
    {
        "UserName": "littleboy",
        "AccessKeyId": "ATIARDAAVTAADAAMAAKD2",
        "Status": "Active",
        "CreateDate": "2022-03-22 02:21:12+00:00"
    }
```

…

],

The above example should generate an alert because there are multiple active access keys for the same user (we have more than one key for *UserName = "littleboy")*.

e.) Ensure IAM policies that allow full "*:*" administrative privileges are not attached (Rule 1.16)

- gather the list of policies (Sensor)
- gather the version information for per policy (Sensor)
- carefully check for the following scenario: an attached policy with the following statement configuration should NOT exists: "Effect" = "Allow" and "Action" = "*" and "Resource": "*", if there is one generate an alert (Detector)

Indicii:

- fiecare cerință se bazează pe câte o regulă reală, pentru detalii adiționale, opțional, se poate consulta și “CIS_Amazon_Web_Services_Foundations_Benchmark_v1.4.0.pdf”
- - colectarea datelor ("gather") trebuie făcută în Detector (py) cu următoarele funcții: list_users, list_access_keys, get_account_password_policy, generate_credential_report, get_credential_report, list_attached_user_policies, list_user_policies, list_policies, list_entities_for_policy, list_server_certificates și get_policy_version - documentația lor se poate consulta în cod, ca și commentarii sau urmărind link-urile către documentația oficială
- verificarea condițiilor ("check", "if ... ") trebuie făcută în Sensor (java) prin analizarea câmpurilor din datele(json format) trimise de Detector
- fiecare abatere ("alert") este necesar să fie mapată/asociată clasei Alert(.java) și să fie trimisă în "CommunicationLib"

Constrângeri

- Codul modulului detector.py trebuie să fie scris în limbajul Python (cel puțin versiunea 3.0)
- Soluția trebuie să fie funcțională folosind forma originală a clasei *Alert* precum și conținutul original din directorul *Libs*
- Comunicarea dintre senzor și detector se face exclusiv prin datele trimise prin biblioteca de comunicare și nu este permis ca senzorul să transmită altceva decât datele obținute prin apelul funcțiilor de la punctul 2 (senzorul nu face detecții sau alte operații complexe, el doar culege și trimite datele mai departe).

Trimiterea rezolvării și termenul limită

Soluția pentru problemă trebuie să o trimiți prin mail până **cel mai târziu marți (17.05.2022) la 23:59**, ca reply la acest mail. Pentru aceasta, se vor atașa individual, la mailul de reply, atât fișierul *sensor.py* conținând codul python pentru senzor cât și fișierul *detector.*java care implementează componenta de detector în limbajul Java*.*

Fișierele tale conținând implementarea trebuie să fie funcționale odată ce vor fi copiate în directorul de lucru (în locația/structura prevăzută în arhivă).

Pentru punctarea soluției se va lua în considerare atât funcționarea corectă a codului cât și atribute non-funcționale precum: eficiența (memorie folosită și timp de procesare), organizarea logică a codului (impărtirea în funcții care au sens), atenția la cazuri speciale, securitatea și robustețea implementării (ar trebui să fie greu de crashat).

Dacă alegi să folosești mai multe fișiere/clase distincte, va fi necesar atunci să trimiți soluția sub forma unei arhive conținând întreg directorul (iar dacă serviciul de email refuză livrarea atașamentului, va fi nevoie să urci pe ceva serviciu de file sharing și să trimiți în mail-ul de reply link-ul oferit de către respectivul site).

Dacă întâmpini dificultăți cu resursele pe care ți le-am trimis sau sunt neclarități / întrebări în privința cerințelor poți să mă contactezi prin telefon, pe adresa de mail sau poți să apelezi și la colegul meu cnacu@bitdefender.com (Cristi).

Succes și mult spor!

Lóránd