Lista 5
Technologie sieciowe

Patryk Majewski
250134

1 Opis zadania

Na podstawie dostarczonego skryptu należy:

- przeanalizować działanie serwera
- zmodyfikować skrypt, aby serwer zwracał użytkownikowi nagłówek wysłanego przez niego żądania
- zmodyfikować skrypt, aby serwer mógł obsługiwać żądania dla prostego serwisu WWW
- przeanalizować komunikaty przesyłane do i od serwera

2 Działanie skryptu

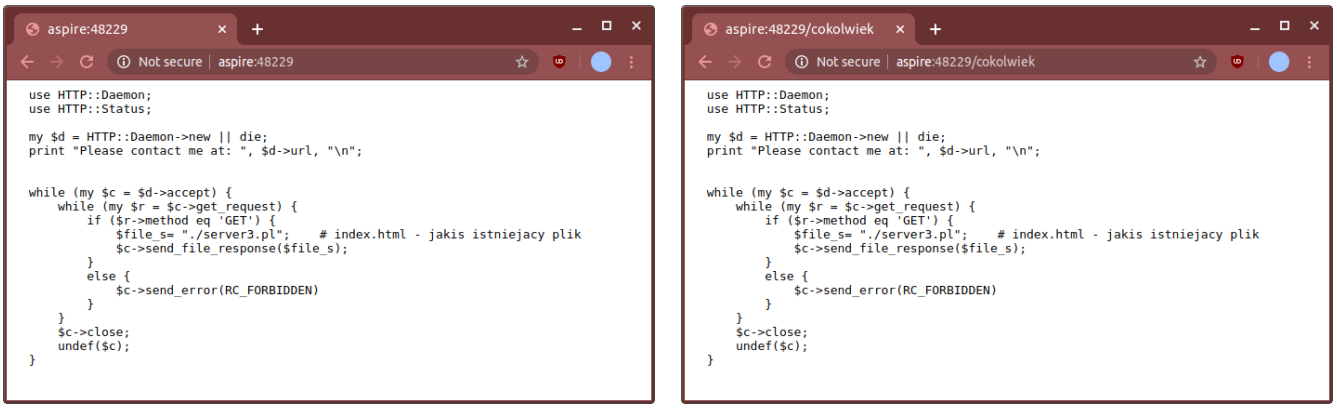
Dostarczony skrypt musiał ulec drobnej modyfikacji z powodu problemów z jego uruchomieniem:

```
my $d = HTTP::Daemon->new || die;
print "Please contact me at: ", $d->url, "\n";
```

Instancja obiektu serwera jest tworzona metodą bez argumentów, tak żeby program sam mógł wybrać wolny port. Następnie użytkownik informowany jest o adresie, na który może wysyłać swoje żądania, na przykład:

```
$ perl server3.pl
Please contact me at: http://aspire:37635/
```

Po ustaleniu połączenia, w nieskończonej pętli (ponieważ nie ustawiliśmy timeoutu) serwer oczekuje na żądania i, jeśli są one typu GET, odpowiada na nie wysłaniem wyspecyfikowanego pliku. Aby wysłać żądanie z poziomu przeglądarki internetowej, musimy wejść pod podany nam adres. Treść zapytania nie jest w żaden sposób przetwarzana, dlatego w adresie możemy umieścić dowolną ścieżkę po znaku '/' i nadal otrzymać tę samą odpowiedź.



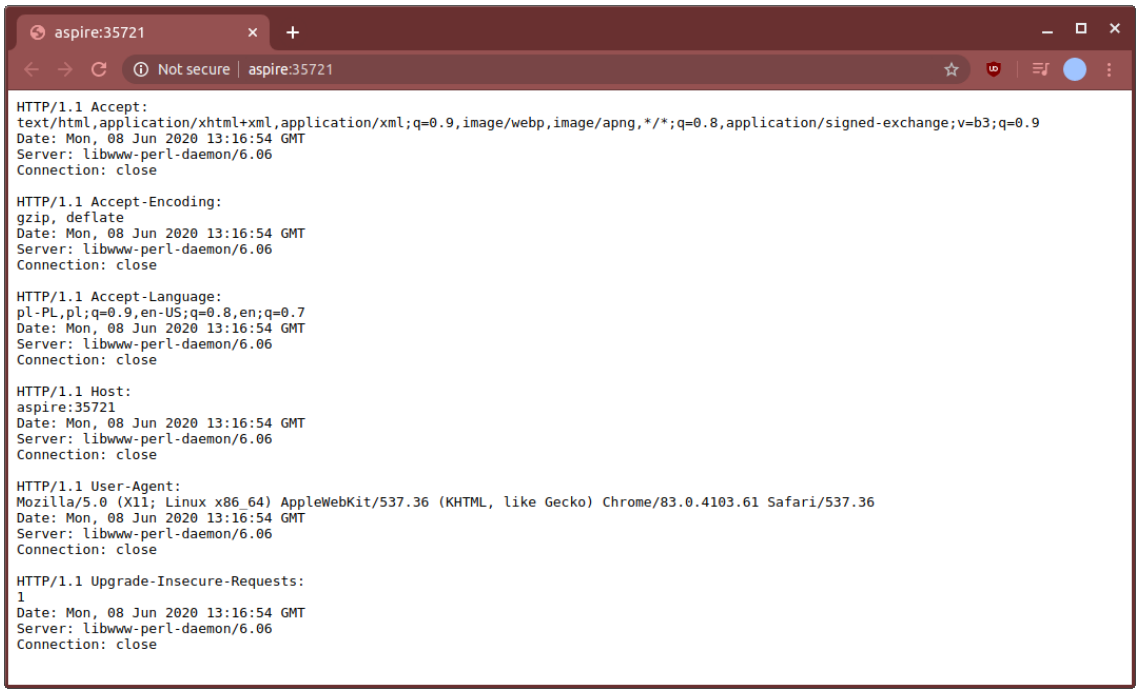
Rysunek 1: Odpowiedź serwera będąca plikiem źródłowym jego skryptu. Dodanie ścieżki `/cokolwiek` nie powoduje różnicy w działaniu.

3 Zwracanie nagłówka

Lokalna zmienna `$r` w najbardziej wewnętrznej pętli jest obiektem klasy `HTTP::Request`. Nagłówek HTTP składa się z kliku pól. Nasz obiekt posiada metody `header_field_names` (zwracającą nazwy pól) oraz `header` (zwracającą zawartość pola nagłówka o podanej nazwie). Zmienimy więc część kodu, w której wcześniej zwracaliśmy plik, umieszczając tam następujący fragment:

```
if ($r->method eq 'GET') {
  foreach ($r->header_field_names) {
    $c->send_response($_ . ":\n" . $r->header($_));
  }
}
```

gdzie `$_` jest obiektem zwróconym przez iterator (aktualnie rozpatrywaną nazwą pola).



Rysunek 2: Zwrócone elementy nagłówka wysłanego przez nas żądania. Faktycznie interesujące nas dane znajdują się w pierwszych dwóch liniach każdego fragmentu, reszta to nagłówek odpowiedzi serwera.

4 Obsługa serwisu

Na potrzeby testu stworzony został prosty projekt zawierający treści list zadań laboratoryjnych. Struktura plików:

```
website
├── index.html
├── aisd
│   └── aisd.html
└── ts
    ├── ts.html
    └── listy
        ├── lista4.html
        └── lista5.html
```

Obiekt `$r` wspomniany w poprzedniej sekcji posiada także pole `uri`, czyli identyfikator zasobu, który chce uzyskać nadawca żądania. W przypadku naszej strony chcemy na przykład, żeby prośba o zasób `/ts/lista4` powodowała wysłanie pliku `lista4.html`.

Serwerowy skrypt zmodyfikujemy następująco:

1. Jeśli ścieżka podana w żądaniu zaczyna się od `ts` oraz
   (a) występuje słowo `lista`, spróbuj przesłać stronę z listą o podanym numerze (jeśli istnieje).
   (b) nie występuje słowo `lista`, wyświetl `ts.html`.
2. Jeśli ścieżka zaczyna się od `aisd`, wyślij `aisd.html`.
3. Jeśli ścieżka jest pusta, wyślij `index.html`.
4. W przeciwnym wypadku wyślij błąd.

Zmieniona wewnętrzna pętla:

```
if ($r->method eq 'GET') {
    $path = $r->uri->path;
    if (index($path, "/ts") == 0) {
        $real_path = "./website/ts";
        if (index($path, "/lista") == 3) {
            $real_path = $real_path . "/listy/" . substr($path, 4) . ".html";
            $c->send_file_response($real_path);
        }
        elsif (length($path) == 3) {
            $real_path = $real_path . "/ts.html";
            $c->send_file_response($real_path);
        }
        else {
            $c->send_error(RC_FORBIDDEN);
        }
    }
    elsif (index($path, "/aisd") == 0 && length($path) == 5) {
        $c->send_file_response("./website/aisd/aisd.html");
    }
    elsif (length($path) == 1) {
        $c->send_file_response("./website/index.html");
    }
    else {
        $c->send_error(RC_FORBIDDEN);
    }
}
```

Rysunek 3: Odpowiedzi serwera na prośby o zasoby `/aisd` oraz `/ts/lista5`.

5 Analiza komunikatów

Celem zbadania konstrukcji komunikatów ponownie skorzystamy z Wiresharka. Ustawimy nasłuchiwanie w interfejsie loopback z filtrem pozostawiającym tylko komunikaty HTTP użytkujące port, na którym uruchomiony jest serwer.

5.1 Żądanie

```
GET /aisd HTTP/1.1
Host: aspire:53919
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
    /83.0.4103.61 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,
    application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
```

Najbardziej zauważalny jest format komunikatu: podczas gdy nagłówki niższych warstw miały postać liczb na odpowiednich miejscach, nagłówek HTTP przesyłany jest w zwyczajnym ASCII. Poszczególne linijki oddzielone są kombinacją znaków carriage return (`\r`, `0x0d`) i line feed (`\n`, `0x0a`), dla czytelności usuniętą z przykładu.

Pierwsza linia zawiera treść naszego zapytania. Rozpoczyna ją pole metody (na przykład GET, POST, PUT, DELETE). W naszym przypadku korzystamy tylko z zapytań typu GET, zatem kolejnym elementem żądania jest identyfikator zasobu, który chcemy pozyskać. Ostatnią częścią jest wersja protokołu HTTP.

Niezbyt intuicyjnie, dopiero kolejne linijki składają się na nagłówek komunikatu. Jak wspomniano wcześniej, zawiera on w sobie dane podzielone na pola. Każda linia jest postaci `Nazwa-Pola: Wartość`. W naszym przypadku zauważyć można na przykład:

- `Host` – określa adres serwera, na którym znajduje się pożądany zasób
- `Connection` – informuje serwer, czy powinien zamykać połączenie zaraz po wysłaniu odpowiedzi
- `Accept` – informuje serwer, jakie typy mediów (standardu MIME) jest w stanie zrozumieć klient
- `Accept-Encoding` – informuje serwer, jakie sposoby kodowania (albo kompresji) danych jest w stanie zrozumieć klient
- `Accept-Language` – informuje serwer o preferowanym języku klienta
- `User-Agent` – informuje serwer o urządzeniu i programie, z którego wysłano żądanie

5.2 Odpowiedź

```
HTTP/1.1 200 OK
Date: Mon, 08 Jun 2020 22:11:13 GMT
Server: libwww-perl-daemon/6.06
Content-Type: text/html
Content-Length: 97
Last-Modified: Mon, 08 Jun 2020 15:30:53 GMT
```

Odpowiedź na nasze zapytanie składa się z trzech sekcji: statusu, nagłówka i faktycznych danych. Jest ono zapisane w sposób podobny do żądania.

Sekcja statusu zawiera wersję HTTP oraz kod – w przypadku zapytania GET `200 OK` oznacza, że zasób został znaleziony i jest przesłany wraz z komunikatem.

Nagłówek zawiera kolejno informacje o czasie wysłania wiadomości, rodzaju serwera (w pewnym sensie analogia do `User-Agent` z żądania), typie danych (powinien być zgodny z `Accept`), ich długości i czasie ostatniej modyfikacji zasobu.