29.04.2022

Na dzisiejszych zajęciach zapoznamy się z protokołem HTTP.

- Zapoznaj się z ogólną strukturą wiadomości HTTP:

  > A message consists of header fields and, optionally, a body. The body is simply a sequence of lines containing ASCII characters. It is separated from the headers by a null line (i.e., a line with nothing preceding the CRLF). (Źródło: RFC 822)

  oraz z bardziej szczegółową definicją ze specyfikacji HTTP (RFC 2616, sekcje 4-6).

- Uruchom Netcat-a w trybie TCP (komunikacja HTTP jest oparta na TCP) na pewnym porcie:

  ```
  ncat -l 127.0.0.1 2020
  ```

  a następnie spróbuj otworzyć adres `127.0.0.1:2020` w przeglądarce internetowej. Czy dane odebrane przez Netcat-a zgadzają się z formatem zaptania HTTP?

- Zapoznaj się ze stronami manuala dla narzędzi `wget` oraz `curl`. Spróbuj ściągnąć zawartość wybranej strony internetowej przy pomocy `wget` oraz `curl`. Powtórz ćwiczenie z poprzedniego punktu z Netcat-em w roli serwera ale wysyłając żądanie przy pomocy `wget`-a:

  ```
  wget 127.0.0.1:2020
  ```

- Spróbuj przy pomocy programu `curl` wykonać zapytanie HTTP `GET` pod adresem zasobu:

  ```
  http://sphinx.if.uj.edu.pl/techwww/httptest/test
  ```

  W przypadku poprawnego zapytania serwer powinien zwrócić w treści odpowiedzi "Gratulacje! Wykonałeś zapytanie HTTP GET."

- Podobnie spróbuj wykonać zapytanie `POST` pod ten sam adres, ale ustawiając pole nagłówka HTTP `"Content-Type: application/json"`. W przypadku poprawnego zapytania serwer powinien zwrócić w treści odpowiedzi "Gratulacje! Wykonałeś zapytanie HTTP POST z typem zawartości JSON."

- Znajdź bibliotekę w swoim wybranym języku programowania pozwalającą na wykonywanie zapytań HTTP. Powtórz dwa ostatnie punkty przy jej pomocy. Sprawdź, czy wybrana biblioteka potrafi obsługiwać inne metody niż GET i POST, w jaki sposób specyfikuje się argumenty przesyłane w zapytaniach POST, czy potrafi obsługiwać ciasteczka, i czy potrafi je zapisywać w tzw. cookie jar.

- Używając wybranego języka programowania i biblioteki, napisz program wykonujący zapytania GET i POST do testowego serwisu tak jak przy użyciu `curl` we wcześniejszych zadaniach.

Pod adresem:

```
POST http://sphinx.if.uj.edu.pl/techwww/httptest/login
```

znajduje się zasób, pod którym można "zalogować" się do serwisu przy pomocy dowolnej nazwy użytkownika (`login=...`) i hasła `0123` (`password=0123`). Na potrzeby testowania można powyższy adres otworzyć w przeglądarce internetowej. Po poprawnym zalogowaniu, serwer ustawia w programie-kliencie ciasteczko identyfikujące sesję. Tylko "zalogowany" użytkownik z odpowiednim ciasteczkiem może dostać się do zasobu:

```
GET http://sphinx.if.uj.edu.pl/techwww/httptest/private
```

- Stwórz program-klienta HTTP, który:

    1. Zaloguje się wysyłając zapytanie HTTP POST poz zasób `login` wraz z odpowiednimi danymi (Gdzie te dane powinny się znaleźć w zapytaniu POST? Warto podejrzeć co wysyła przeglądarka kiedy użyjemy formularza na stronie HTML, może się też przydać `curl` z opcją `-d`.).
    2. Zachowa ciasteczko otrzymane od serwera w wyniku zalogowania.
    3. Wykona zapytanie GET pod zasób `private` wysyłając przy tym przechowane ciasteczko tak, aby zapytanie zakończyło się sukcesem.

    Uważaj na obsługę błędów, szczególnie na kody statusu HTTP. Wszystkie zapytania w powyższym schemacie powinny kończyć się odpowiedzią serwera ze statusem `2XX`. Co się dzieje np. przy zapytaniu o zasób `private` bez odpowiedniego ciasteczka?

    Do testowania powyższego schematu i dla lepszego jego zrozumienia warto również powtórzyć go przy użyciu narzędzia `curl` z odpowiednimi opcjami.