

Technologie sieciowe

Lista 5

HTTP

Michał Kalina
250088

Zadanie 1

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

my $d = HTTP::Daemon->new(                                # domena HTTP
    LocalAddr => 'localhost',
    LocalPort => 4000,
)|| die;

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

while (my $c = $d->accept) {                                # akceptuj połączenia
    while (my $r = $c->get_request) {                        # pobieranie żądania
        if ($r->method eq 'GET') {                          # jeśli metoda to GET

            $file_s = "/index.html";                        # index.html - jakiś istniejący plik
            $c->send_file_response($file_s);                # wyślij w odpowiedzi plik file_s

        }
        else {                                              # jeśli błąd
            $c->send_error(RC_FORBIDDEN)

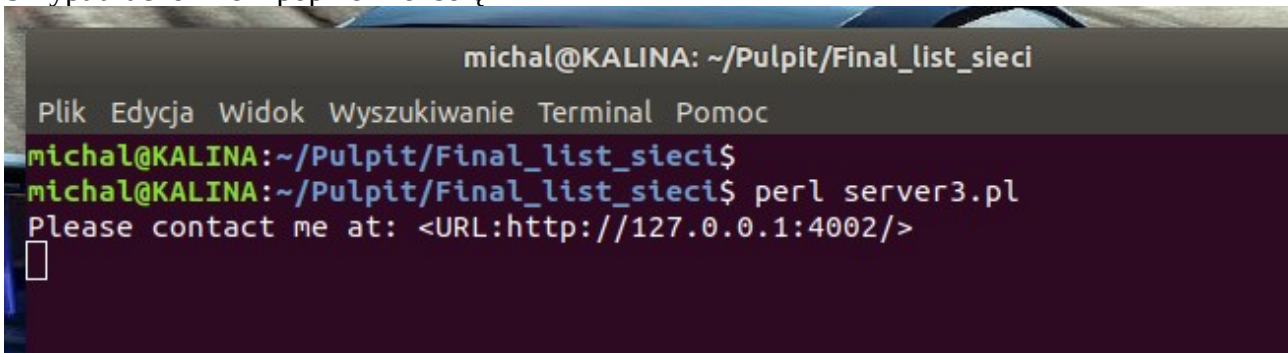
        }
    }
    $c->close;                                              # zamknij połączenie
    undef($c);                                              # usuń c
}
```

Skrypt jest serwerem HTTP, który odpowiada na żądanie GET i zwraca stronę index.html lub RC_FORBIDDEN, czyli błąd 403 forbidden.

Rozwinięcie skrótów:

- \$c => connection
- \$r => request
- \$d => daemon (proces lub program wykonywany wewnątrz środowiska)

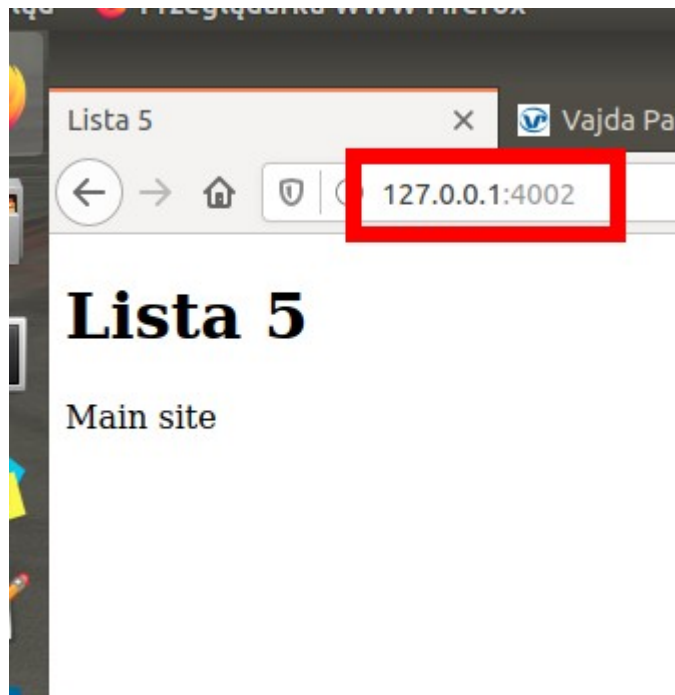
Skrypt uruchomiłem poprzez konsolę



The screenshot shows a terminal window with the title bar 'michal@KALINA: ~/Pulpit/Final_list_sieci'. The terminal has a menu bar with 'Plik', 'Edycja', 'Widok', 'Wyszukiwanie', 'Terminal', and 'Pomoc'. The prompt is 'michal@KALINA:~/Pulpit/Final_list_sieci\$'. The user has entered 'perl server3.pl'. The output of the script is 'Please contact me at: <URL:http://127.0.0.1:4002/>' followed by a cursor.

Zadanie 2

Stronę można odwiedzić w przeglądarce internetowej



Zadanie 3

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

my $d = HTTP::Daemon->new(
    LocalAddr => 'localhost',
    LocalPort => 4016,
)|| 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') {
            my $data = $r->as_string;
            my $r = HTTP::Response->new(200);
            $r->content($data);
            $r->header("Content-Type" => "text/plain");
            $c->send_response($r);
        }
        else {
            $c->send_error(RC_FORBIDDEN);
        }
    }
    $c->close;
```

```
}  
    undef($c);  
}
```

W odpowiedzi na żądanie GET tworzę odpowiedź 200(czyli że jest OK) i ustawiam typ MIME(*Multipurpose Internet Mail Extensions* - dwuczęściowy identyfikator formatu plików w Internecie) na *text/plain*(dane tekstowe).

Odpowiedzią jest treść żądania w formie napisu.

Na końcu wysyłam wszystko do klienta.



Connection: keep alive, połączenie będzie utrzymywane oraz będzie czekało na kolejne zapytanie.

Accept określa listy akceptowalnych przez przeglądarkę typów MIME dokumentu, oraz opcjonalnie hierarchii każdego typu.

Accept-Encoding określa kodowanie, za pomocą którego zostanie przesłana zawartość.

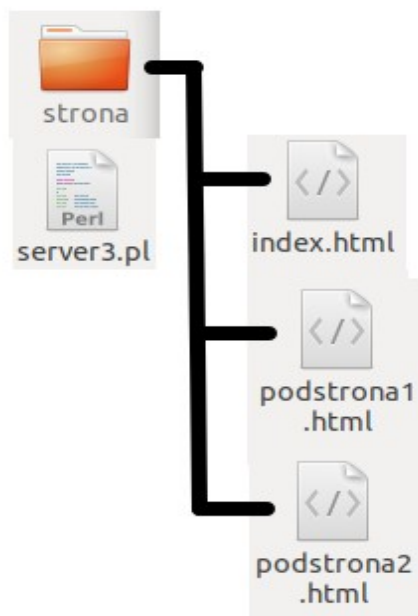
Accept-Language określa preferowany język

Upgrade-Insecure-Requests wysyła do serwera sygnał wyrażający preferencje klienta dotyczące zaszyfrowanej i uwierzytelnionej odpowiedzi oraz że może on skutecznie obsłużyć dyrektywę CSP(Content Security Policy) dotyczącą niezabezpieczonych żądań uaktualnienia.

W zależności od przeglądarki, zawartość strony może się różnić, np. w przeglądarce Firefox nie będą wyświetlane informacje Sec-Fetch-.....

Zadanie 4

Stworzyłem foldery i zamieściłem w nich pliki html jak na załączonym obrazku



Skrypt:

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

my $d = HTTP::Daemon->new(
    LocalAddr => 'localhost',
    LocalPort => 2030,
)|| die;

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

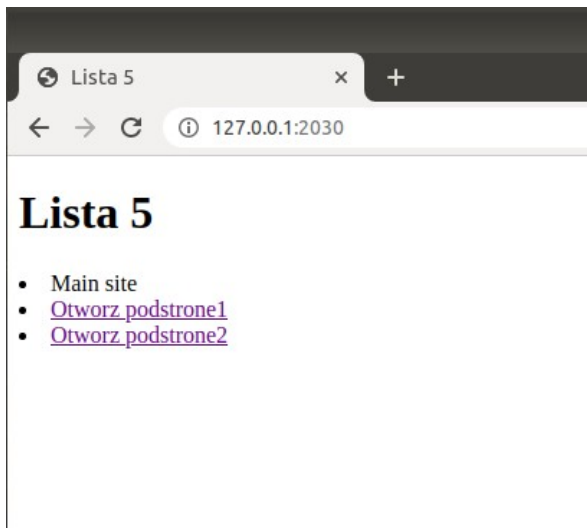
while (my $c = $d->accept) {
    while (my $r = $c->get_request) {
        if ($r->method eq 'GET') {

            my $my_uri=$r->uri;

            if ($my_uri eq "/") {
                $my_uri = "/index.html";
            }

            my $subpage = "strona" . $my_uri;

            if ( -e $subpage) {
                $c->send_file_response($subpage);
            }
            else {
                $c->send_error(RC_NOT_FOUND);
            }
        }
        else {
            $c->send_error(RC_FORBIDDEN)
        }
    }
    $c->close;
    undef($c);
}
```



Gdy odwołujemy się do strony głównej otwierany jest plik *index.html*.
Po wybraniu odpowiedniego linku przekierowywani jesteśmy do jednej z pozostałych stron.
W przypadku wpisania złego adresu zostanie pokazany komunikat *404 Not Found*.

Zadanie 5

Analizy sygnałów dokonałem przy pomocy programu *WireShark*

No.	Time	Source	Destination	Protocol	Length	Info
108	15.174137860	127.0.0.1	127.0.0.1	HTTP	476	HTTP/1.1 200 OK (text/html)
195	40.525797741	127.0.0.1	127.0.0.1	HTTP	783	GET / HTTP/1.1
210	40.527517013	127.0.0.1	127.0.0.1	HTTP	476	HTTP/1.1 200 OK (text/html)
213	40.683587212	127.0.0.1	127.0.0.1	HTTP	783	GET / HTTP/1.1
226	40.684530792	127.0.0.1	127.0.0.1	HTTP	476	HTTP/1.1 200 OK (text/html)
228	40.830960066	127.0.0.1	127.0.0.1	HTTP	783	GET / HTTP/1.1
237	40.832102864	127.0.0.1	127.0.0.1	HTTP	476	HTTP/1.1 200 OK (text/html)
239	40.974234621	127.0.0.1	127.0.0.1	HTTP	783	GET / HTTP/1.1
248	40.975115622	127.0.0.1	127.0.0.1	HTTP	476	HTTP/1.1 200 OK (text/html)
250	41.186633431	127.0.0.1	127.0.0.1	HTTP	783	GET / HTTP/1.1

▶ Frame 195: 783 bytes on wire (6264 bits), 783 bytes captured (6264 bits) on interface 0 ▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00) ▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1 ▶ Transmission Control Protocol, Src Port: 39224, Dst Port: 2030, Seq: 4277, Ack: 3547, Len: 717 ▶ Hypertext Transfer Protocol	
--	--

0000	00 00 00 00 00 00 00 00	00 00 00 00 08 00 45 00E.
0010	03 01 4f 00 40 00 40 06	ea f4 7f 00 00 01 7f 00	..0_@_@.....
0020	00 01 99 38 07 ee e4 bf	42 a4 71 29 09 90 80 18	...8....B.q)....
0030	02 00 00 f6 00 00 01 01	08 0a d1 77 f3 ad d1 77w...w
0040	90 a6 47 45 54 20 2f 20	48 54 54 50 2f 31 2e 31	..GET / HTTP/1.1
0050	0d 0a 48 6f 73 74 3a 20	31 32 37 2e 30 2e 30 2e	..Host: 127.0.0.
0060	31 3a 32 30 33 30 0d 0a	43 6f 6e 6e 65 63 74 69	1:2030.. Connecti
0070	6f 6e 3a 20 6b 65 65 70	2d 61 6c 69 76 65 0d 0a	on: keep -alive..
0080	43 61 63 68 65 2d 43 6f	6e 74 72 6f 6c 3a 20 6d	Cache-Co ntrol: m
0090	61 78 2d 61 67 65 3d 30	0d 0a 55 70 67 72 61 64	ax-age=0 ..Upgrad
00a0	65 2d 49 6e 73 65 63 75	72 65 2d 52 65 71 75 65	e-Insecu re-Reque
00b0	73 74 73 3a 20 31 0d 0a	55 73 65 72 2d 41 67 65	sts: 1.. User-Age

Io.	Time	Source	Destination	Protocol	Length	Info
108	15.174137860	127.0.0.1	127.0.0.1	HTTP	476	HTTP/1.1 200 OK (text/html)
195	40.525797741	127.0.0.1	127.0.0.1	HTTP	783	GET / HTTP/1.1
210	40.527517013	127.0.0.1	127.0.0.1	HTTP	476	HTTP/1.1 200 OK (text/html)
213	40.683587212	127.0.0.1	127.0.0.1	HTTP	783	GET / HTTP/1.1
226	40.684530792	127.0.0.1	127.0.0.1	HTTP	476	HTTP/1.1 200 OK (text/html)
228	40.830960066	127.0.0.1	127.0.0.1	HTTP	783	GET / HTTP/1.1
237	40.832102864	127.0.0.1	127.0.0.1	HTTP	476	HTTP/1.1 200 OK (text/html)
239	40.974234621	127.0.0.1	127.0.0.1	HTTP	783	GET / HTTP/1.1
248	40.975115622	127.0.0.1	127.0.0.1	HTTP	476	HTTP/1.1 200 OK (text/html)
250	41.186633431	127.0.0.1	127.0.0.1	HTTP	783	GET / HTTP/1.1
Transmission Control Protocol, Src Port: 39224, Dst Port: 2030, Seq: 4277, Ack: 3547, Len: 717						
Hypertext Transfer Protocol						
GET / HTTP/1.1\r\n						
Host: 127.0.0.1:2030\r\n						
Connection: keep-alive\r\n						
Cache-Control: max-age=0\r\n						
Upgrade-Insecure-Requests: 1\r\n						
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) snap Chromium/83.0.4103.97 Chrome/83.0.4103.97 Safari/537.36\r\n						
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\r\n						
Sec-Fetch-Site: none\r\n						
Sec-Fetch-Mode: navigate\r\n						
Sec-Fetch-User: ?1\r\n						
Sec-Fetch-Dest: document\r\n						
Accept-Encoding: gzip, deflate, br\r\n						
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7\r\n						
Cookie: csrftoken=PrwMscflfI39o5RW5JzEDGdwV37km1wP9V0a1AxCDWA0nhRDxUNjHtnGdz7GCiva\r\n						
If-Modified-Since: Tue, 09 Jun 2020 13:09:40 GMT\r\n						
0000 00 00 00 00 00 00 00 00 00 08 00 45 00E						
0010 03 01 4f 00 40 00 40 06 ea f4 7f 00 00 01 7f 00 ..O.@. .						
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1						
Transmission Control Protocol, Src Port: 39224, Dst Port: 2030, Seq: 4277, Ack: 3547, Len: 717						
Source Port: 39224						
Destination Port: 2030						
[Stream index: 2]						
[TCP Segment Len: 717]						
Sequence number: 4277 (relative sequence number)						
[Next sequence number: 4994 (relative sequence number)]						
Acknowledgment number: 3547 (relative ack number)						
1000 = Header Length: 32 bytes (8)						
Flags: 0x018 (PSH, ACK)						
Window size value: 512						
[Calculated window size: 512]						
[Window size scaling factor: -1 (unknown)]						
Checksum: 0x00f6 [unverified]						
[Checksum Status: Unverified]						
Urgent pointer: 0						
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps						
[SEQ/ACK analysis]						
[Timestamps]						
TCP payload (717 bytes)						
Hypertext Transfer Protocol						
0000 00 00 00 00 00 00 00 00 00 08 00 45 00E						
0010 03 01 4f 00 40 00 40 06 ea f4 7f 00 00 01 7f 00 ..O.@. .						
0020 00 01 99 38 07 ee e4 bf 42 a4 71 29 09 90 80 18 ...8...B.q)...						
0030 02 00 00 f6 00 00 01 01 08 0a d1 77 f3 ad d1 77w..w						
0040 90 a6 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 ..GET / HTTP/1.1						
0050 0d 0a 48 6f 73 74 3a 20 31 32 37 2e 30 2e 30 2e ..Host: 127.0.0.						
0060 31 3a 32 30 33 30 0d 0a 43 6f 6e 6e 65 63 74 69 1:2030.. Connecti						
0070 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 65 0d 0a on: keep -alive..						
0080 43 61 63 68 65 2d 43 6f 6e 74 72 6f 6c 3a 20 6d Cache-Co ntrol: m						
0090 61 78 2d 61 67 65 3d 30 0d 0a 55 70 67 72 61 64 ax-age=0 ..Upgrad						

Widać że port to 2030 czyli się zgadza.

Widać że w 4 warstwie są 2 flagi (PSH, ACK):

PSH: Flaga Push informuje stos sieciowy odbiornika, aby „wrzucił” dane bezpośrednio do gniazda odbiorczego i aby nie czekał na kolejne pakiety.

ACK: Flaga potwierdzenia - służy do potwierdzenia pomyślnego odebrania pakietu.

W pakiecie jest także zawarta informacja o długości (03 01)₁₆ i inne informacje które opisywałem przy okazji poprzednich list.

W 3 warstwie znajduje się flaga (40 00), czyli *Don't Fragment*.

Zapytanie:

No.	Time	Source	Destination	Protocol	Length	Info
139	26.373995534	127.0.0.1	127.0.0.1	HTTP	803	GET / HTTP/1.1
143	26.374853183	127.0.0.1	127.0.0.1	HTTP	642	HTTP/1.1 200 OK (text/html)
148	26.452894542	127.0.0.1	127.0.0.1	HTTP	665	GET /favicon.ico HTTP/1.1
162	26.453641958	127.0.0.1	127.0.0.1	HTTP	121	HTTP/1.1 404 Not Found (text/html)
448	28.334984251	127.0.0.1	127.0.0.1	HTTP	858	GET /podstrona1.html HTTP/1.1
464	28.335757819	127.0.0.1	127.0.0.1	HTTP	381	HTTP/1.1 200 OK (text/html)

Frame 139: 803 bytes on wire (6424 bits), 803 bytes captured (6424 bits) on interface 0

Linux cooked capture

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

Transmission Control Protocol, Src Port: 43974, Dst Port: 2030, Seq: 1, Ack: 1, Len: 735

Hypertext Transfer Protocol

GET / HTTP/1.1\r\n

Host: 127.0.0.1:2030\r\n

Connection: keep-alive\r\n

Upgrade-Insecure-Requests: 1\r\n

User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) snap Chromium/83.0.4103.97 Chrome/83.0.4103.97 Safari/

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\r\n

Sec-Fetch-Site: none\r\n

Sec-Fetch-Mode: navigate\r\n

Sec-Fetch-User: ?1\r\n

Sec-Fetch-Dest: document\r\n

Accept-Encoding: gzip, deflate, br\r\n

Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7\r\n

Cookie: csrftoken=2m8UmyZjifej21Yu0TbkYthP6hTfr5m0FAo111QTw2metro046tCPQyaLCln02; sessionId=mwdo15uhx1fuowu87akmgn91ajwynosw\r\n

If-Modified-Since: Tue, 09 Jun 2020 13:09:40 GMT\r\n

Korzystamy z metody GET. Adres nadawcy i odbiorcy jest taki sam. Zawarta jest tu także informacja o sposobie w jaki serwer ma komunikować się z naszą przeglądarką.

Odpowiedź:

Linux cooked capture

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

Transmission Control Protocol, Src Port: 2030, Dst Port: 44066, Seq: 773, Ack: 1523, Len: 410

[8 Reassembled TCP Segments (591 bytes): #334(17), #336(37), #338(33), #340(25), #342(21), #344(46), #346(2), #348(410)]

Hypertext Transfer Protocol

HTTP/1.1 200 OK\r\n

Date: Sun, 14 Jun 2020 07:38:28 GMT\r\n

Server: libwww-perl-daemon/6.01\r\n

Content-Type: text/html\r\n

Content-Length: 410\r\n

Last-Modified: Tue, 09 Jun 2020 13:09:40 GMT\r\n

\r\n

[HTTP response 2/3]

[Time since request: 0.000985838 seconds]

[\[Prev request in frame: 314\]](#)

[\[Prev response in frame: 330\]](#)

[\[Request in frame: 332\]](#)

[\[Next request in frame: 350\]](#)

[\[Next response in frame: 354\]](#)

[Request URI: http://127.0.0.1:2030/]

W odpowiedzi widać: *HTTP/1.1 200 OK\r\n* czyli wiemy że nie wystąpiły błędy. 1.1 to wersja protokołu. Kod 200 oznacza że zapytanie zostało pozytywnie obsłużone. Następnie data i serwer. Na końcu znajduje się typ danych(nie widać) *text/html*, znajduje się w nim strona html. W warstwie transportowej widać że odpowiedź przyszła w 8 ramkach(*[8 Reassembled TCP Segments...]*), poskładanych w całość dzięki protokołowi TCP.