Laboratorium 5 – Wyrażenia regularne, interfejsy linii komend.

Języki skryptowe

Cele dydaktyczne

1. Przetwarzanie wyrażeń regularnych.
2. Zapoznanie z wybranymi modułami biblioteki standardowej Pythona.
3. Tworzenie tekstowych interfejsów użytkownika.

Wprowadzenie

W ramach niniejszego laboratorium przedmiotem analizy będą logi SSH. SSH (*secure shell)* to protokół sieciowy, który zapewnia bezpieczne połączenie sieciowe między dwoma urządzeniami. SSH może służyć do zdalnego logowania do komputera i wykonywania poleceń w powłoce systemowej.
Przykładowe pliki z logami SSH można znaleźć tutaj, a jego próbę tutaj. Każdy wiersz zawiera Znacznik czasowy, nazwę hosta, komponent aplikacji, i numer PID, opis zdarzenia.
Przeanalizuj strukturę opisów zdarzeń. Zawierają one m.in. informacje na temat nieudanego połączenia SSH, takie jak błędne logowanie i zamknięcie połączenia. Logi zawierają również szczegółowe informacje na temat procesu autentykacji, w tym informacje o użytkowniku, który próbuje się zalogować oraz dane o połączeniu, takie jak numer portu.
Przykładowo, opis zdarzenia udanego logowania rozpoczyna się od `"Accepted password for"` natomiast nieudanego logowania: `"Failed password for"`.

Zadania

1. Skonstruuj skrypt, który będzie w stanie:
    a. odczytać plik z logami SSH,
    b. reprezentować wiersze w wybranej formie (np. strumienia słowników lub

nazwanych krotek),
    c. przetwarzać dane dla kolejnych wierszy, zgodnie ze specyfikacją określoną w kolejnych zadaniach.
2. Z wykorzystaniem wyrażeń regularnych:
    a. Napisz funkcję, która przyjmuje na wejściu ciąg znaków określający wiersz, a zwraca na wyjściu wybraną ustrukturalizowaną reprezentację (np. słownik, namedtuple),
    b. Napisz funkcję get_ipv4s_from_log(), która przyjmuje na wejściu reprezentację wiersza, przy użyciu wyrażeń regularnych wyszukuje w opisie zdarzenia adresy IPv4 i na wyjściu zwraca listę tych adresów.
    c. Napisz funkcję get_user_from_log(), która przyjmuje na wejściu reprezentację wiersza, a zwraca ciąg znaków określający nazwę użytkownika, którego dotyczy opis zdarzenia w ramach wpisu. W przypadku braku odniesienia do nazwy użytkownika we wpisie, należy zwrócić None
    d. Napisz funkcję get_message_type(), która przyjmuje na wejściu opis zdarzenia, i zwraca informację określającą, czy jest to wiersz dotyczący: udanego logowania, nieudanego logowania, zamknięcia połączenia, błędnego hasła, błędnej nazwy użytkownika, próby włamania. Dla pozostałych wierszy, zwróć *inne*.
3. Skonfiguruj moduł logging, tak, aby:
    a. Po przeczytaniu każdego wiersza logować na poziomie DEBUG liczbę przeczytanych bajtów.
    b. Po przeczytaniu wiersza z informacją o udanym zalogowaniu lub zamknięciu połączenia, logować odpowiednią informację w trybie INFO.
    c. Po przeczytaniu wiersza z informacją o próbie nieudanego logowania, logować odpowiednią informację w trybie WARNING.
    d. Po przeczytania wiersza z informacją o błędzie, logować w trybie ERROR.
    e. Po przeczytaniu wiersza z informacją o próbie włamania, logować w trybie CRITICAL.
    f. Logi na poziomach DEBUG, INFO, WARNING i CRITICAL były wypisywane na stdout, natomiast ERROR na stderr (wyjście błędu).
4. Korzystając z opracowanych funkcji oraz modułów random, datetime oraz statistics, skonstruuj poniższe funkcje:
    a. Funkcja zwracająca *n* losowo wybranych wpisów z logami dotyczących losowo wybranego użytkownika.
    b. Funkcja obliczająca średni czas trwania i odchylenie standardowe czasu trwania połączeń SSH.
        i. globalnie, dla całego pliku z logami,

        ii. dla każdego użytkownika niezależnie.
    c. Funkcja obliczająca użytkowników, którzy logowali się najrzadziej i najczęściej.
    d. Funkcja zwracająca
5. Korzystając z modułu argparse, zaprojektuj i skonstruuj przyjazny użytkownikowi CLI (ang. *command-line interface* – interfejs linii komend), który:
    a. z wykorzystaniem argumentów:
        i. umożliwi wskazanie lokalizacji pliku z logiem (wymaganego),
    b. pozwoli na określenie minimalnego poziomu logowania (opcjonalnego).
    c. z wykorzystaniem podkomend, pozwoli uruchomić niezależnie każdą z funkcjonalności z zadań 2 i 4

Zadania rozszerzające (dla ambitnych)

1. Napisz własną funkcję, która będzie analizować logi i wykrywać próby ataku typu brute-force, tj. wielokrotne próby logowania się do serwera poprzez próbę weryfikacji wszystkich kombinacji haseł. Na potrzeby zadania należy przyjąć, że *brute force* oznacza łańcuch nieudanych połączeń wykonanych w pewnym interwale z tego samego adresu IP. Niech funkcja będzie parametryzowalna, tj.
    a. niech pozwala na określenie maksymalnego interwału między kolejnymi połączeniami w ramach łańcucha połączeń,
    b. niech pozwala na określenie, czy wykrywać ataki na pojedynczą nazwę użytkownika, czy na wiele.

Funkcja powinna przyjąć na wejście listę reprezentacji kolejnych wierszy i zwrócić na wyjście wykryte próby ataku zawierające: znacznik czasowy, adres IP atakującego i liczbę wykonanych prób ataku. Skonstruuj CLI do uruchomienia tej funkcji.

2. Przejrzyj narzędzia takie jak typer, click, docopt. Wybierz jedno z nich. Skonstruuj kolejny interfejs linii komend z wykorzystaniem wybranego narzędzia, według tego samego projektu, co CLI z zadania 5. Porównaj i przeanalizuj oba interfejsy.