BITDEFENDER CHALLENGE - MY FIRST ANTIVIRUS - PCLP1

Deadline hard: **19.01.2023 23:55**

CUPRINS

Changelog

- **28.12.2022**
  - **NU** accesați niciun URL dintre cele primite ca date de intrare.
  - Checker și Makefile updatat. Descărcați din nou arhiva checkerului.
  - Pentru a putea primi punctaj pentru implementarea C, este obligatoriu să rezolvați toate erorile apărute în urma rulării **valgrind**.
  - Soluțiile pot fi submise pe VMChecker, selectați *Programarea Calculatoarelor și Limbaje de Programare (CA, CB, CD)* aici.
  - Leaderboardul este acum disponibil aici.

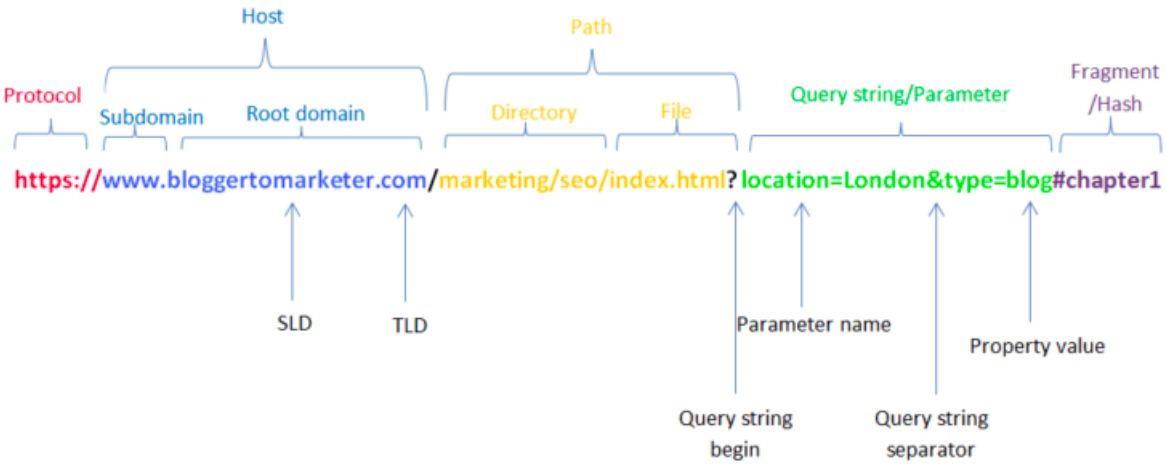
1 ENUNȚ

Un antivirus este un program care se instalează pe un calculator și care te ajută să te ferești de descărcarea și pornirea unor programe malițioase pe calculatorul tău și te projectează împotriva atacurilor din exterior.
În cadrul acestui challenge vei avea de implementat primul (sau poate nu) tău antivirus minimal. El va avea două componente **individuale**, care vor fi punctate **individual**. Mai multe detalii găsiți în secțiunile următoare.

1.1 Task 1 - Detecția URL-urilor malițioase

Prima componentă a antivirusului vostru va fi cea care detectează dacă un URL este benign sau nu. URL-urile **non-benigne** sau **malițioase** sunt create în mod voit de către cineva rău intenționat, cu scopul de obține informații personale de la utilizatori (phishing), de a distribui malware, de a folosi resursele browserului tău pentru a mina criptomonede (cryptomining), etc.
În cadrul acestei teme ne vom concentra pe doar URL-urile de tip **phishing**, **malware** și **benigne**.

Pentru rezolvarea acestui task veți avea nevoie să studiați formatul unui URL. Vedeți în imaginea de mai jos structura detaliată al unui URL.



1.1.1 *Date de intrare*

ATENȚIE! Nu intrați pe niciunul dintre URL-urile primite ca date de intrare.

Fiindcă nu putem spune garantat că un URL este malițios decât accesându-l și văzând ce se întâmplă după ce dăm click, acuratețea implementării voastre nu poate fi 100%. Scopul este însă să obțineți o acurtatețe cât mai mare, fapt ce înseamnă că predicțiile anitvirusului vostru sunt foarte bune.

Veți primi un set de date de intrare public pentru a putea testa euristicile pe care le găsiți, însă punctajul final va fi calculat pe baza unui set de date privat. El se află sub forma următoarei ierarhii:

```
data
    traffic
       [...]
    urls
       domains_database
       urls_classes
       urls.in
```

Fișierul **urls.in** este fișier de intrare pe care îl veți citi din cadrul programului vostru. El conține pe fiecare linie câte un URL care poate fi de tip **phishing**, **malware** sau **benign**.

Fișierul **urls_classes** este fișierul ajutător, nu îl veți citi în cadrul programului vostru. El conține pe fiecare linie tipul URL-ului asociat din fișierului **urls.in**, deci unul dintre cuvintele **phishing**, **malware** sau **benign**. Spre exemplu, dacă pe linia 17 din fișierul **urls_classes** se află cuvântul **phishing**, înseamnă că al 17-lea URL din **urls.in** este de tip **phishing**. Acest fișier vă ajută pe voi să vă dați seama cum arată un URL malițios, astfel încât să creați euristici cât mai bune (a.k.a. să obțineți acuratețe cât mai bună).

1.1.2 Date de ieșire

Fișierul de ieșire se va numit **urls-predictions.out** și va conține verdictul pentru toate URL-urile primite ca date de intrare. Verdictul va fi **0** (zero) dacă URL-ul este benign și **1** (unu) dacă este malițios (adică ori **phishing** ori **malware**). Spre exemplu, dacă al 17-lea URL din **urls.in** este unul malițios, linia 17 din **urls.out** va conține cifra **1**.

Găsiți în fișierul **tasks/my_av/tests/00-my_av/00-my_av.ref** un exemplu de fișier de ieșire pentru setul de date public primit.

1.1.3 Euristici din partea noastră

Pentru a începe implementarea acestei componente, recomandăm să porniți de la euristicile de mai jos. Ele garantează că într-o oarecare măsură predicția va fi bună, însă nu cea mai bună. Pentru a o îmbunătăți, trebuie să veniți voi cu alte idei prin care să dați verdictul despre un URL. Vă puteți folosi de setul de date public pentru a găsi noi euristici. Euristicile din partea noastră sunt:

- Un mod foarte eficient și des folosit pentru a detecta URL-uri malițioase este folosirea unei baze de date cu URL-uri sau domenii ce au fost descoperite anterior ca fiind malițioase. Astfel, în fisierul **urls_database** găsiți o lista de domenii ce sunt cunoscute ca fiind malițioase (**malware** sau **phishing**). Astfel, pentru orice URL pentru care domeniul se află în baza de date, veți considera că este un URL malițios (predicție **1**).
- Malware-ul este de cele mai multe ori este un fișier binar (executabil). Astfel, dacă fișierul care este accesat în URL este un fișier cu extensia **.exe**, atunci este posibil ca acesta să fie malware. De exemplu, URL-ul **hmbwgroup.com/wp-includes/images/media/files/elb.exe** este posibil să fie de tip malware.

Implementând euristicile de mai sus, ar trebui să obțineți un scor F1 de aproximativ **0.67**, ce vă va aduce punctajul maxim pentru acest task. Mai multe detalii despre scorul F1 și despre checker, găsiți in secțiunea 1.3.

ATENȚIE! NU deschideți niciun URL întrucât unele sunt malițioase. Faceți doar analiză pe URL în sine.

1.2 Task 2 - Malicious Network Traffic Detection

A doua componentă a antivirusului vostru este cea care va fi capabilă să detecteze dacă traficul de internet interceptat este benign sau nu. Traficul **non-benign** este traficul capturat în timpului unui atac. Atacurile pot fi de tip Bruteforce, Crypto Mining, Denial of Service, etc.
În cadrul acestei teme ne vom concentra pe doar traficul de tip **cryptominer**, **bruteforce** și **benign**.

1.2.1 Date de intrare

La fel ca la Taskul 1, nici despre o captură de trafic nu putem spune cu siguranță că face parte dintr-un atac. Scopul este să prezicem corect în cat mai multe cazuri.

Veți primi un set de date de intrare public pentru a putea testa euristicile pe care le găsiți, însă punctajul final se va da pe baza unui set de date privat. El se află sub forma următoarei ierarhii:

```
data
    traffic
       traffic_classes
       traffic.in
    urls
       [...]
```

Fișierul **traffic.in** este fișierul de intrare de tip CSV pe care îl veți citi din cadrul programului vostru. Fiecare linie din el descrie traficul interceptat pe o anumită perioadă de timp. Fiecare linie se încadrează la trafic de tip **cryptominer**, **bruteforce** și **benign**. Excepție de la cele de mai sus este prima linie a fișier care conține antetul fișierului CSV, care descrie fiecare coloană.

Fișierul **traffic_classes** este un fișier ajutător, nu îl veți citi din cadrul programului vostru. El conține pe fiecare linie tipul traficului asociat din fișierului **traffic.in**, deci unul dintre cuvintele **cryptominer**, **bruteforce** sau **benign**. Spre exemplu, dacă pe linia 17 din fișierul **traffic_classes** se află cuvântul **bruteforce**, înseamnă că a 17-a intrare din CSV-ul cu traffic (linia 18 din **traffic.in**) este de tip **bruteforce**. Acest fișier vă ajută pe voi să învățați cum arată traficul malițios astfel încât să creați euristici cât mai bune (a.k.a. să obțineți acuratețe cât mai bună).

1.2.2 Date de ieșire

Fișierul de ieșire de va numit **traffic-predictions.out** și va conține verdictul pentru toate URL-urile primite ca date de intrare. Verdictul va fi **0** dacă traficul este benign și **1** dacă este malițios (adică ori **bruteforce** ori **cryptominer**). Spre exemplu, dacă a 17-a intrare din **traffic.in** este unul malițios, linia 17 din **traffic-predictions.out** va conține cifra **1**.

Găsiți în fișierul **tasks/my_av/tests/00-my_av/00-my_av.ref** un exemplu de fișier de ieșire pentru setul de date public primit.

1.2.3 Euristici din partea noastră

Pentru a începe implementarea acestei componente, recomandăm să porniți de la euristicile de mai jos. Ele garantează că într-o oarecare măsură predicția va fi bună, însă nu cea mai bună. Pentru a o îmbunătăți trebuie să veniți voi cu alte idei prin care să dați verdictul despre o captură de trafic. Vă puteți folosi de setul de date public pentru a găsi noi euristici. Euristicile din partea noastră sunt:

- Deoarece într-un atac de tip Bruteforce, atacatorul încearcă în mod repetat parole în încercarea de a o ghici pe cea corecta, în general **flow_duration** este mai mare decât în mod normal. Vom considera ca fiind bruteforce traficul ce are **flow_duration** mai mare de o secundă.
- Totuși, exista multe situatii în care și pentru un trafic normal, durata să fie mai mare, în care se fac multe schimburi de pachete cu un payload gol (veți afla în anii mai mari mai multe detalii). Astfel, ne interesează trafic în care payload-ul mediu (**flow_pkts_payload.avg**) este diferit de **0** (zero).

Considerând cele 2 euristici anterioare, veți obține un **scor F1** de aproximativ **0.68**, ce vă va aduce punctajul maxim pentru acest task.

1.3 Evaluare detecții

Modalitatea de evaluare a performanței modelului pe care o vom folosi este scorul F1. Pentru a putea verifica usor detectorul, am pus la dispoziție și un checker, care va calcula acest scor pentru implementarea voastră. Chiar dacă nu este nevoie să implementați voi acest scor, recomandăm să vă documentați puțin despre el. Pe scurt, cu cât detectorul prezice corect (prezice ca fiind malițioase URL-urile malițioase și ca nefiind malițioase cele care nu sunt malițioase) mai multe URL-uri, cu atât scorul o sa fie mai mare.

2 NOTARE

Punctajul temei pe checker este de 100 puncte indiferent de limbaj, distribuit astfel:

- Task1 1: 50p vor fi date proportional cu scorul F1 obținut după formula

$$min(max(50 * (F1 - 0.5)/(0.67 - 0.5), 0), 50)$$

  astfel:

  - F1 $\leqslant$ 0.5 va fi punctat cu 0p
  - F1 $\geqslant$ 0.67 va fi punctat cu 50p
  - Un scor în (0.5, 0.67] obține un punctaj proporțional (de exemplu, pentru F1 = 0.6 vor fi acordate 29.4/50p).

- Task 2: 50p vor fi date proportional cu scorul F1 obținut după formula

$$min(max(50 * (F1 - 0.5)/(0.68 - 0.5), 0), 50)$$

  astfel:

  - F1 $\leqslant$ 0.5 va fi punctat cu 0p
  - F1 $\geqslant$ 0.68 va fi punctat cu 50p
  - Un scor în (0.5, 0.68] obține un punctaj proporțional (de exemplu, pentru F1 = 0.6 vor fi acordate 27.7/50p).

Programul vostru va trebui să rezolve la o singură rulare atât task-ul 1, cât și task-ul 2. Cu alte cuvinte, executabilul generat va rezolva mai întâi răspunsul pentru task01 și apoi răspunsul pentru task02.

ATENȚIE! Această temă este bonus, punctajul nefiind inclus în nota de pe parcurs. Cu toate acestea, tema valorează 0.5p bonus pentru implementarea în C **și** 0.2p bonus pentru implementarea în Python, care se adaugă la nota PCLP din regulamentele seriilor CA, respectiv CB+CD, după obținerea punctajului minim pe parcurs și promovarea examenului (acest punctaj nu poate fi folosit pentru a promova materia, însă poate ajuta la obținerea unei note finale mai mari).

3 CHECKER

Pentru testarea locala, aveți disponibil un set de teste publice (de o dificultate similară) în arhiva temei.

```
.
├── check
├── check_utils
├── data
│   ├── traffic
│   │   ├── traffic_classes
│   │   └── traffic.in
│   └── urls
│       ├── domains_database
│       ├── urls_classes
│       └── urls.in
├── install.sh
├── Makefile
├── README.md
└── tasks
    ├── config.json
    └── my_av
        ├── grader
        └── tests
            ├── 00-my_av
            │   ├── 00-my_av.in
            │   └── 00-my_av.ref
            └── 01-my_av
                ├── 01-my_av.in
                └── 01-my_av.ref

```

• Rulați comenzile **./install.sh** și **./check** pentru a instala / rula checkerul. Mai multe detalii în README.md.

• Pentru evaluarea pe checker a implementării Python, fișierul **my_av.py** trebuie să aibă shebang setat pe prima linie (**#!/usr/bin/python3**) și fie executabil (**chmod +x my_av.py**).

• Limita de timp pe checker este de 60 de secunde, atât pentru C, cât și pentru Python.

4 PRECIZĂRI

• Orice încercare de rezolvare frauduloasă a temei va fi notată cu 0p și nu va putea fi considerată pentru stabilirea câștigătorilor. Metode frauduloase presupun (dar nu sunt limitate la): hardcodare (stabilirea manuală pentru URL / scanare), modificarea fișierelor de referință, rularea de alte procese care au scop rezolvarea temei, modificarea checker-ului, atacul asupra infrastructurii pentru a prelua codul altor colegi sau testele private, copierea din orice sursă.
• Pentru detalii puteți să vă uitați și peste regulile generale de trimitere a temelor (regulile generale CA și reguli generale CB-CD).
• O temă care **NU** compilează va fi punctată cu 0. O temă care **NU** trece niciun test pe vmchecker va fi punctată cu 0.
• Vor exista atât teste publice, cât și teste private pentru fiecare problemă în parte. Punctajul pe temă și scorul final **se calculează exclusiv pe baza rulării programului pe testele private**.
• Pe lângă codul sursă, este **obligatorie** trimiterea unui fișier README, în care să descrieți soluția voastră și să descrieți eventualele euristici implementate, diferite de cele menționate în enunț.

5 FORMAT ARHIVĂ SUBMISĂ

Arhiva cu rezolvarea temei trebuie să fie **.zip**, având un nume de forma **Grupa_NumePrenume_AV.zip** și va conține:

- Fișierul/fișierele sursă. Sursa pentru C în care se află funcția *main()* trebuie denumită obligatoriu **my_av.c**, iar pentru Python 3 trebuie denumită obligatoriu **my_av.py**.
- Fișierul **Makefile** (nu **makefile.txt**, nu **makefile** sau orice variațiune) care va conține **obligatoriu** următoarele reguli:
  - **build**, care va compila sursele C și va obține executabilul numit **my_av** pentru C. Dacă ați implementat doar în Python 3 această regulă nu face ceva, dar prezența sa este obligatorie!
  - **clean**, care șterge executabilele și fișierele obiect generate.
- Fișierul **README** (nu **README.txt**, nu **readme.txt**, nu **Readme** sau orice variațiune)
- **Opțional** Alte fișiere create de voi care sunt necesare pentru ca rezolvarea voastră să funcționeze.

RECOMANDARE Puteți generați arhiva folosind comanda **make pack**. Ea se va numi **solution.zip** în urma rulării acestei comenzi. Redenumiți ulterior arhiva în **Grupa_NumePrenume_AV.zip**.

ATENȚIE! Toate fișierele menționate trebuie să fie în **rădăcina** arhivei.

ATENȚIE! Tema va fi compilată și testată **DOAR într-un mediu Linux**.

ATENȚIE! Numele regulilor și a surselor trebuie să fie exact cele de mai sus. Absența sau denumirea diferită a acestora va avea drept consecință obținerea a 0 puncte pe temă.

ATENȚIE! NU este permisă compilarea cu opțiuni de optimizare a codului (O1, O2, etc.).

ATENȚIE! Orice nerespectare a restricțiilor duce la pierderea punctajului (după regulile de mai sus).