Technologie Sieciowe - lista piąta

Bartosz Rajczyk

13 czerwca 2019

1 Opisy rozwiązań

1. Uruchomiłem oryginalny skrypt, aby zorientować się, że nie działa on w ogóle. Problemem był parametr `LocalAddr` ustawiony na `lukim`, który to niestety nie jest nazwą hosta mojego komputera. Z tego powodu zmieniłego go na starego, dobrego `localhost`. Analizu skryptu wykazała, że przyjmuje on requesty w nieskończonej pętli, sprawdza, czy ich metodą jest GET, jeżeli tak, to wysyła jakiś plik, jeżeli nie, to wysyła status błędu.
2. Czego?
3. Nawiązałem połączenie przy pomocy przeglądarki internetowej, został mi wysłany html, który wcześniej przygotowałem.
4. Napisałem czterolinijkowy skrypt w JS-ie odsyłający wszystkie nagłówki zapytania w formie tekstu. W moim przypadku jest ich dziewięć w przypadku zapytania z Opery
   - Host
   - Connection
   - Cache-control
   - Upgrade-Insecure-Requests
   - User-Agent
   - Accept
   - Referer
   - Accept-Encoding
   - Accept-Language

   tylko siedem przy zapytaniu przy pomocy Edge'a (nie ma Connection ani Cache-Control) i tylko dwa (User-Agent i Host) przy użyciu curla w Powershellu.
5. Napisałem prymitywny skrypt w JS-ie hostujący statycznie pliki z wybranego folderu, routujący zapytania pod / na `index.html` i wysyłający informację o braku strony w przypadku braku strony.

6. Użyłem WireSharka na Npcap Loopback Adapter, aby przeanalizować ruch lokalny powstający przy poprzednich zadaniach. Zauważyłem, że przeglądarka wysyła zapytanie o konkretną stronę, a po jej otrzymaniu wysyła zapytanie o podlinkowane zasoby (jak chociażby style) i favicon. Hostowanie faviconu nie jest możliwe ze względu na prymitywność serwera, ale jako że css jest formatem tekstowym, możemy hostować także go. Analizując pakiety widzimy także, dlaczego HTTP jest protokołem tekstowym; wszystkie przekazywane dane są dosłownie tekstem, możemy zobaczyć wysyłaną treść HTML, wszystkie nagłówki i adresy.

   Rozwiązaniem tego problemu jest protokół HTTPS.