

Sieci komputerowe

Warsztaty 6

Mateusz Markiewicz

16 maja 2020

1 Pierwsze zadanie do zaprezentowania

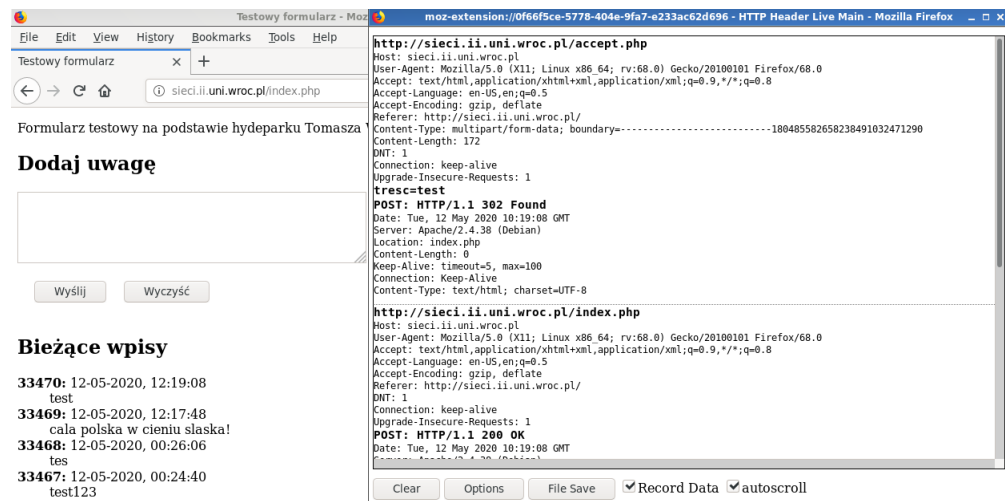
Pierwszym etapem zadania było stworzenie maszyny wirtualnej z domyślną konfiguracją sieciową.

Następnie nazwałem interfejs tej maszyny, użyłem w tym celu następujących poleceń:

- `V1#> ip link set enp0s3 name enp0`
- `V1#> dhclient -v enp0`

Następnie do pliku `/etc/hosts` za pomocą vim'a dodałem nowy wiersz:
`156.17.4.30 sieci.ii.uni.wroc.pl`

Następnie wszedłem na stronę `http://sieci.ii.uni.wroc.pl/`, uruchomiłem rozszerzenie *Live HTTP Header* oraz dodałem nowy wpis.

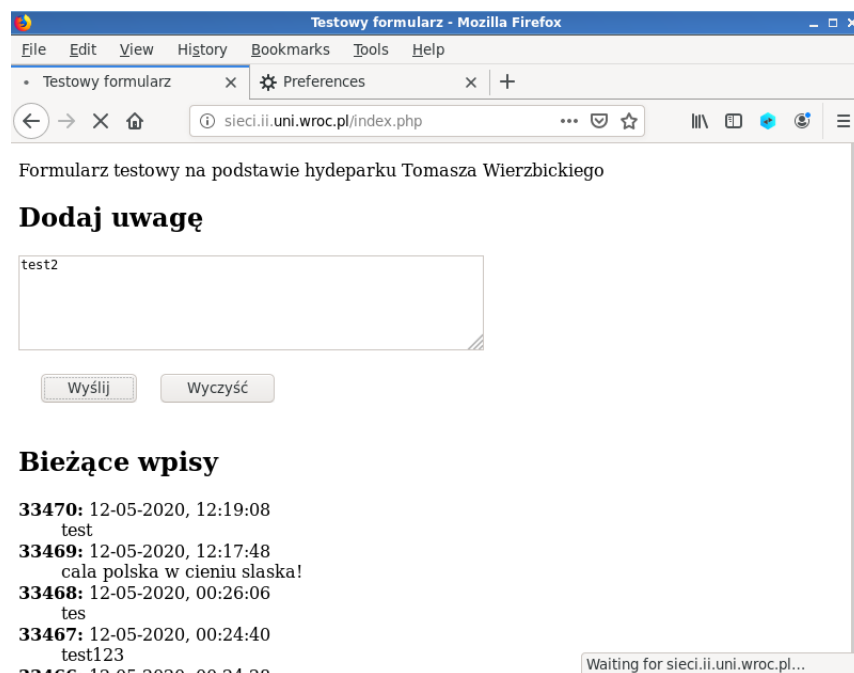


Dzięki rozszerzeniu możemy zaobserwować wysłane zapytanie HTTP (Post, z treścią, którą wysłaliśmy) oraz otrzymaną odpowiedź (200 OK).

Następnie uruchomiłem program nc w trybie serwera TCP nasłuchującego na porcie 8888 za pomocą polecenia:

- `V1$> nc -l -p 8888 | tee http_request`

oraz w przeglądarce ustawiłem manualną konfigurację proxy pod adres localhost oraz port 8888 (czyli adres i port na którym działa nasz serwer TCP). Następnie spróbowałem dodać następny wpis na stronie `http://sieci.ii.uni.wroc.pl/`:



Jak widać przeglądarka wyświetla "Waiting for sieci.ii.uni.wroc.pl...". Dzieje się tak z powodu konfiguracji proxy, którą ustawiliśmy. Przeglądarka wysyła zapytanie HTTP pod adres localhost oraz port 8888. Zapytanie to dociera do naszego serwera TCP, ale nie dociera do serwera obsługującego formularz, dlatego wpis nie został dodany.

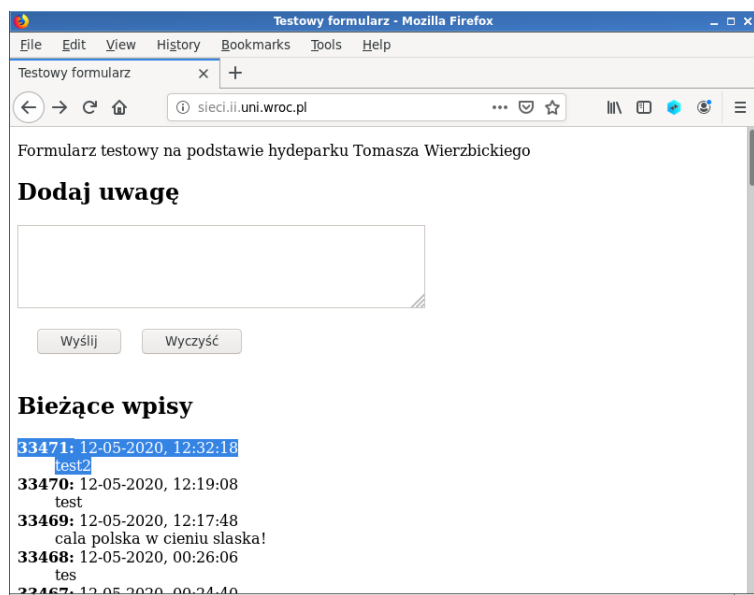
Następnie wyświetliłem zawartość pliku http_request:

```
user@virbian: ~  
00000070 4c 69 6e 75 78 20 78 38 36 5f 36 34 3b 20 72 76 |Linux x86 64; rv|  
00000080 3a 36 38 2e 30 29 20 47 65 63 6b 6f 2f 32 30 31 |:68.0) Gecko/201|  
00000090 30 30 31 30 31 20 46 69 72 65 66 6f 78 2f 36 38 |00101 Firefox/68|  
000000a0 2e 30 0d 0a 41 63 63 65 70 74 3a 20 74 65 78 74 ||.0..Accept: text|  
000000b0 2f 68 74 6d 6c 2c 61 70 70 6c 69 63 61 74 69 6f |/html,applicatio|  
000000c0 6e 2f 78 68 74 6d 6c 2b 78 6d 6c 2c 61 70 70 6c |n/xhtml+xml,appl|  
000000d0 69 63 61 74 69 6f 6e 2f 78 6d 6c 3b 71 3d 30 2e |ication/xml;q=0.|  
000000e0 39 2c 2a 2f 2a 3b 71 3d 30 2e 38 0d 0a 41 63 63 |9,*/*;q=0.8..Acc|  
000000f0 65 70 74 2d 4c 61 6e 67 75 61 67 65 3a 20 65 6e |ept-Language: en|  
00000100 2d 55 53 2c 65 6e 3b 71 3d 30 2e 35 0d 0a 41 63 |.US,en;q=0.5..Ac|  
00000110 63 65 70 74 2d 45 6e 63 6f 64 69 6e 67 3a 20 67 |cept-Encoding: g|  
00000120 7a 69 70 2c 20 64 65 66 6c 61 74 65 0d 0a 52 65 |zip, deflate..Re|  
00000130 66 65 72 65 72 3a 20 68 74 74 70 3a 2f 2f 73 69 |ferer: http://si|  
00000140 65 63 69 2e 69 69 2e 75 6e 69 2e 77 72 6f 63 2e |eci.ii.uni.wroc.|  
00000150 70 6c 2f 69 6e 64 65 78 2e 70 68 70 0d 0a 43 6f |pl/index.php..Co|  
00000160 6e 74 65 6e 74 2d 54 79 70 65 3a 20 6d 75 6c 74 |ntent-Type: mult|  
00000170 69 70 61 72 74 2f 66 6f 72 6d 2d 64 61 74 61 3b |ipart/form-data;|  
00000180 20 62 6f 75 6e 64 61 72 79 3d 2d 2d 2d 2d 2d 2d |boundary=-----|  
00000190 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d |-----|  
000001a0 2d 2d 2d 2d 2d 2d 32 35 31 32 38 33 33 37 31 31 37 |-----25128337117|  
000001b0 37 38 38 30 36 32 32 38 38 32 31 32 35 32 31 34 |7880622882125214|  
000001c0 33 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74 |3..Content-Lengt|  
000001d0 68 3a 20 31 37 35 0d 0a 44 4e 54 3a 20 31 0d 0a |h: 175..DNT: 1..|  
000001e0 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 6b 65 65 70 |Connection: keep|  
000001f0 2d 61 6c 69 76 65 0d 0a 55 70 67 72 61 64 65 2d |-alive..Upgrade-|  
00000200 49 6e 73 65 63 75 72 65 2d 52 65 71 75 65 73 74 |Insecure-Request|  
00000210 73 3a 20 31 0d 0a 0d 0a 2d 2d 2d 2d 2d 2d 2d 2d |s: 1.....|  
00000220 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d |-----|  
00000230 2d 2d 2d 2d 2d 32 35 31 32 38 33 33 37 31 31 37 |-----25128337117|  
00000240 37 38 38 30 36 32 32 38 38 32 31 32 35 32 31 34 |7880622882125214|  
00000250 33 0d 0a 43 6f 6e 74 65 6e 74 2d 44 69 73 70 6f |3..Content-Dispo|  
00000260 73 69 74 69 6f 6e 3a 20 66 6f 72 6d 2d 64 61 74 |sition: form-dat|  
00000270 61 3b 20 6e 61 6d 65 3d 22 74 72 65 73 63 22 0d |a; name="tresc".|  
00000280 0a 0d 0a 74 65 73 74 32 0d 0a 2d 2d 2d 2d 2d 2d |...test2.....|  
00000290 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d |-----|
```

Jak widzimy znajduje się w nim zapytanie HTTP. Możemy przesłać je do sieci.ii.uni.wroc.pl za pomocą polecenia:

- `V1$> nc -q 3 sieci.ii.uni.wroc.pl 80 < http_request`

po czym możemy zaobserwować, że wpis został dodany.



Następnie za pomocą vim'a edytowałem zawartość zapytania HTTP, po czym ponownie je wysłałem za pomocą tego samego polecenia co wcześniej. Ten wpis również został dodany.

Bieżące wpisy

33473: 12-05-2020, 12:41:02

test3

2 Drugie zadanie do zaprezentowania

Na początku drugiego zadania za pomocą polecenia:

- V1\$> dig www.debian.org

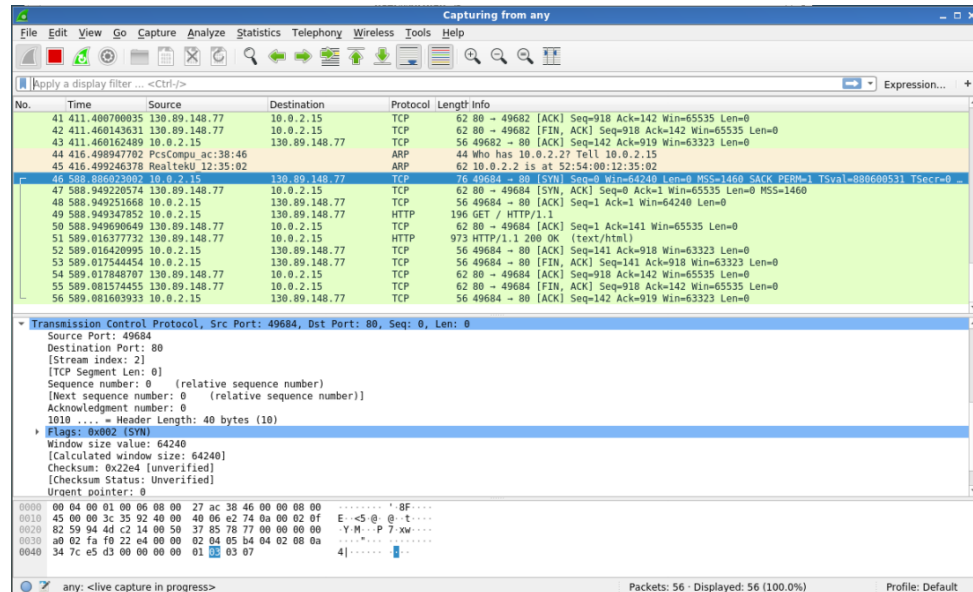
sprawdziłem adres IP domeny www.debian.org., który wynosi 130.89.148.77. Następnie w dwóch różnych konsolach uruchomiłem polecenia:

- V1C1\$> (while true; do netstat -tan | grep 130.89.148.77 ; done) | tee tcp_log
- V1C2\$> wget http://130.89.148.77/

W rezultacie plik tcp_log miał następującą zawartość:

```
user@virbian: ~  
tcp      0      1 10.0.2.15:49684      130.89.148.77:80      SYN_SENT  
tcp      0      1 10.0.2.15:49684      130.89.148.77:80      SYN_SENT  
tcp      0      1 10.0.2.15:49684      130.89.148.77:80      SYN_SENT  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      ESTABLISHED  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      FIN_WAIT2  
tcp      0      0 10.0.2.15:49684      130.89.148.77:80      FIN_WAIT2  
48,1 1%
```

Za pomocą Wiresharka sprawdziłem jak wyglądały kolejne pakiety:



W celu pobrania strony tworzone jest gniazdo na podstawie adresu IP 10.0.2.15 oraz portu 49684, jest to port źródłowy połączenia. Portem docelowym jest 80. Przyjrzyjmy się każdemu wysłanemu / otrzymanemu segmentowi TCP:

- Pierwszy segment zostaje wysłany od nas do 130.89.148.77, ma flagę SYN, wysyłane są bajty od 0. Po stronie klienta połączenie TCP przechodzi w stan SYN SENT. Po tym jak server otrzyma ten segment połączenie TCP po stronie serwera przejdzie w stan SYN RECEIVED.
- Drugim segmentem jest odpowiedź otrzymana od 130.89.148.77, która ma flagi SYN oraz ACK. Wysyłane są bajty od 0, potwierdzane są bajty do 1. Połączenie po stronie klienta przechodzi w stan ESTABLISHED.
- Trzeci segment zostaje wysłany do 130.89.148.77, ma flagę ACK, przesyła bajty od 1, potwierdza bajty do 1. Po tym jak server otrzyma ten segment jego połączenie również przejdzie w stan ESTABLISHED.
- Następnie wysłane zostaje zapytanie HTTP oraz otrzymujemy segment od 130.89.148.77, który ma flagę ACK. Przesyła bity od 1, oraz potwierdza bity do 141. Stan połączenia po obu stronach pozostaje bez zmian.
- Następnie otrzymujemy odpowiedź HTTP oraz wysyłamy segment do 130.89.148.77 z flagą ACK, wysyłamy bajty od 141, a potwierdzamy bajty do 918. Po tym jak server otrzyma ten segment jego połączenie przejdzie w stan CLOSE WAIT.

- Kolejny segment jest wysłany przez nas do 130.89.148.77 z flagami ACK oraz FIN. W tym momencie zaczynamy zamykać połączenie. Połączenie po stronie klienta przechodzi w stan FIN WAIT 1.
- Następne dwa segmenty przychodzą od 130.89.148.77, pierwszy z nich ma flagę ACK, drugi ma flagi FIN i ACK. W obu widzimy, że Seq = 918, oraz ACK = 142, czyli server wysłał wszystko do bajtu 918 oraz otrzymał wszystko do bajtu 142. Po otrzymaniu pierwszego segmentu połączenie po stronie klienta przechodzi w stan FIN WAIT 2, po otrzymaniu drugiego segmentu połączenie to przechodzi w stan TIME WAIT
- Ostatni segment jest wysłany do 130.89.148.77 z flagą ACK, potwierdza on, że otrzymaliśmy wszystko do bajtu 919. Po tym jak server otrzyma ten segment połączenie po jego stronie przejdzie w stan CLOSED. Po upływie timeouta połączenie po stronie klienta również przejdzie w stan CLOSED.

Wynika z tego, że strona klienta wykonuje aktywne otwarcie oraz aktywne zamknięcie. W pliku tcp_log mogliśmy zaobserwować wszystkie stany połączenia po stronie klienta z wyjątkiem FIN WAIT 1.