Kryminalistyka Cyfrowa (KRYCY)

Laboratorium 1: Blue Team Python

Analiza cyberzagrażeń na podstawie źródeł danych z hostów i sieci

Informacje organizacyjne

Semestr: 23Z
Oryginalny termin ogłoszenia: 2023-10-13
Termin oddania całego zadania: do 15 grudnia 2023 r. 23:59
Oddawanie rozwiązań:

- na koniec: formularz MS Forms z linkiem do OneDrive zespołu, a tam pliki
- bieżące przekazywanie rozwiązań poprzez wybrany system Git

Laboratorium 1 jest przeznaczone do realizacji w zespole 4-osobowym.

Liczba punktów do uzyskania: **12**

Historia

Zadaniem zespołu cyberbezpieczeństwa, którego jesteście członkami jest monitorowania sieci i hostów, zarządzanie właściwymi danymi w tych systemach dla detekcji cyberzagrożeń oraz sama detekcja. Wśród źródeł danych, do których macie dostęp warto wskazać:

- pliki PCAP
- pliki tekstowe txt z danymi o dowolnym formacie (np. syslog)
- pliki tekstowe w uniwersalnych formatach - JSON i XML
- pliki binarne z danymi formacie EVTX, właściwym dla logów

Zostaliście poproszeni o przygotowanie prototypu narzędzia w języku `Python`, które pozwoli na:

- zintegrowane zarządzanie źródłami danych z sieci i hostów
- zunifikowaną interakcję z systemem operacyjnym z poziomu jednej aplikacji w języku `Python`
- detekcję cyberzagrożeń wykorzystującą 3 różne sposoby:
  - wyrażenie regularne
  - reguły jako funkcje i operacje logiczne dostępne w języku `Python`
  - detekcja regułowa z wykorzystaniem znanego języka reguł SIGMA

Ponadto macie przedstawić możliwy przykład wybranej detekcji anomalii.

Wymagania dla rozwiązania zadania

Prototyp systemu EDR/XDR

Wymagania dotyczące szkieletu systemu

Analizator Cyberzagrożeń (aplikacja CLI)

`GEN.MGMT.1` Opracować aplikację w języku `Python` z interfejsem CLI, która pozwoli na realizację wskazanych wymagań. Moduł do tworzenia aplikacji CLI - `Click`
https://click.palletsprojects.com/en/8.0.x/

`GEN.MGMT.2` Aplikacja działa w trybie CLI z wypisywaniem akcji na to CLI oraz do pliku loga.

`GEN.MGMT.3` Aplikacja ma możliwość wskazania pojedynczych plików, folderu lub grupy folderów do przeszukania w poszukiwaniu plików, które mają być wykorzystane do analizy

`GEN.MGMT.4` Obsługiwane formaty wejściowe plików do analizy:

- pliki tekstowe w formatach `.txt` , `.xml` , `.json`
- pliki ze zrzutami ruchu PCAP `.pcap`
- pliki logów Sysmon Windows EVTX - `.evtx` , przetwarzane na format tekstowy JSON, XML lub innych tekstowy (do wyboru według własnego uznania)

`GEN.MGMT.5` Wprowadzić komunikację ze Zdalnym Kolektorem Zdarzeń.
`GEN.MGMT.5.1` Metodą komunikacji aplikacji głównej ze Zdalnym Kolektorem Zdarzeń alertów jest REST API.

Zdalny Kolektor Zdarzeń (Aplikacja CLI)

`GEN.LOG.1` Przygotować aplikację CLI opartą na `Click` do odbierania wiadomości po REST API związanych z wykrywanymi zdarzeniami.

`GEN.LOG.2` Aplikacja posiada dwa tryby działania:
`GEN.LOG.2.1` Odbierane wiadomości ze zdarzeniami i wypisywanie ich w trybie ciągłym na CLI oraz zapisywane do plikowej bazy danych `sqlite`
`GEN.LOG.2.2` Odczytywanie historii zdarzeń z plikowej bazy danych `sqlite` z możliwością filtrowania

Wymagania dotyczące realizacji scenariuszy operacyjnych

Scenariusz 1: Analiza plików offline - PCAP

`OFF.PCAP.1` Analizator Cyberzagrożeń ma możliwość wyświetlania zawartości pakietów z wczytanych plików PCAP.

`OFF.PCAP.2` Analizator Cyberzagrożeń ma możliwość przekazania filtru zgodnego z formatem BPF (wykorzystywanego przez `libpcap` / `tshark` / `pyshark` / `Wireshark` / `Scapy`) do funkcji otwierającej i wczytującej plik PCAP.

Scenariusz 2: Analiza plików offline - TXT / Logi

`OFF.LOG.1` Analizator Cyberzagrożeń ma możliwość wywołania operacji systemowej `grep` na wskazanych plikach tekstowych. Argumentem przekazywanym do operacji jest właściwe wyrażenie

regularne.

> Tego zadania nie realizujemy dla plików EVTX, chyba, że je przetworzono do formatu XML/JSON/innego tekstowego i zapisano do pliku

`OFF.LOG.2` Analizator Cyberzagrożeń ma możliwość wywołania działania wyrażenia regularnego z modułu Python `re` na wskazanych plikach tekstowych lub EVTX przetworzonych do formatu JSON/XML/inny tekstowy. Argumentem przekazywanym do operacji jest właściwe wyrażenie regularne.

Scenariusz 3: Reguły jako funkcje języka `Python`

`OFF.DETPY.1` Analizator Cyberzagrożeń ma możliwość załadowania *reguł* do detekcji zdarzeń opisanych za pomocą tych reguł i przechowywania ich w wybranej strukturze danych. Każdorazowe wywołanie ładowania reguł ma oznaczać usunięcie istniejących w pamięci programu i załadowanie nowego zestawu reguł.

`OFF.DETPY.1.1` Reguły będą opisywane jako funkcje Pythona w pliku `detection-rules.py` (**nazwa na sztywno**)

`OFF.DETPY.1.2` Każda reguła ma być zdefiniowana jako oddzielna funkcja w języku Python we wskazanym pliku w `OFF.DETPY.1.1` . Format pojedynczej reguły:

```
def nazwa_funkcji_reguly(**kwargs):
    # ciało funkcji – właściwa reguła operująca na danych z args

    # procesowanie pcap
    # for pcap in kwargs[pcap]:

    # procesowanie evtx
    # for evtx in kwargs[evtx]:

    # procesowanie xml
    # for xml in kwargs[xml]:

    # procesowanie json
    # for json in kwargs[json]:

    # procesowanie txt
    # for txt in kwargs[txt]:


    # ostateczna reguła – tj. co ma się wykonać
    if condition=True:
        action_alert = "..." # akcja: "local", "remote"
        description = "Alert ..."
    else:
        action_alert = None
        description = None
    return action_alert, description
```

`OFF.DETPY.1.3` Informacjami zwrotnymi z reguły są:

- `action_alert` - jedna z dwóch akcji alertujących z grupy:
  - `local` - oznacza wypisanie informacji o zdarzeniu na CLI oraz do loga
  - `remote` - oznacza to samo co `local` oraz wysłanie informacji po REST API do aplikacji Zdalnego Kolektora Zdarzeń (Wymagania `GEN.MGMT.5` , `GEN.LOG.1`)
- `description` - opis tekstowy według przyjętego formatu, może to być także JSON, XML czy syslog.

`OFF.DETPY.2` Interfejs wywołania reguł umożliwia ich użycie w dwóch trybach:

`OFF.DETPY.2.1` Wywołanie całego zestawu reguł na wybranym zestawie plików

`OFF.DETPY.2.2` Wywołanie wybranej reguły - poprzez wskazanie jej nazwy (nazwa funkcji z `OFF.DETPY.1.2`) - i na wybranym zestawie plików przekazanym do reguły

Scenariusz 4: Wykorzystanie uniwersalnego formatu reguł - SIGMA

`REG.DET.1` W ramach środowiska stworzonej aplikacji Analizatora Cyberzagrożeń należy zintegrować rozwiązanie do detekcji z wykorzystaniem reguł SIGMA.
`REG.DET.1.1` Silnik dla reguł SIGMA w języku Python: https://github.com/wagga40/Zircolite
`REG.DET.1.2` Integracja taka może opierać się na wskazaniu pojedynczej reguły lub zestawu reguł. Poza samą detekcją należy pamiętać o integracji związanej prezentacją alertu.

`REG.DET.2` Przetestować zintegrowane rozwiązanie dla reguł SIGMA.
`REG.DET.2.1` Wykonać na wybranym zestawie danych EVTX/JSON względem znanej reguły SIGMA dla tego zestawu danych (szukać w podanych źródłach i w Internecie). Można też okroić regułę SIGMA, aby na posiadanych danych zademonstrować działanie detekcji.

Dodatkowe pytania teoretyczne (dla chętnych)

W jaki sposób można wyskalować budowane rozwiązanie z punktu widzenia potrzeb, na które odpowiadają systemy EDR/XDR? Uwzględnij kwestię wbudowania mechanizmów *odpowiedzi* (*response*) na wykryte cyberzagrożenie.

Rozliczenie zadania

Wymagany wynik zadania

Wynikiem zadania do sprawdzenia jest raport oraz kod aplikacji. Raport ma spełniać wymagania raportu technicznego (strona tytułowa, spis treści, wstęp, podsumowanie, formatowanie, numeracja stron). Część implementacyjną można opisać zwięźle za pomocą tabeli, która dla każdego głównego wymagania zawiera informację o sposobie implementacji oraz zrzut ekranu z działania.

ID wymagania	Opis realizacji wymagania	Zrzut ekranu z krótkim opisem działa

Zwieńczeniem raportu mają być **wnioski**, w szczególności wskazanie najciekawszych decyzji projektowych aplikacji oraz pomysły na rozwój samego prototypu aplikacji, np. wskazanie jak można byłoby zintegrować różne algorytmy detekcji. Do tego można odpowiedzieć na dodatkowe pytanie teoretyczne.

Raport można przygotować jako:

- plik PDF
- plik Markdown dołączony do repozytorium z kodem

Bieżący dostęp do repozytorium Git z kodem aplikacji można ustanowić w serwisach:

- `Github` - login Prowadzącego: `jbieniasz`
- `Gitlab` zewnętrzny - login Prowadzącego: `bieniop`
- `Gitlab` Elka - login Prowadzącego: `jbienias`

Ostatecznie, wyniki laboratorium należy przekazać przez wskazany w LeOnie formularz MS Forms. W formularzu należy:

- wpisać skład zespołu (adresy e-mail w domenie `*@pw.edu.pl`)
- dołączyć link do folderu OneDrive z materiałami do rozliczenia. W tym folderze należy zawrzeć:
  - plik ZIP z kodem (pobrany z danego repozytorium Git)
  - raport - plik PDF
    - jeżeli raport jest opracowywany źródłowo w Markdown - wygenerować plik PDF lub HTML.

Wskazówki i narzędzia

1. Moduły Python potrzebne do realizacji zadania:
   - dystrybucja `miniconda3` - na każdym z hostów do realizacji
     - warto przygotować wydzielone środowisko z właściwymi modułami, tj. nie mieszać go z wcześniej używanym do ćwiczeń z analityki
   - `Click`
   - `msticpy` , `pandas` , `sqlite` , `scapy` , `pyshark` (wymagany zainstalowany tshark w ramach wiresharka), `fastapi` https://fastapi.tiangolo.com (do tego serwer ASGI Uvicorn)
   - inne moduły według potrzeb tworzenia rozwiązania
2. Python jest wykorzystywany jako narzędzie do pisania skryptów systemowych, w których tradycyjne polecenia powłoki Bash są *wrappowane* w funkcje Pythonowe (wygoda). Część operacji Pythona de facto wykonuje pod spodem operacje systemowe danej powłoki systemu operacyjnego, w którym działa - np. interfejs do systemu plików.
3. Należy przemyśleć, w którym miejscu aplikacji następuje zapakowanie do właściwych formatów danych, np. JSON w przypadku wysyłania danych po `REST API` . Jest to istotne dla realizacji Wymagań
4. Celem jest jak najprostsze opracowanie każdego z punktów wymagań, mimo, że wydają się one rozbudowane. Przy tej okazji dokładnie wskazują/podpowiadają co należy zrealizować.

5. Wszystkie właściwe endpointy w koniecznej komunikacji `REST API` skonfigurować według własnego uznania.
6. Na potrzeby sterowania zdalnego należy wybrać metodę przekazywania komend, np. odpowiednie słowo kluczowe przekazywane jako tekst w obiekcie JSON.
7. Wszystkie rozwiązania i potrzebne obejścia są dopuszczalne, przy czym należy zachowywać prostotę oraz cele całego ćwiczenia.

Dane z sieci i hostów - na potrzeby testów

Źródła danych cyber do ćwiczeń z inżynierii detekcji (przykłady)

- Detection Hackaton APT29 https://github.com/OTRF/detection-hackathon-apt29
  - Dane z dnia 1: https://github.com/OTRF/detection-hackathon-apt29/tree/master/datasets/day1
  - Dane z dnia 2: https://github.com/OTRF/detection-hackathon-apt29/tree/master/datasets/day2
- zdarzenia EVTX: https://github.com/sbousseaden/EVTX-ATTACK-SAMPLES
- projekt Security Datasets (w grupie projektów Open Threat Research)
  - https://securitydatasets.com/introduction.html
    - Przykład techniki ze źródłami dla hosta i sieci: https://securitydatasets.com/notebooks/atomic/windows/lateral_movement/SDWIN-190518210652.html
- różne PCAPy: https://www.netresec.com/?page=PcapFiles
- dane dla nowych malware: http://www.malware-traffic-analysis.net
- dane towarzyszące case'om z bloga Security Onion: https://blog.securityonion.net

Przydatne w zadaniu z modelem detekcji anomalii:

- logi Zeek: https://github.com/SuperCowPowers/zat/tree/master/data
  - opis sposobu logowania w Zeek: https://docs.zeek.org/en/master/frameworks/logging.html
  - dane z TSV (*T - tabbed, CSV - C - comma*) można przetworzyć na JSON, lub wykorzystać Zeek Analysis Tools do ładowania (https://github.com/SuperCowPowers/zat)
- w przypadku przetwarzania logów Zeek: `zat` (https://github.com/SuperCowPowers/zat)
- dataset dla IoT - PCAP oraz logi Zeek https://www.stratosphereips.org/blog/2020/1/22/aposemat-iot-23-a-labeled-dataset-with-malicious-and-benign-iot-network-traffic
- logi Zeek/Suricata: https://drive.google.com/drive/folders/1rr765z3XwW_NwhcFwWfgPrGsSLW2LuKk?usp=sharing