Bezpieczeństwo Systemów Informatycznych

Plan Testów

Analiza ruchu sieciowego w celu identyfikacji problemów z bezpieczeństwem na podstawie plików PCAPs za pomocą wybranych bibliotek w Pythonie

Adrianna Kopeć

Anna Nagi

Izabela Pachel

1. Wstęp

Celem niniejszego dokumentu jest określenie ogólnego podejścia do procesu testowego tworzonej aplikacji - wskazanie przedmiotu i zakresu testów, opisanie środowiska testowego i użytych narzędzi oraz opracowanie harmonogramu.

Celem działań testowych jest sprawdzenie poprawności implementacji aplikacji oraz wykrycie i naprawa istniejących błędów.

2. Przedmiot testów

Przedmiotem testów będą poszczególne elementy i funkcjonalności tworzonej aplikacji. Szczegółowy zakres i rodzaj planowanych testów zostały opisane w następnej sekcji.

3. Zakres testów

➢ Testy jednostkowe - najbardziej podstawowy poziom:

(a) walidacja wprowadzanych argumentów programu

(b) sprawdzenie poprawności poszczególnych funkcji programu - walidacja argumentów i zwracanych wartości

(c) sprawdzenie zapisu do plików

➢ Testy funkcjonalne - sprawdzenie czy funkcjonalności działają zgodnie z założeniami:

(a) Załadowanie pliku PCAP do programu

(b) Pobranie bieżących logów - określenie liczby pakietów i ich zapis do pliku PCAP

(c) Oglądanie statystyk analizowanego pliku

(d) Sprawdzenie czy nie wykryto zagrożeń w analizowanych danych

➢ Testy integracyjne - sprawdzenie poprawności interakcji między komponentami:
(a) wywołanie zewnętrznego programu Suricata
(b) przechwytywanie logów z sieci przez bibliotekę Scapy

➢ Testy wydajnościowe - aplikacja jest bardzo prostym programem, który nie komunikuje się z żadnym serwerem i może obsłużyć tylko jednego użytkownika na raz, dlatego zmierzone zostaną jedynie:
(a) średni czas pobrania logów z sieci w zależności od natężenia ruchu w sieci
(b) średni czas przetwarzania pliku PCAP pod kątem obliczenia statystyk w zależności od wielkości pliku
(c) średni czas przetwarzania pliku PCAP pod kątem wykrycia zagrożeń w zależności od wielkości pliku
(d) średnie wykorzystanie pamięci przez program

➢ Testy bezpieczeństwa - ze względu na małą złożoność programu i brak przetwarzania poufnych danych zrezygnowano z ich przeprowadzenia
➢ Testy akceptacyjne - ze względu na małą złożoność programu zrezygnowano z ich przeprowadzenia, ponieważ pozostałe testy są wyczerpujące

4. Kryterium wejścia/wyjścia

- Kryterium wejścia:
  - przygotowane środowisko testowe: zainstalowane potrzebne biblioteki i frameworki
  - zaimplementowane odpowiednie fragmenty programu
  - zdefiniowane przypadki testowe
- Kryterium wyjścia:
  - wykonanie wszystkich zaplanowanych testów

5. Podstawowe przypadki testowe

	Nazwa	Dane testowe	Warunki wstępne	Kroki wykonania	Oczekiwany rezultat
1	Wyświetlenie pomocy.	-	Poprawnie zdefiniowano opcje i ich opisy w pomocy.	1.Użytkownik uruchamia program z argumentem '--help'.	Zostaną wyświetlone dostępne opcje programu wraz z opisem.
2	Załadowanie pliku PCAP do programu bez podania ścieżki.	-	Opcja -f istnieje. Poprawnie zaimplementowano walidację argumentu.	1.Użytkownik uruchamia program z argumentem '-f', nie podając ścieżki do pliku.	Zostaje wyświetlony komunikat o brakującej ścieżce. Plik nie zostaje załadowany do programu.
3	Załadowanie pliku PCAP do programu, podając błędną ścieżkę.	Zła ścieżka do pliku.	Opcja -f istnieje. Poprawnie zaimplementowano walidację argumentu.	1.Użytkownik uruchamia program z argumentem '-f' i błędną ścieżką do pliku.	Zostaje wyświetlony komunikat o złej ścieżce. Plik nie zostaje załadowany do programu.
4	Załadowanie pliku PCAP do programu.	Plik z rozszerzeniem .pcap	Opcja -f istnieje. Poprawnie zaimplementowano walidację argumentu.	1.Użytkownik uruchamia program z argumentem '-f' i ścieżką do pliku PCAP.	Plik zostaje poprawnie załadowany do programu.
5	Załadowanie innego pliku niż PCAP do programu.	Plik z rozszerzeniem innym niż .pcap	Opcja -f istnieje. Poprawnie zaimplementowano walidację argumentu.	1.Użytkownik uruchamia program z argumentem '-f' i ścieżką do innego pliku.	Zostaje wyświetlony komunikat o błędnym formacie. Plik nie zostaje załadowany do programu.
6	Przechwycenie plików PCAP bez podania l. pakietów.	-	Opcja -c istnieje. Poprawnie zaimplementowano walidację argumentu.	1.Użytkownik uruchamia program z argumentem '-c' i bez podania l. pakietów.	Zostaje przechwycona domyślna liczba pakietów.
7	Przechwycenie plików PCAP błędnie podając l. pakietów.	Argument inny niż int.	Opcja -c istnieje. Poprawnie zaimplementowano walidację argumentu.	1.Użytkownik uruchamia program z argumentem '-c' i podając zły typ arg.	Zostaje wyświetlony komunikat o błędnym formacie. Pakiety nie zostały przechwycone.
8	Przechwycenie plików PCAP.	Liczba int jako argument.	Opcja -c istnieje. Poprawnie zaimplementowano walidację argumentu.	1.Użytkownik uruchamia program z argumentem '-c', podając l. pakietów.	Zostaje przechwycona określona liczba pakietów.

9	Analiza statystyczna pliku PCAP.	-	Opcja -a istnieje.	1.Użytkownik uruchamia program z argumentem ‘-a’.	Zostaje dokonana analiza statystyczna. Wykresy zostają zapisane w folderze Results\Statistics\.
10	Detekcja zagrożeń w pliku PCAP.	-	Opcja -d istnieje.	1.Użytkownik uruchamia program z argumentem ‘-d’.	Zostaje dokonana detekcja zagrożeń. Zostaje wyświetlone podsumowanie.

6. Środowisko testowe

Testy zostaną przeprowadzone na 3 laptopach o specyfikacjach:

1. Laptop Dell Inspiron 5567
   - Procesor: Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.70 GHz, 2 rdzenie, 4 procesory logiczne
   - RAM: 16 GB
   - Karta graficzna: Radeon ™ R7 M445
   - System: Windows 10
2. Laptop ASUS K501L
   - Procesor: Intel Core i5
   - RAM: 8 GB
   - Karta graficzna: Nvidia Geforce GTX 950M
   - System: Windows 8.1
3. Lenovo IdeaPad S340
   - Procesor: AMD Ryzen™ 5 3500U (4 rdzenie, 8 wątków, 2.10–3.70 GHz)
   - RAM: 20 GB
   - Karta graficzna: AMD Radeon™ Vega 8
   - System: Windows 10

Do przeprowadzenia testów należy zainstalować i skonfigurować narzędzia wymienione w następnej sekcji.

7. Narzędzia

- *pytest* - framework do testów jednostkowych, funkcjonalnych, integracyjnych
- *pytest-cov* - rozszerzenie do tworzenia raportu pokrycia kodu testami
- *timeit* - biblioteka umożliwiająca pomiar czasu w testach wydajnościowych
- *memory-profiler* - biblioteka umożliwiająca pomiar zużycia pamięci w testach wydajnościowych
- *Github* - zgłaszanie i śledzenie wykrytych błędów w programie

8. Harmonogram testów

12.11 - przygotowanie planu testów
24.11 - rozpoczęcie implementacji programu i testów jednostkowych
01.12 - przedstawienie planu testów
03.01 - przeprowadzenie testów funkcjonalnych
07.01 - przeprowadzenie testów integracyjnych
10.01 - przeprowadzenie testów wydajnościowych
12.01 - podsumowanie testów i dołączenie ich do dokumentacji

Testy jednostkowe będą wykonywane na bieżąco, równolegle z implementacją aplikacji.

9. Zarządzanie błędami

Wszelkie incydenty stwierdzone podczas testów zostaną zgłoszone w sekcji Issues z etykietą “bug” w repozytorium projektu na GitHubie:

https://github.com/kopeadri/BSI-Project/issues

10. Raport z testów

Na koniec procesu testowego w dokumentacji projektu znajdą się:

- plan testów
- lista uwzględnionych przypadków testowych wraz z ich statusami
- raport pokrycia kodu testami
- raporty z poszczególnych rodzajów testów

11. Role i odpowiedzialność

- Adrianna Kopeć - projektant, tester
- Anna Nagi - tester
- Izabela Pachel - tester

Projektant :
- Określenie zakresu i strategii testów
- Wybranie narzędzi
- Określenie danych do automatycznych testów
- Sporządzenie dokumentacji

Tester:
- Implementacja testów
- Przeprowadzenie testów
- Zgłaszanie napotkanych błędów
- Przygotowanie raportów

12. Ryzyka

Ryzyka, które mogą wystąpić w trakcie testów:

- Choroba/nieobecność członka zespołu
- Problemy ze sprzętem, awarie
- Braki w umiejętnościach testowych
- Błędne oszacowanie harmonogramu - opóźnienia w implementacji
- Brak komunikacji w zespole i zgłaszania błędów
- Złe zarządzanie projektem - brak podziału zadań

Harmonogram tworzony był z uwzględnieniem możliwych opóźnień w implementacji aplikacji i testów. Wszelkie zmiany są na bieżąco aktualizowane. Aby być w stałym kontakcie z członkami zespołu została utworzona grupa konwersacyjna.