Sprawozdanie nr 5

Technologie sieciowe

Laboratorium piątek 9:15

Weronika Jasiak
236733

1 Cel ćwiczenia:

Przetestowanie programu serwera protokołu `server.pl`, a następnie napisanie własnego serwera w dowolnym języku i wykonanie następujących zadań:

1.1 Uruchom `server.pl`, przetestuj, zastanów się jak działa.

1.2 Nawiąż połączenie za pomocą przykładowych klientów z listy poprzedniej.

1.3 Nawiąż połączenie za pomocą przeglądarki internetowej.

1.4 Zmień skrypt tak aby wysyłał do klienta nagłówek jego żądania.

1.5 Zmień skrypt tak aby obsługiwał żądania klienta do prostego tekstowego serwisu WWW (kilka statycznych stron z wzajemnymi odwołaniami) zapisanego w pewnym katalogu dysku lokalnego komputera na którym uruchomiony jest skrypt serwera.

1.6 Przechwyć komunikaty do/od serwera za pomocą analizatora sieciowego `'sniffera'` - przeanalizuj ich konstrukcję.

2 Realizacja:

2.1 Poniżej zostanie przedstawiony listing programu `serwer.pl` i nastąpi jego omówienie.

```
use HTTP::Daemon;
use HTTP::Status;
#use IO::File;

my $d = HTTP::Daemon->new(
         LocalAddr => 'localhost',
         LocalPort => 8080,
     )|| die;

print "Please contact me at: <URL:", $d->url, ">\n";

while (my $c = $d->accept) {
    while (my $r = $c->get_request) {
        if ($r->method eq 'GET') {
            $file_s= "./index.html";    # index.html - jakis istniejacy plik
            $c->send_file_response($file_s);
        } else {
            $c->send_error(RC_FORBIDDEN)
        }
    }
    $c->close;
    undef($c);
}
```

Listing 1: Program `server3.pl`

Na początku program uruchamia serwer `HTTP` na porcie 8080. Następnie w nieskończonej pętli przyjmuje żądania. Jeśli żądanie jest typu `GET` to je obsługuje, a jeśli nie to wyświetla błąd 403 (dostęp zabroniony). Obsługa żądania polega na załadowaniu pliku `index.html` i wysłaniu go do przeglądarki.

```
C:\Users\weron\Desktop\curl-7.60.0-win64-mingw\bin>curl -v http://127.0.0.1:8080/
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to 127.0.0.1 (127.0.0.1) port 8080 (#0)
> GET / HTTP/1.1
> Host: 127.0.0.1:8080
> User-Agent: curl/7.60.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Thu, 07 Jun 2018 15:25:49 GMT
```

```
< Server: libwww-perl-daemon/6.01
< Content-Type: text/html
< Content-Length: 217
< Last-Modified: Thu, 07 Jun 2018 14:49:06 GMT
<
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Strona domowa</title>
</head>
<body>
<center>
<h1>Strona główna</h1>
<a href="substrona/1.html">Link do podstrony</a>
</center>
</body>
</html>* Connection #0 to host 127.0.0.1 left intact
```

Listing 1a: Program `server3.pl`, odpowiedź na żądanie GET

```
C:\Users\weron\Desktop\curl-7.60.0-win64-mingw\bin>curl --data "post" http://127.0.0.1:8080/
<title>403 Forbidden</title>
<h1>403 Forbidden</h1>
```

Listing 1b: Program `server3.pl`, odpowiedź na żądanie POST

2.2 Za pomocą klientów z poprzedniej listy nie jest możliwe nawiązanie połączenia z tym serwerem. Wynika to z tego, że klienci korzystają z protokołu `UDP`, a serwer z protokołu `HTTP`.

2.3 Po otworzeniu za pomocą przeglądarki wyświetli się zawartość wspomniana w pkt. 2.1, czyli plik `index.html`.

2.4 Serwer został napisany w języku `PERL`. Po wywołaniu żądania `POST` wyświetla nagłówek żądania klienta.

```
C:\Users\weron\Desktop\curl-7.60.0-win64-mingw\bin>curl --data "post" http://127.0.0.1:8080/
POST / HTTP/1.1
Accept: */*
Host: 127.0.0.1:8080
User-Agent: curl/7.60.0
Content-Length: 4
Content-Type: application/x-www-form-urlencoded

post
```

Listing 2: Nagłówek żądania klienta, otrzymany w wyniku żądania POST

```
use HTTP::Daemon;
use HTTP::Status;
use IO::File;

  my $d = HTTP::Daemon->new(
           LocalAddr => 'localhost',
           LocalPort => 8080,
       )|| die;

  print "Please contact me at: <URL:", $d->url, ">\n";

  while (my $c = $d->accept) {
    while (my $r = $c->get_request) {
      $response = HTTP::Response->new(300,'OK');
      $response->content($r->as_string);
      $c->send_response($response);
```

```
    }
    $c->close;
    undef($c);
  }
```

Listing 2a: Kod przedstawiający `server1.pl`, który umożliwia wysłanie nagłówku

2.5 Poniżej znajduje się listing przedstawiający kod serwera:

```
use HTTP::Daemon;
use HTTP::Status;
use IO::File;

  my $d = HTTP::Daemon->new(
           LocalAddr => 'localhost',
           LocalPort => 8080,
       )|| die;

  while (my $c = $d->accept) {
    while (my $r = $c->get_request) {
      if ($r->method eq 'GET') {
        $file_s = $r->uri;
        if ($file_s eq "/") {
          $file_s = "./index.html";
        }
        $c->send_file_response($WEBDIR.$file_s);
      } else {
        $c->send_error(RC_FORBIDDEN)
      }
    }
    $c->close;
    undef($c);
  }
```

Listing 3: Kod przedstawiający `server2.pl`

2.6 Do przechwycenia pakietów przesyłanych pomiędzy klientem, a serwerem, użyto programu `RawCap`, który zapisuje informacje do pliku, a ten zostaje otworzony przez `Wireshark`. Przechwycono dwa pakiety:

a) żądanie klienta:

```
GET /index.html HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/66.0.3359.181 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*
;q=0.8
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Accept-Encoding: gzip, deflate
If-Modified-Since: Thu, 07 Jun 2018 18:16:18 GMT
[HTTP request 1/1]
```

Listing 4: Przechwycone żądanie

b) odpowiedź serwera

```
HTTP/1.1 200 OK
Date: Thu, 07 Jun 2018 18:24:28 GMT
Server: libwww-perl-daemon/6.01
Last-Modified: Thu, 07 Jun 2018 18:16:18 GMT
[HTTP response 1/1]
[Time since request: 0.000997000 seconds]
```

Listing 5: Przechwycona odpowiedź serwera

3 Wnioski:

Przetestowany program działał stabilnie, nawet pomimo wprowadzonych poprawek. Skrypt nie działa zbyt szybko ze względu na brak zrównoleglenia żądań, lecz bardzo dobrze obrazuje schemat działania serwera `HTTP`.