Laboratorium 5 – Wyrażenia regularne, interfejsy linii komend.

Cele dydaktyczne

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

Program można zgłosić jako zrobiony, jeśli spełnione są poniższe warunki:

1. Program jest zgodny z podaną specyfikacją.
2. Został przetestowany.
3. Student go rozumie i potrafi wyjaśnić.

UWAGA:

Wykonywanie zadań przy użyciu czatu GPT będzie traktowane jako praca niesamodzielna i będzie skutkować oceną niedostateczną.

- Zadbaj o to, aby każda funkcja w programie miała tylko jedną odpowiedzialność.
- Zadbaj o rozdzielenie funkcji przetwarzających dane od funkcji najwyższego poziomu wypisujących tekst na wyjście standardowe.
- W przypadku, gdy kilka funkcjonalności wymaga skorzystania z samych funkcji, umieść je w osobnym module, który będzie ponownie użyty.
- Przed potencjalnie źle sformatowanymi danymi zabezpiecz się wykorzystując mechanizm wyjątków.

- **z wykorzystaniem *mechanizmu obsługi wyjątków* zadbaj o poprawność działania funkcji**
- **należy zabezpieczyć funkcje przed błędnymi danymi wejściowymi i wyjątkami. (np. dzieleniem przez zero).**
- **do zadań przygotować 3 – 5 testów sprawdzających poprawność działania**
- **do prowadzącego wysłać pliki z kodem źródłowym oraz rzuty ekranu z przygotowany testami**

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

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.

1.1. Z wykorzystaniem wyrażeń regularnych:

1.1.1 Napisz funkcję, która przyjmuje na wejściu ciąg znaków określający wiersz, a zwraca na wyjściu wybraną ustrukturalizowaną reprezentację (np. słownik, namedtuple),

1.1.2. 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.

1.1.3. 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

1.1.4. 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*.

Punkty: 3

1.2. Skonfiguruj moduł **logging**, tak, aby:

1.2.1 Po przeczytaniu każdego wiersza logować na poziomie DEBUG liczbę przeczytanych bajtów.

1.2.2. Po przeczytaniu wiersza z informacją o udanym zalogowaniu lub zamknięciu połączenia, logować odpowiednią informację w trybie INFO.

1.2.3. Po przeczytaniu wiersza z informacją o próbie nieudanego logowania, logować odpowiednią informację w trybie WARNING.

1.2.4. Po przeczytania wiersza z informacją o błędzie, logować w trybie ERROR.

1.2.5. Po przeczytaniu wiersza z informacją o próbie włamania, logować w trybie CRITICAL.

1.2.6. Logi na poziomach DEBUG, INFO, WARNING były wypisywane na stdout, natomiast ERROR i CRITICAL na stderr (wyjście błędu).

Punkty: 2

1.3. Korzystając z opracowanych funkcji oraz modułów random, datetime oraz statistics, skonstruuj poniższe funkcje:

1.3.1. Funkcja zwracająca *n* losowo wybranych wpisów z logami dotyczących losowo wybranego użytkownika.

1.3.2. Funkcja obliczająca średni czas trwania i odchylenie standardowe czasu trwania połączeń SSH.

- globalnie, dla całego pliku z logami,
- dla każdego użytkownika niezależnie.

1.3.3. Funkcja obliczająca użytkowników, którzy logowali się najrzadziej i najczęściej.

Punkty: 2

1.4. Korzystając z modułu argparse, zaprojektuj i skonstruuj przyjazny użytkownikowi CLI (ang. *command-line interface* – interfejs linii komend), który:

1.4.1. z wykorzystaniem argumentów umożliwi wskazanie lokalizacji pliku z logiem (wymaganego),

1.4.2. pozwoli na określenie minimalnego poziomu logowania (opcjonalnego).

1.4.3. z wykorzystaniem podkomend, pozwoli uruchomić niezależnie każdą z funkcjonalności z zadań 1.1 i 1.3

Punkty: 3

Zadanie 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.

Punkty: 2