Łukasz Matusewicz

DOKUMENT TECHNICZNY PROJEKTU
Threat Inteligence Feed for SRX -TIFS
PRZEDMIOT: PROGRAMOWANIE ZAAWANSOWANIE

Link do repozytorium:

1. Tematyka projektu

Tematem projektu jest platforma pobierająca, i analizująca źródła Thereat Inteligence z różnego pochodzenia (bazy GeoIP, black listy, listy reputacji itp.)
Można wydzielić moduły pracujące jako klient:

1. Klient serwisów:
   A) denyIP (geoIP)
   B) blacklisty feodotracker.abuse.ch (Serwery C&C, itp.)
   C) Alienvoult reputation list
   D) inne znalezione zaufane blacklisty odświeżane cyklicznie.
2. Analizator – usługa parsująca zebrane dane w celu stworzenia black list. Typy analiz:
   A) zsumaryzuje adresy IP w przypadku wykluczenia większych bloków adresowych.
3. moduł whiteListy i sanityzacji:
   A) wykluczający ze stworzonych list adresy zdefiniowane przez administratora.
   B) Sanityzacja optymalizująca żeby wyeliminować powtórzenia adresów IP, posortowanie stworzonych list (wymagane przez firewall). Listy będą nadpisywane i skorelowane z ustawieniami Firewalla.

Moduł pracujący jako serwer

1. Serwer HTTP serwujący pliki pobierane przez Juniper SRX do tworzenia blacklist/alertów.
2. Moduł zapisywania list:
   1. Zapis do płaskiego pliku tekstowego.
   2. Tworzenie archiwów gzip.

2. Diagram przypadków użycia

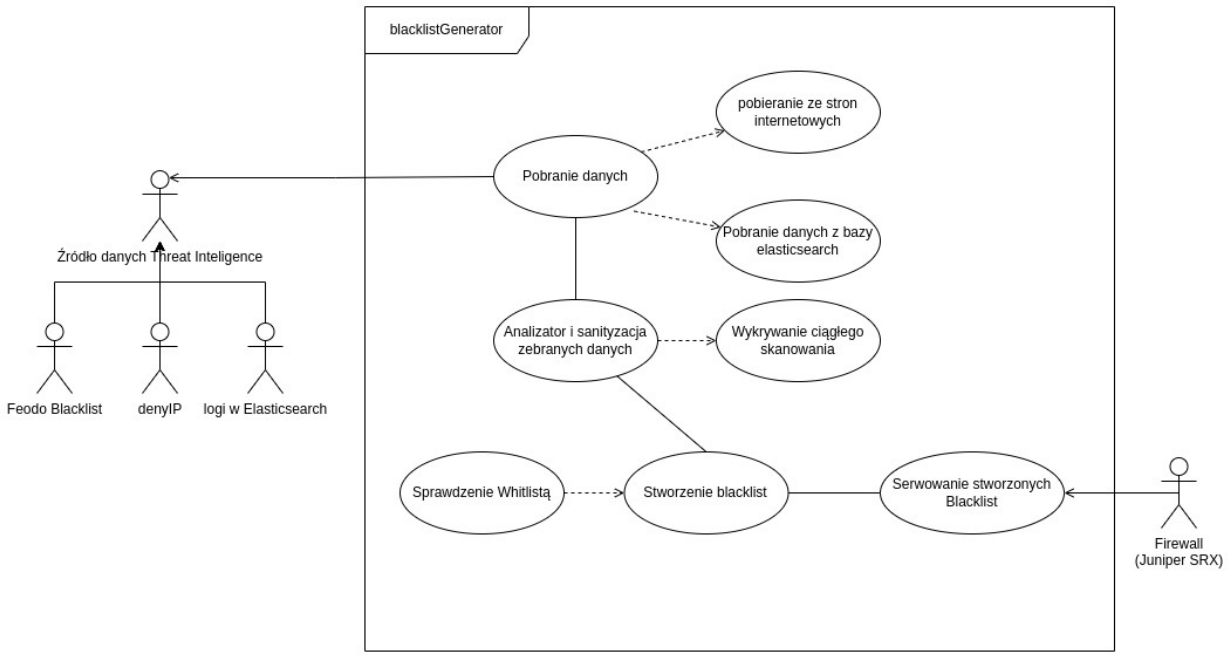

Figure 1: Diagram przypadków użycia

3. Diagram klas

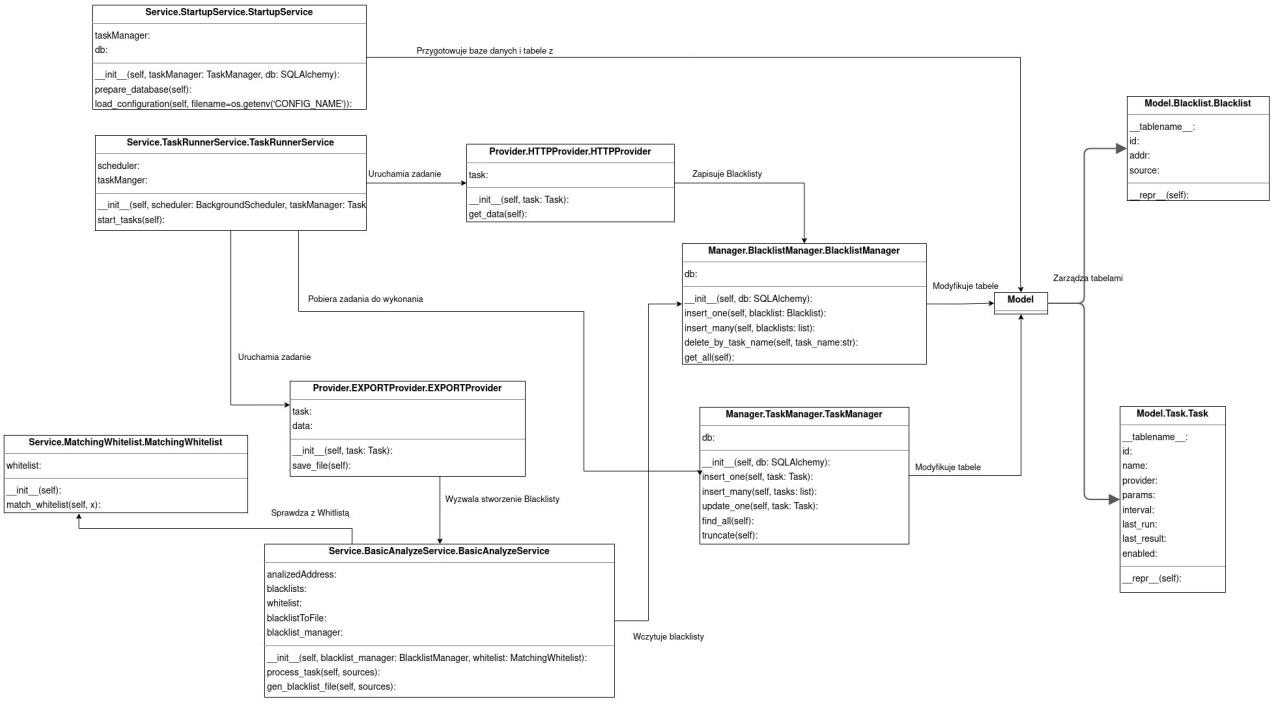

Figure 2: Diagram Klas UML

4. Opis techniczny projektu

Projekt ma powstać z wykorzystaniem języka Python i frameworka Flask, ze względu na łatwość implementacji w środowisku linuxowym, usługa ma finalnie zostać skonteneryzowana.

Projekt ze względu na typowo linuxowo usługową naturę konfiguruje się przy pomocy plików. Jedne plik .env wykorzystywany do konfiguracji bazy danych, whitelisty i wskazania nazwy pliku z konfiguracją zadań. Drugi z zadaniami w postaci pliku .yaml z eksportowaniem i pobieraniem blacklist, rozpoznawanych na podstawie providera. Eksport pozwala na wybór z których sparsowanych blacklist ma powstać dana lista. Jest to przydatne ze względu na różne przeznaczenie list, niektóre zawierają wskaźniki kompromitacji (IOC) które sprawdza się w ruchu wychodzącym i nie blokuje się tylko służą do wykrywania słabości i incydentów w sieci.

Przykład poniżej

Table 1: Przykład konfiguracji app_config.yaml

```yaml
otx_reputation_IOC:
  provider: "HTTP"
  params:
    url: "https://reputation.alienvault.com/reputation.generic"
  interval: 60
  enabled: 1
export-blacklist:
  provider: "EXPORT"
  params:
    file_name: "generated_blacklist"
    source_blacklists_names: ["feod_blacklist","otx_reputation_IOC"]
  interval: 60
  enabled: 1
```

Aktualnie jedyną metodą pobierania danych ze źródeł Threat Inteligence jest request do serwisów http z wykorzystaniem biblioteki requests. Tutaj problemem było różne formatowane plików (w niektórych były puste linie w innych komentarze itp.) Trzeba było obsłużyć takie wyjątki aby nie zaśmiecać bazy danych.

Przetrzymywanie adresów IP przeniesiono na bazę danych, tak aby był dostęp do danych i trzymane były w sanityzowany sposób (Postrgres SQL ma typ IP Cidr co parsuje dane) do interakcji z Bazą wykorzystano ORM (Object Relational Mapper) „SQLAlchemy” dzięki czemu interakcje z danymi były typowo obiektowe i modele baz danych również stały się klasami. Przysporzyło to wiele problemów w czasie

implementacji. W wykorzystaniu **Flask** potrzebna była dodatkowa biblioteka która trochę zmieniła semantykę dostępu do danych w stosunku do większości dostępnej dokumentacji.

W celu dostępu do pojedynczego obiektu danej klasy wykorzystano singleton w postaci frameworku **container** w której składuje się utworzone obiekty. Co daje dostęp do nich w całym projekcie bez potrzeby ich powielania.

Problematyczne okazało się parsowanie adresów IP, przez co wykorzystywane są aż dwie biblioteki i do sanityzacji i whitlistowania dane są mapowane z wykorzystaniem dwóch bibliotek pomimo tego, że obie dotyczą tego samego typu danych. Szczególnie przydatne metody to:

- **.overlaps()** z biblioteki **ipaddress** wbudowanej w pythona
- **cidr_merge()** z biblioteki **netaddr**

Aby zadania (taski z pobieraniem i tworzeniem blacklist) wykonywały się cyklicznie i równolegle zarazem zgodnie z zadanym przez konfigurację interwałem wykorzystano bibliotekę **apscheduler.schedulers.background import BackgroundScheduler** która dodaje zadania zgodnie z konfiguracją i wybranymi providerami.

Serwer serwuje pliki, na stronie głównej można znaleźć listę blacklist przygotowanych przez system.

Pliki są przygotowane w formie płaskiej i skompresowanej tak aby różne systemy mogły skorzystać z przygotowanych list, np. firewalle innych producentów takich jak Palo Alto czy Check Point.

Napotkany i rozwiązany problem: Podatność Path Traversal.

Opis:
Znaleziona i wykorzystana metoda do listowania i serwowania plików przy pomocy flaska posiadała podatność Path Traversal. Nie można jej wykorzystać z poziomu przeglądarki ze względu na to że normalizują ścieżkę przez co z „/../../../../../../../etc/passwd” wysyłają w requeście *„/etc/passwd”*

- generated_blacklist.gz
- generated_blacklist
- ioc_list.gz
- ioc_list

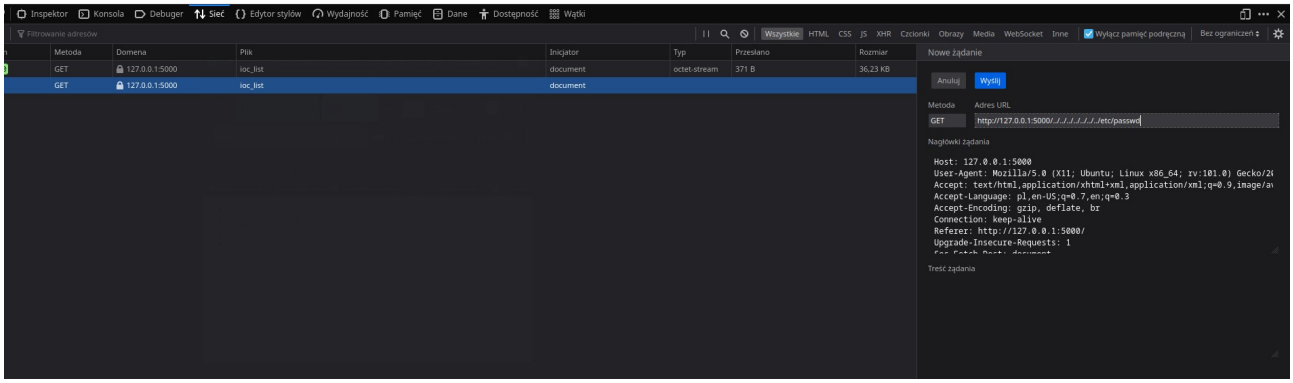

Figure 3: Nie udany atak Path Traversal

Ale gdy stworzymy zapytanie już uzyskamy dostęp:

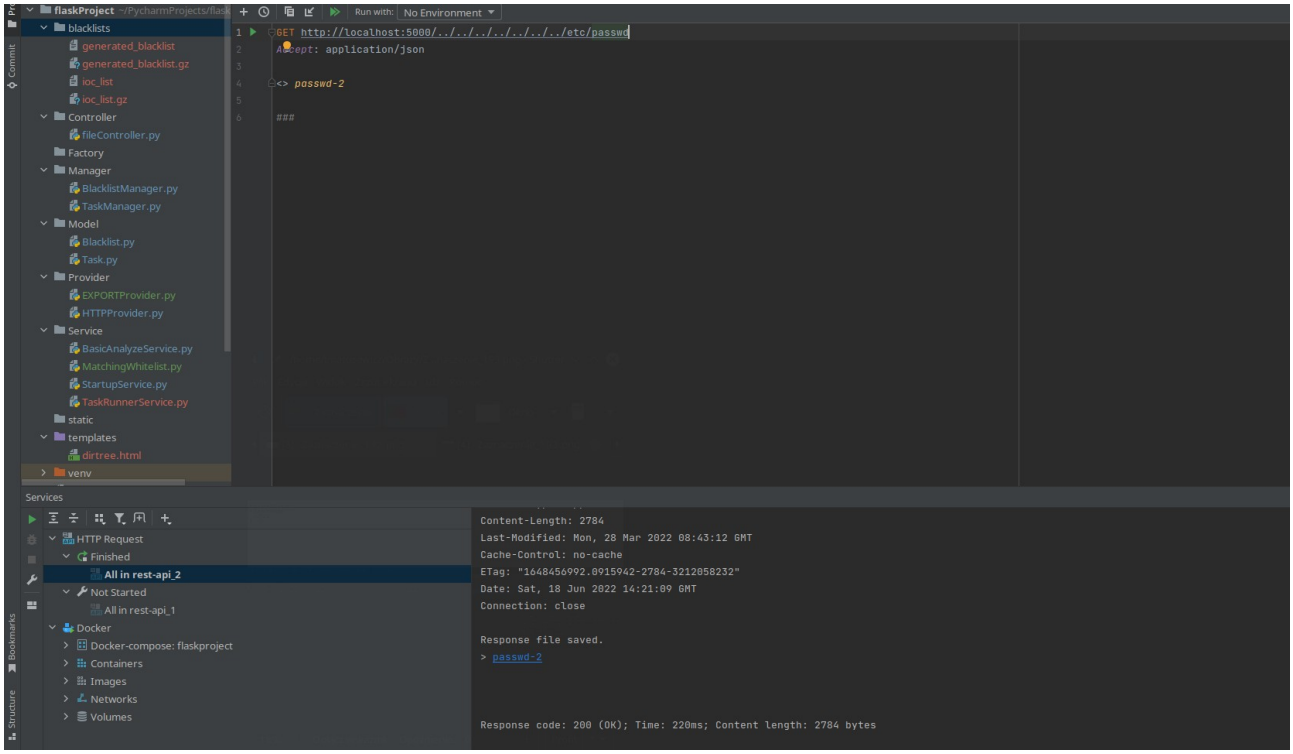

Figure 4: Udany atak Path Traversal

Stworzono zabezpieczenie w postaci parsowania i normalizacji ścieżki i sprawdzenie czy prowadzi do folderu w którym znajdują się blacklisty:

Table 2: Metoda zabezpieczenia przeciw Path Traversal

```
abs_path_to_check = os.path.normpath(abs_path)
if abs_path_to_check.startswith(os.path.normpath(os.path.abspath(BASE_DIR)
+os.sep)):
```

5. Potencjalne możliwe problemy i zagrożenia (do części technicznej)
Możliwe że trzymanie IP w słowniku nie będzie optymalne i trzeba będzie zastanowić się nad wykorzystaniem bazy danych lub po prostu tablicy. Różnież sanityzacja możliwe że będzie potrzebowała dodatkowej klasy do operacji sumaryzacji podsieci w celu optymalizacji blacklisty.

Oba potencjalne problemy okazały się realne i zostały odpowiednio zaadresowane.

6. Scenariusze testów

Testy jednostkowe planowano z wykorzystanie frameworku **pyTest**. Jednak ze względu na brak zasobów osobowych – jednoosobowa grupa, skupiono się na testach funkcjonalnych.

Test funkcjonalny został wykonany z wykorzystaniem Firewalla SRX 5400 o poniższej konfiguracji:

Table 3: Stworzenie listy feedów pobieranych przez SRXa w tym przypadku dla blacklisty generated_blacklist

```
set security dynamic-address feed-server blacklist_feed hostname geoip.apps.paas-dev.psnc.pl
set security dynamic-address feed-server blacklist_feed feed-name belarus path /generated_blacklist.gz
set security dynamic-address feed-server blacklist_feed feed-name belarus update-interval 3600
set security dynamic-address feed-server blacklist_feed feed-name belarus hold-interval 3700
```

Table 4: Stworzenie adresów wykorzystywanych przez politykę bezpieczeństwa

```
set security dynamic-address address-name generated_blacklist profile feed-name generated_blacklist
set security dynamic-address address-name generated_blacklist_denay profile feed-name
generated_blacklist
```

Table 5: Polityka bezpieczeństwa stworzona w oparciu o dynamic-address

```
set security policies from-zone  EXTERNAL-main to-zone Internal-Zones policy blacklist_policy
description "Testowa polityka zatrzymująca  LM"
set security policies from-zone  EXTERNAL-main to-zone Internal-Zones policy blacklist_policy match
source-address blacklist_feed_denay
set security policies from-zone  EXTERNAL-main to-zone Internal-Zones policy blacklist_policy match
source-address belarus_denay
set security policies from-zone  EXTERNAL-main to-zone Internal-Zones policy blacklist_policy match
destination-address sophora-169.man.poznan.pl
set security policies from-zone  EXTERNAL-main to-zone Internal-Zones policy blacklist_policy match
destination-address sophora-166.man.poznan.pl
set security policies from-zone  EXTERNAL-main to-zone Internal-Zones policy blacklist_policy match
application any
set security policies from-zone  EXTERNAL-main to-zone Internal-Zones policy blacklist_policy then
deny
set security policies from-zone  EXTERNAL-main to-zone Internal-Zones policy blacklist_policy then
log session-init
insert security policies from-zone EXTERNAL-main to-zone Internal-Zones policy blacklist_policy
before policy ALL-ACCEPT
```

Table 6: Sprawdzenie czy feed-serwer działa

```
show security dynamic-address summary

node0:
--------------------------------------------------------------------------


  Server Name                : blacklist_feed
    CA  Profile Name         : ---
    Hostname/IP              : geoip.apps.paas-dev.psnc.pl
    Update interval          : 3600
```

```
    Hold   interval           : 3700
    TLS Profile Name          : ---
    User        Name          : ---


    Feed Name                             : blacklist_feed
        Mapped dynamic address name       : blacklist_feed_denay
        CA  Profile Name                  : ---
        URL                               :
https://geoip.apps.paas-dev.psnc.pl/generated_blacklist.gz
        Feed update interval              : 3600  Feed hold interval :3700
        Total update                      : 18
        Total IPv4 entries                : 80
        Total IPv6 entries                : 0
        Total download   errors           : 0   Last occurence N/A
        Total db         errors           : 0   Last occurence N/A
        Total other      errors           : 0   Last occurence N/A
        Total ageout                      : 0   Last occurence N/A
        Next update time                  : Wed Mar  2 14:28:14 2022
        Next expire time                  : Thu Mar  3 13:28:14 2022
        Flags                             : 0x0
        Last update file size             : 508
        Last update IPv4 entries          : 80
        Last update IPv6 entries          : 0
        Last update   begin time          : Wed Jul  2 13:28:12 2022
        Last update   end   time          : Wed Jul  2 13:28:14 2022
        Last update   time cost(s)        : 2
        Last download begin time          : Wed Jul  2 13:28:12 2022
        Last download end   time          : Wed Jul  2 13:28:13 2022
        Last update   status              : 4
        Last download time cost(s)        : 1
```

Table 7: Sprawdzenie zawartości listy

```
show security dynamic-address address-name blacklist_feed_denay

node0:
--------------------------------------------------------------------------
No.     IP-start             IP-end               Feed             Address
1   2.56.24.0     2.56.27.255    blacklist_feed blacklist_feed_denay
2   2.56.88.0     2.56.91.255    blacklist_feed blacklist_feed_denay
3   2.56.112.0     2.56.115.255    blacklist_feed blacklist_feed_denay
4   2.56.124.0     2.56.127.255    blacklist_feed blacklist_feed_denay
5   2.56.136.0     2.56.139.255    blacklist_feed blacklist_feed_denay
```

7. Spis rysunków i tabel

Index of Tables

Table of Figures

8. Lista zmian w dokumencie

Rewizja	Imię i nazwisko	Opis
1	Łukasz Matusewicz	Stworzenie opisu projektu
2	Łukasz Matusewicz	Dodanie opisu w punktach 4 i 5
3	Łukasz Matusewicz	Modernizacja dokumentacji po developmencie