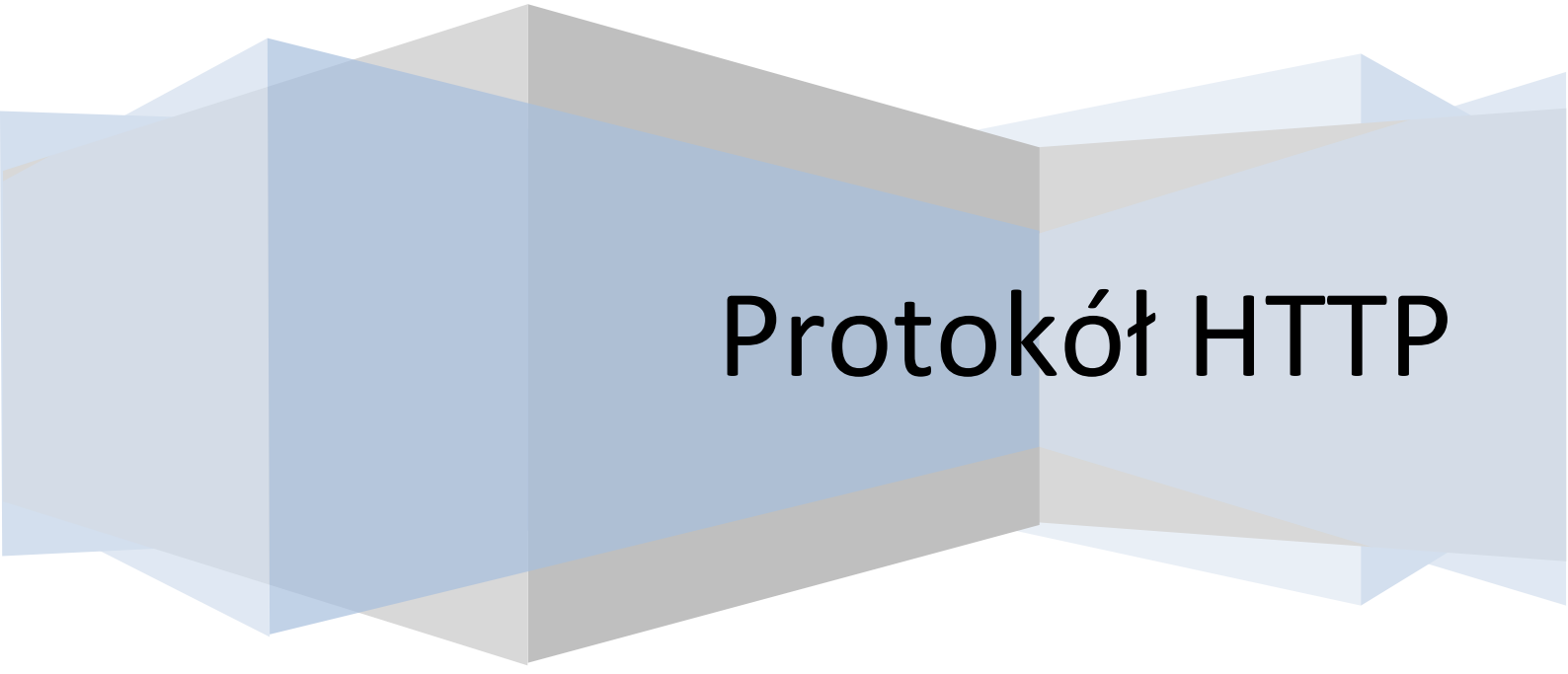
Politechnika Krakowska

Katedra Automatyki i Technik Informacyjnych

Laboratorium Sieci Komputerowych

2010/2011



Protokół HTTP

1. Wprowadzenie

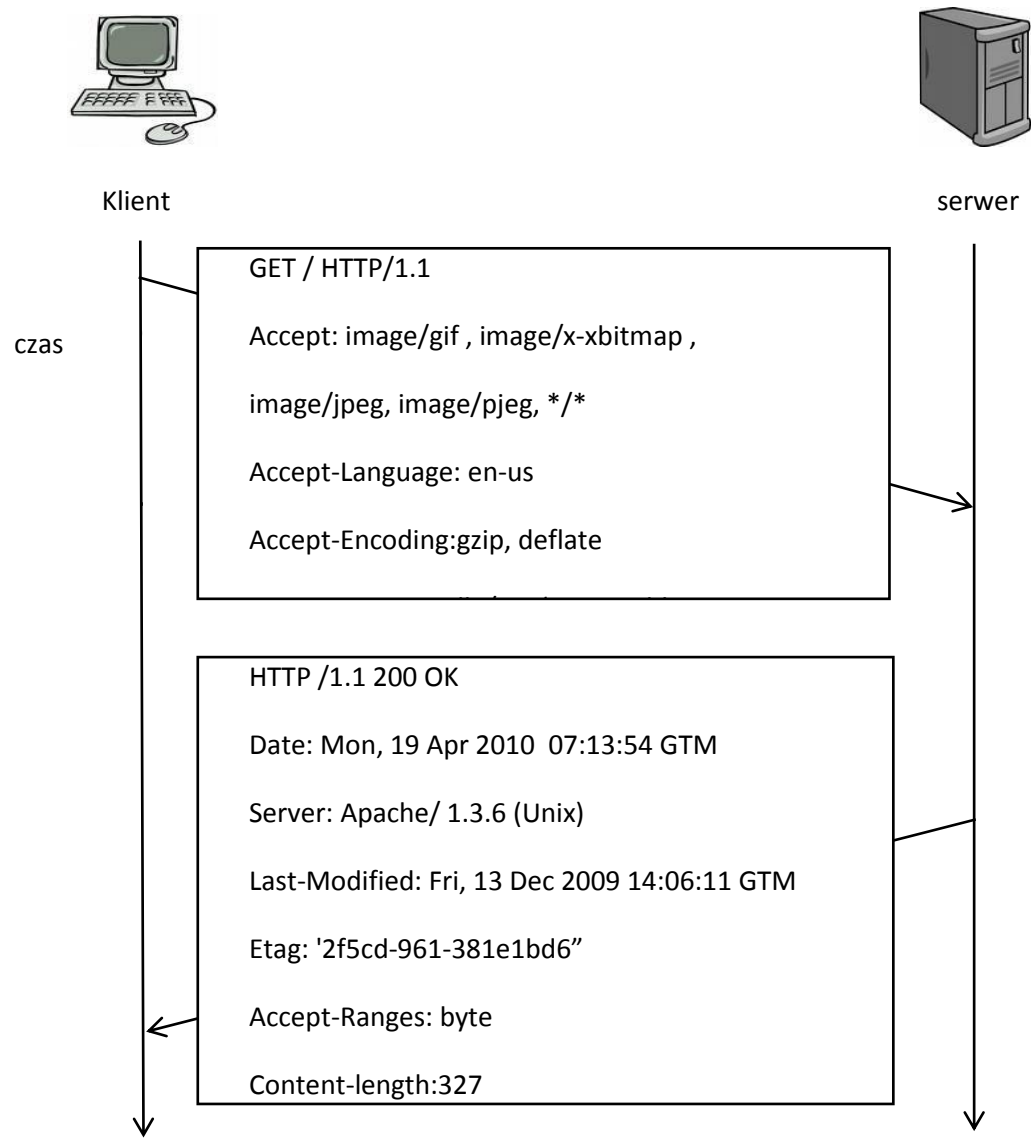
HTTP (ang. Hypertext Transfer Protocol – protokół przesyłania dokumentów hipertekstowych) jest protokołem zaprojektowanym pierwotnie do przesyłania danych sieci WWW, jednak od czasu powstania został rozbudowany i zyskał wiele nowych zastosowań. W swojej pierwszej wersji protokół HTTP w ramach jednego połączenia pozwalał klientowi na pobranie zasobu z komputera serwera na podstawie jego lokalizacji. W swojej obecnej wersji pozwala na przesyłanie wielu żądań w ramach jednego połączenia TCP (aby zwiększyć wydajność), pozostawiono jednak możliwość wykorzystania go w wersji z jednym żądaniem na połączenie (nazywanej bezpołączeniową). Fakt ten ułatwia zadanie zapewniania anonimowości klientów. Obecną definicję HTTP stanowi RFC 2616. HTTP standardowo korzysta z protokołu **TCP** i portu nr **80**. Wiadomości są przesyłane **otwartym tekstem**.

Model komunikacji wykorzystywany w HTTP, gdzie jednostką komunikacji jest wysłanie pewnych danych i synchroniczne odebranie odpowiedzi, nazywany jest modelem **żądanie – odpowiedź** lub **klient – serwer**. Strona wydająca żądania nazywana jest klientem, a odpowiadająca - serwerem. W takim modelu komunikacji trudniej jest zapewnić anonimowość nadawcy (czyli klienta HTTP, użytkownika WWW) niż w przypadku gdy odpowiedzi w ogóle nie są wymagane. Sytuacja jest jednak lepsza od ogólnego przypadku, w którym każda ze stron może w dowolnym momencie zainicjować komunikację.

HTTP powstał w 1991 roku. Wtedy to uznano, że potrzebny jest nowy protokół, przeznaczony specjalnie do przekazywania hipertekstu. Powstała wówczas pierwsza, prototypowa implementacja protokołu, znana jako HTTP 0.9. Późniejsze wersje zachowują zgodność z HTTP 0.9. Kolejną wersję HTTP/1.0 (RFC1945) udokumentowano w 1996r, została ona rozszerzona o zwracane nagłówki i metody. Aktualna wersja 1.1 jest już dostępna w RFC2068. HTTP/1.1 odejmuje wiele nowych funkcji. Przede wszystkim gwarantuje, że przeglądarki i serwery różnych wersji mogą poprawnie współpracować. Wprowadzono: identyfikacje nazwy hosta, stałe połączenie z serwerem, fragmentowane transfery, wsparcie dla serwerów proxy i "ciasteczek".

Celem ćwiczenia jest zapoznanie się z z funkcjonowaniem protokołu HTTP oraz podstawowymi aspektami pracy, konfiguracji i administracji serwerem Apache.

2. Schemat transakcji HTTP



Żądanie :

Po podaniu adresu URL np.: http://pk.edu.pl:80/
przeglądarka interpretując jego części, stwierdza że :

http://

należy wykorzystać protokół HTTP. HTTP najczęściej implementowane jest nad TCP/IP, lecz nie jest to wymóg - HTTP może pracować nad dowolnym protokołem warstwy transportowej, o ile jest on zorientowany strumieniowo.

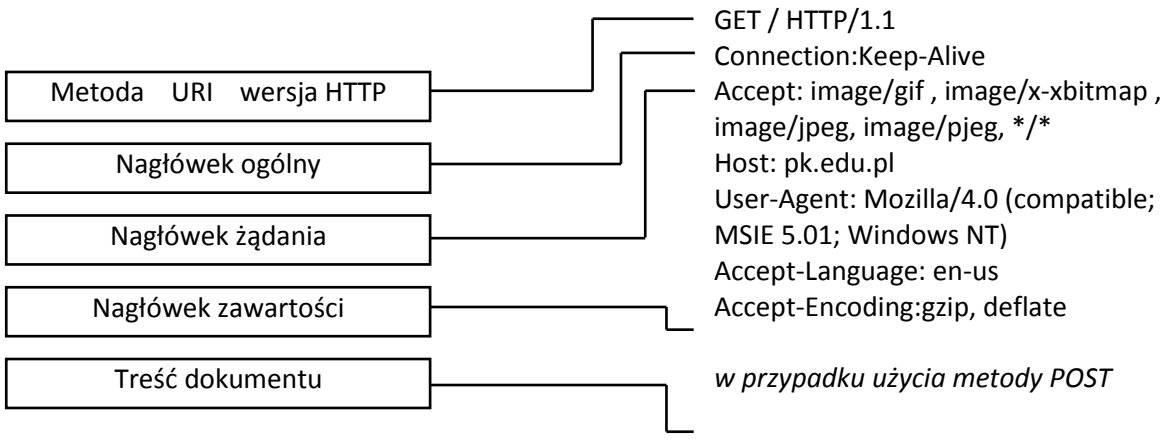
pk.edu.pl

i za jego pomocą połączyć się poprzez sieć z serwerem o nazwie domenowej *pk.edu.pl*

:80

poprzez port 80 tego serwera. Można wykorzystać dowolny port z zakresu od 1 do 65535 włącznie. W przypadku pominięcia dwukropka przeglądarka próbuje połączyć się z domyślnym portem HTTP(80).

2.1 Ogólna Konstrukcja żądania HTTP



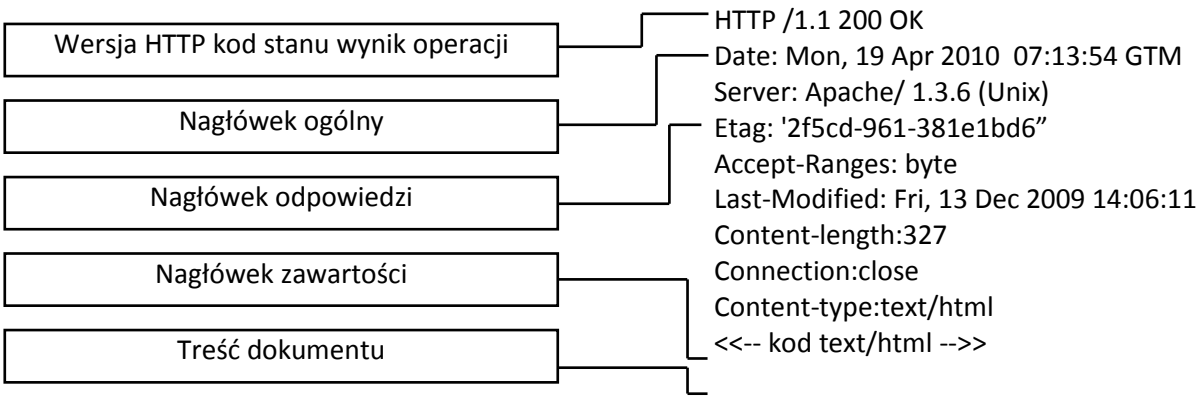
Żądania klienta (zgodnie z ilustracją).

Przeglądarka klienta łączy się z *pk.edu.pl* wysyłając do serwera komunikat :

- ***GET / HTTP/1.1*** Zawiera żądanie przesłania z serwera dokumentu położonego pod adresem / . HTTP/1.1 to wersja protokołu wykorzystywana przez przeglądarkę.
- **Drugi /trzeci wiersz** informuje serwer, jakiego rodzaju dokumenty może przyjmować przeglądarka.
- ***Accept-Language:*** **en-us** -informacja, iż preferowanym językiem klienta jest angielski.
- **Accept-Encoding:gzip, deflate** informuje serwer, że program „wie", jak interpretować treść odpowiedzi serwera skompresowaną za pomocą gzip i deflate.
- **W szóstym/siódmym wierszu** znajdują się informacje o kliencie.
- **W ósmym wierszu** klient podaje serwerowi nazwę domenową, do której chce się odwołać. (w protokole HTTP 1.0 nagłówek ten był opcjonalny).
- **Connection:Keep-Alive** nakazuje serwerowi utrzymywanie otwartego połączenia TCP, dopóki nie otrzyma jawnego żądania rozłączenia.

Wszystkie te wiersze razem tworzą żądanie. Wiersze od drugiego do siódmego to pola nagłówka żądania.

2.2. Konstrukcja odpowiedzi serwera HTTP



Odpowiedzią jest komunikat w HTML-u lub innym języku, wywodzącym sie z SGML-a . Wiersze od 2 do 9 tworzą **nagłówek odpowiedzi**, zaś następująca po nich treść jest określana mianem **body** (ang. ciało). Przyjrzyjmy się informacją zawartym w nagłówku.

- **Pierwszy wiersz** informuje oprogramowanie klienta o wersji protokołu http wykorzystywanej przez serwer. Co ważniejsze jednak, zwraca kod stanu *200 OK (poniżej zamieszczono tabelę kodów)*, serwer informuje, iż dokument został znaleziony, a jego treść zostanie przekazana w treści odpowiedzi.
- **W drugim wierszu** znajduje się informacja o czasie lokalnym serwera. Czas jest podawany zgodnie z czasem Greenwitch.
- **Trzeci wiersz** zwraca informacje o oprogramowaniu serwera.
- **Czwarty wiersz** informuje o dacie ostatniej modyfikacji dokumentu, którego żąda klient.
- **Piąty wiersz** zawiera *znacznik zawartości (ang. entity tab*). Jest to jednoznaczny dla przeglądarki identyfikator zasobu serwera.
- ***Accept-Ranges*** -oznacza, że serwer ma możliwość zwracania fragmentów dokumentu zamiast całej jego zawartośći.
- **W siódmym wierszu** znajduje się informacja o liczbie bajtów wchodzących w skład treści dokumentu wysłanej po nagłówku.
- **Connection:close** - połączenie TCP/IP jest zrywane przez serwer, kiedy już przekaże cały dokument. Klient może wcześniej zerwać połączenie - serwer nie powinien w tej sytuacji odnotowywać błędu.
- **W dziewiątym wierszu** (Content- Type) znajdują się informacje o rodzaju zwracanego przez serwer dokumentu – np. plik HTML. Po wszystkich tych informacjach następuje wiersz odstępu, a po nim treść samego dokumentu.

2.3. Kody odpowiedzi serwera

Pierwszy wiersz odpowiedzi serwera zawiera kolejno: informacje o wersji protokołu HTTP, trzycyfrowy kod stanu oraz opis wyniku przetworzenia żądania w formie zrozumiałej dla użytkownika. W protokole HTTP sprecyzowano tylko kilka kodów w każdym z zakresów, jednak w miarę rozwoju protokołu niezdefiniowane dotąd wartości będą zapewne stopniowo uzupełniane.

Zakres kodów	Znaczenie odpowiedzi
100-199	Informacja
200-299	Żądanie zostało przetworzone poprawnie
300-399	Żądanie zostało przeadresowane; niezbędne jest wykonanie dalszych kroków
400-499	Żądanie jest niekompletne (w tym słynne 404)
500-599	Wystąpił błąd w pracy serwera

3. Metody HTTP

Metoda HTTP jest poleceniem lub żądaniem wysyłanym przez klienta do serwera. Można ją obrazowo porównać do deklaracji „intencji" klienta.

Metoda	**Znaczenie**
GET	pobranie zasobu wskazanego przez URI, może mieć postać warunkową jeśli w nagłówku występują pola warunkowe takie jak "If-Modified-Since"
HEAD	pobiera informacje o zasobie - stosowane do sprawdzania dostępności zasobu. Pobiera nagłówki: *Last-Modified, Content-Length , Content-Range, Content-Type, Server*
PUT	przyjęcie danych w postaci pliku przesyłanych od klienta do serwera
POST	przyjęcie danych przesyłanych od klienta do serwera (np. wysyłanie zawartości formularzy). Żądanie powinno zawierać nagłówek *Content-type ,* określające format treści przesyłanej przez klienta. Najczęściej używanym formatem jest kodowanie danych w adresie URL.
DELETE	żądanie usunięcia zasobu, włączone oczywiście jedynie dla uprawnionych użytkowników.

OPTIONS	Klient żąda podania spisu opcji dla określonego zasobu serwera. W tej metodzie można podać adres URL pojedynczego dokumentu albo znak gwiazdki (*), oznaczający odwołanie do całego serwera. Serwer zwraca pola *Allow* (dla pojedynczego zasobu) , *Public* (dla całego serwera), np. Public: GET , HEAD , PUT
TRACE	Umożliwia ustalenie, w jaki sposób wiadomość wysłana przez klienta zostaje zmodyfikowana podczas przechodzenia poprzez łańcuch serwerów pośredniczących. Serwer zwraca nagłówki: *Max-Forwards* – maksymalna liczba serwerów leżących pomiędzy serwerem głównym. Każdy kolejny serwer zmniejsza tą wartość o 1 oraz dopisuje do nagłówka *Via* numer używanej wersji HTTP oraz nazwę hosta.
CONNECT	Żądanie przeznaczone dla serwerów pośredniczących pełniących funkcje tunelowania. Gdy klient chce nawiązać połączenie z serwerem wykorzystującym protokół HTTPS, musi zrobić to poprzez serwer pośredniczący, wysyłając temu ostatniemu polecenie CONNECT. Metoda CONNECT nie jest częścią standardu HTTP/1.1, jednak jest powszechnie implementowana na podstawie dokumentu internet-draft wygasłego w 1999 roku.

Zadanie:

W jaki sposób ,klient wykonując zapytanie do serwera oddziela poszczególne nagłówki w żądaniu ? (podpowiedź : wykonaj zapytanie używając http://web-sniffer.net/)

4. Konfiguracja serwera Apache

Serwery WWW są jednym z istotnych elementów każdej sieci. Apache jest jednym z najbardziej popularnych rozwiązań stosowanych we współczesnym Internecie.

Apache działa jako demon HTTP (`httpd`), a jego tradycyjna konfiguracja znajduje się w plikach:

httpd.conf - jest to podstawowy plik konfiguracyjny, zawierający parametry związane z protokołem HTTP i działaniem serwera),

srm.conf - plik ten zwyczajowo zawiera ustawienia określające sposób odpowiadania przez serwer na zapytania klientów)

access.conf - definiujący zazwyczaj kontrolę dostępu do serwera i udostępniane przezeń informacje).

Wszystkie one są plikami tekstowymi, zawierającymi komentarze zaczynające się od znaku # oraz dyrektywy mające przeważnie postać opcji, po której następuje wartość. Możliwe jest zebranie zawartości wszystkich powyższych plików w jednym (taki model jest obecnie preferowany).

Struktura katalogów oraz ich zawartość w serwerze Apache2 jest następująca (może się ona nieco różnić w zależności od stosowanego systemu operacyjnego i dystrybucji):

Lokalizacja	**Opis**
/etc/apache2/	folder zawiera pliki konfiguracyjne
apache2.conf	główny plik konfiguracyjny
ports.conf	ustawienia portów
Envvar	zmienne środowiskowe
/conf.d/char set	ustawienie domyślnego kodowania znaków
/conf.d/security	globalne ustawienia dostępu
/sites-available/000-default	ustawienia hostów
/etc/apache2/ modules-available i module-enabled	tutaj znajdziemy moduły, z którymi działa serwer Apache
/var/log/apache2	pliki z logami, gdzie zapisane są informacje dotyczące działania serwera Apache
/var/www/	katalog główny (tj. strona główna)

3.1. Podstawowe dyrektywy pliku httpd.conf

ServerAdmin adres_pocztowy_administratora

Określenie adresu pocztowego administratora serwera www.

ServerName nazwa_domenowa_serwera

Zdefiniowanie nazwy hosta zwracanej klientom podczas pobierania danych z serwera.

ServerRoot "katalog"

Określenie katalogu z plikami używanymi przez httpd, m.in. plikami błędów i dzienników.

ServerType typ

Możliwe wartości typu to standalone (ustawiany gdy httpd uruchamiany jest samodzielnie) lub inetd (co jest rozwiązaniem preferowanym - serwer uruchamiany jest wtedy za pośrednictwem demona inetd).

Port nr_portu

Numer portu TCP używanego przez serwer (standardowo 80; inne popularne wartości to 8080 i 8000). Jeśli serwer jest typu inetd, to wskazane jest ustawienie wartości portu większej niż 1024 - porty o niższych numerach są uprzywilejowane i inetd uruchamia httpd z identyfikatorem superużytkownika. W przypadku serwera standalone nie ma to znaczenia - usługi świadczone są nie przez początkowy proces httpd, lecz przez jego procesy potomne, działające z mniejszymi uprawnieniami.

BindAddress adres_IP

Okrela adres używany do komunikacji z serwerem, jeśli posiada on kilka adresów IP. Wartość adresu * nakazuje serwerowi odpowiadać na żądania kierowane pod dowolny z jego adresów.

Listen port

Listen adres:port

Określenie adresów i portów, które oprócz domyślnego portu i adresu mają być monitorowane pod kątem pojawienia się żądań www.

MinSpareServers liczba

MaxSpareServers liczba

Minimalna i maksymalna liczba bezczynnych procesów potomnych serwera standalone.

StartServers liczba

Liczba procesów potomnych demona httpd uruchamianych przy starcie systemu.

MaxClients liczba

Maksymalna liczba równocześnie obsługiwanych klientów.

DocumentRoot "katalog"

Katalog z główną stroną serwera.

UserDir katalog

Określenie nazwy katalogu w którym każdy z użytkowników systemu może przechowywać swoją stronę domową.

DirectoryIndex nazwa_pliku

Nazwa pliku przesyłanego klientom gdy żądanie nie zawiera nazwy pliku (np. plik o tej nazwie umieszczony w katalogu podanym w *DocumentRoot* traktowany jest jako strona główna serwera, a w katalogu podanym jako UserDir - jako strona domowa użytkownika).

< Directory "katalog"> ... < /Directory >

Określenie dyrektyw odnoszących się jedynie do danego katalogu.

< VirtualHost "nazwa_domenowa"> ... < /VirtualHost >

określenie nazwy na którą reaguje serwer (musi byc ona nazwą równoważną dla nazwy domenowej serwera, zdefiniowaną na serwerze DNS), oraz dyrektyw odnoszących się do tego serwera.

4. Realizacja ćwiczenia.

Dane podstawowe:

Logowanie do systemu przy użyciu danych: login > root haslo > sk2010

Ustawienia sieci:

Serwer ip: 10.0.2.2

Klient ip: 10.0.2.15

Lokalizacja pliku resetującego ustawienia: /etc/init.d/skrypt2

4.1. Uruchomienie i zatrzymanie pracy serwera HTTP

W tej części proszę przetestować następujące polecenia :

/etc/init.d/apache2 start

/etc/init.d/apache2 stop

/etc/init.d/apache2 restart

/etc/init.d/apache2 status

Zadanie 1:

Wydaj polecenie serwerowi ponownego załadowania konfiguracji, nie tracąc ciągłości jego działania (czy jest to możliwe?).

4.2 Konfiguracja podstawowa.

Tworzenie stron użytkowników.

/home/nazwa_uzytkownika/public_html

Katalog ze stronami użytkownika. Należy pamiętać, że aby strony użytkowników działały musimy z folderu **modules-available** podlinkować pliki **userdir.conf** i **userdir.load** do folderu **module-enabled** (poleceniem *ln –s* lub – co jest preferowane – skorzystać z polecenia *a2enmod*). Proszę wykonać *man a2enmod* by uzyskać więcej informacji.

Podstawowe dyrektywy pliku apache2.conf

ServerRoot ‘katalog’

Ścieżka do podstawowych plików konfiguracyjnych serwera. Domyślnie /etc/apache2/

AccessFileName 'nazwa_pliku'

Określa nazwę pod jaką będą rozpoznawane pliki .htaccess (określające prawa dostępu do zasobów).

<Files ~ „^\.ht">

Order allow, deny

Deny from All

</Files>

Sprawia, że pliki .htaccess i htpasswd są niewidoczne dla przeglądających zawartość serwera.

ErrorLog 'sciezka_do_pliku'

CustomLog 'sciezka_do_pliku'

Określa miejsce, w którym będą zapisywane dzienniki błędów i zdarzeń.

Include /etc/apache2/mods-enabled/.load*

Include /etc/apache2/mods-enabled/*.conf

Pozwala na ładowanie dodatkowych modułów i ich konfiguracji.

Include /etc/apache2/conf.d/update-rc.d skrypt2 defaults

Katalog ustawień kodowania znaków i zabezpieczeń.

Hosty wirtualne

Include /etc/apache2/sites-enabled/

Katalog zawierający plik 000-default a w nim ustawienia wirtualnych hostów, ścieżek dla stron głównych, skryptów itp.

Plik zaczyna się od od określenia jakiego portu lub hosta dotyczy dana konfiguracja poprzez umieszczenie jej w tagach:

<VirtualHost nazwa_hosta:nr_portu>

</VirtualHost>

Na samą konfigurację składają się następujące dyrektywy:

Server Admin mail

Email do administratora.

Document Root sciezka_do_folderu

Określa katalog główny dla danego hosta. Po tej dyrektywie ustawia się opcje dla katalogu w tagach:

<Directory />

</Directory>

Następnie w taki sam sposób konfiguruje się ustawienia katalogu zawierającego stronę główną:

<Directory sciezka_do_katalogu>

</Directory>

Podstawowe opcje katalogu głównego:

Options FollowSymLinks

AllowOverride None

Dla katalogu stron:

Options Indexes FollowSymLinks MultiViews

Porty

Konfriguracja portów odbywa się w pliku:

/etc/apache2/ports.conf

Definiujemy wirtualny host i port, na którym serwer ma nasłuchiwać.

W pliku tym każdy kolejny port określa się następującymi liniami:

*NameVirtualHost *:nr_portu*

Listen nr_portu

Zadanie 2:

Proszę skonfigurować serwer tak, aby :

- katalog strony głównej znajdował się w */var/www/imie_studenta/ (proszę umieścić w tym katalogu przykładową stronę)*

- dostęp do strony ma być wyłącznie na porcie 8080
- dziennik błędów był zapisany w pliku /var/www/errors

i przetestować konfigurację wpisując w okno przeglądarki adres serwera.

Ograniczenia dostępu do indywidualnych katalogów i dla określonych adresów IP

a) ograniczanie dostępu do indywidualnych katalogów.

Może odbyć się dwoma sposobami :

1. Dodając do pliku : ***/etc/apache2/sites-enabled/000-default***

<Directory nasz_katalog>

Deny from all

</Directory>

2. Dodając plik .htaccess w folderze, który chcemy ukryć, wpisując do niego:

Deny from all

b) Ograniczenie dostępu dla określonego IP

Można dokonać poprzez modyfikację pliku :

/etc/apache2/sites-enabled/000-default

dodając do niego:

<Directory nasz_katalog>

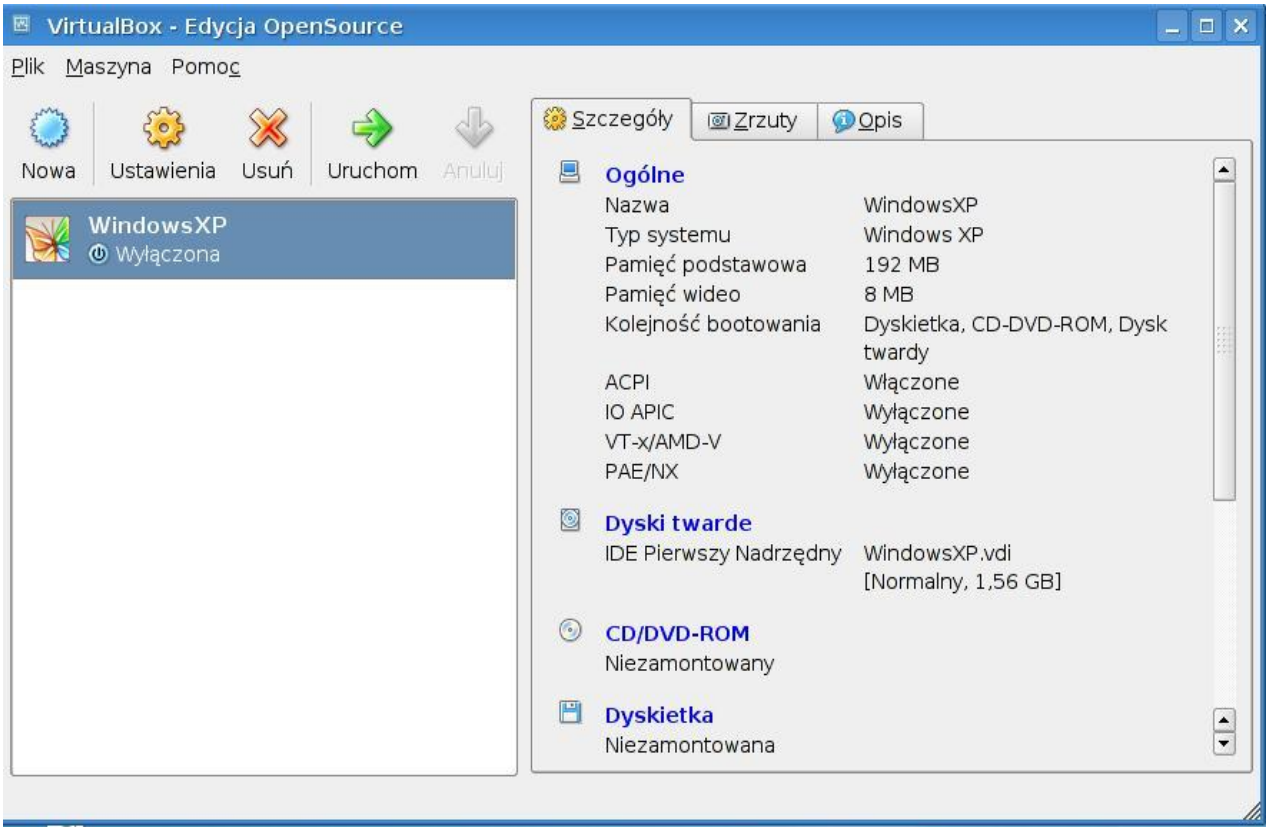
Deny from IP_adres

</Directory>

nasz_katalog – oznacza miejsce, do którego nie będzie miał dostępu IP_adres.

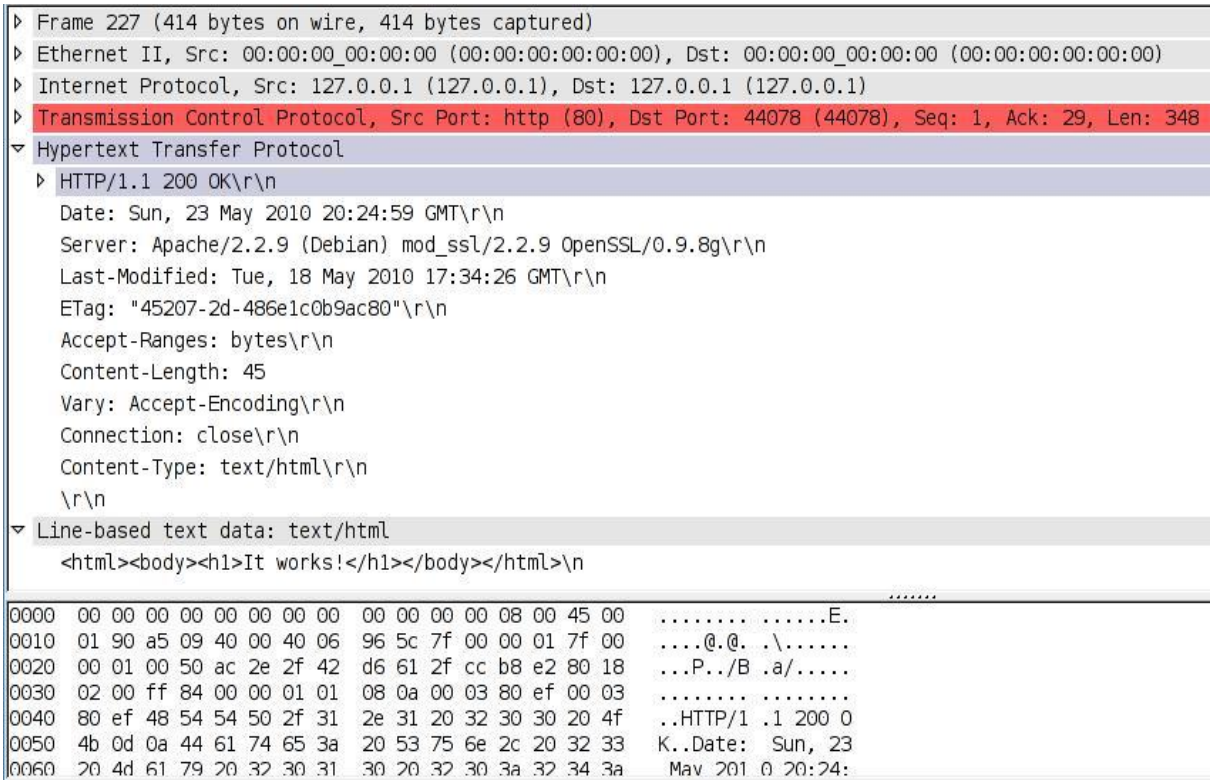
4.3. *Praca protokołu HTTP.*

Proszę uruchomić VirtualBox'a , a następnie WindowsXP.

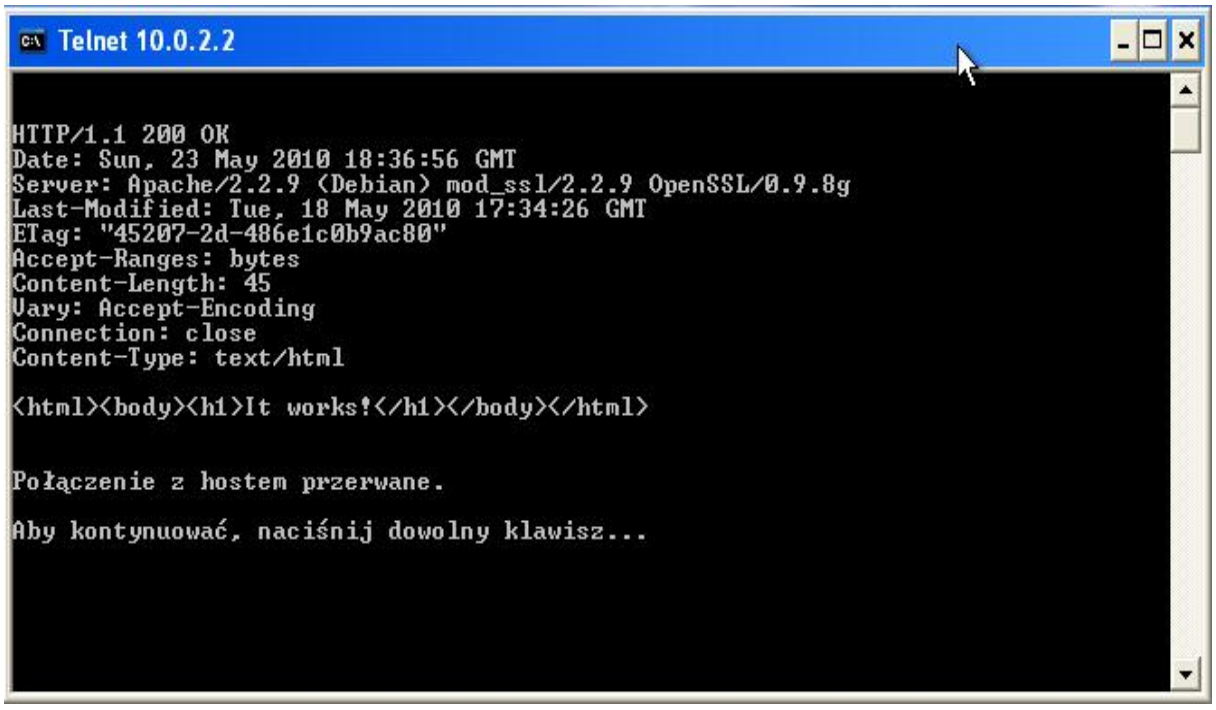


Pamiętaj! Wyjście z okienka Windows do Debiana po naciśnięciu ***prawy Ctrl***

- Sprawdź poprzez przeglądarkę dostępność serwera wpisując : http://10.0.0.2/ jeśli serwer odpowiada uruchom konsolę.
- W systemie Debian uruchom aplikacje:*Wireshark* (*k->internet->wireshark*)
- a następnie *Capture->Options...* i w *interface* wybrać *eth0 (lub analogiczny interfejs Ethernetowy),* potem *Start*.
- Przechodzimy z powrotem do systemu Windows,w konsoli poprzez **telnet** łączymy się z serwerem- poleceniem : telnet 10.0.2.2 80 (patrz: Dodatek)
- Wysyłamy zapytanie do serwera : ***GET /index.html /HTTP1.1 (dwa entery na końcu)***
- Przechodzimy do Debiana i obserwujemy ruch w sieci w obrębie protokołu HTTP. Serwer wysyła odpowiedź :



W konsoli Windows ostajemy w odpowiedzi pełen zestaw nagłówków, a przede wszystkim zawartość dokumentu *index.html*



4.4. *Szyfrowane połączenie HTTP*

Jak Państwo zdołaliście zauważyć HTTP przesyła dane otwartym tekstem. Spróbujmy transmisje zaszyfrować. Nasze połączenie będzie zabezpieczone typowym szyfrowaniem RSA.

Musimy na wstępie wygenerować klucz i certyfikat dla naszego lokalnego serwera, zazwyczaj określa się go z ważnością na 1 rok. Używamy w tym celu komendy :

openssl genrsa -out /etc/apache2/apache.key 1024

openssl req -new -x509 -days 365 -key /etc/apache2/apache.key –out /etc/apache2/apache.crt

I sprawdzamy czy w pliku: */etc/apache2/ports.conf* jest włączone nasłuchiwanie na porcie 443 (Listen 443).

Uruchamiamy moduł SSL komendą :

a2enmod ssl

Modyfikujemy plik /etc/apache2/sites-available/000-default dodając:

*NameVirtualHost *:443*

ServerAdmin webmaster@localhost

SSLEngine On

SSLCertificateFile /etc/apache2/apache.crt

SSLCertificateKeyFile /etc/apache2/apache.key

DocumentRoot /var/www/

Na koniec restartujemy serwer. Możemy teraz przetestować połączenie szyfrowane : *https://10.0.0.2.* Proszę zbadać treść przesyłanych komunikatów w programie Wireshark.

4.5. Obsługa skryptów CGI

CGI jest to znormalizowany interfejs, umożliwiający komunikację pomiędzy oprogramowaniem WWW a innymi programami znajdującymi się na serwerze. Standardowo obsługa plików CGI jest w apache2 wyłączona. W celu jej uruchomienia musimy zmodyfikować plik :

/etc/apache2/sites-available/default

dodając do niego :

<Directory "/var/www">

AddHandler cgi-script .cgi

Options +ExecCGI

</Directory>

pierwsza linia określa ścieżkę do katalogu, w którym będziemy mieli prawo do wykonywania plików CGI. Druga określa rozszerzenie plików CGI jako .cgi.

Najważniejszą czynnością realizowaną w tej części jest nadanie prawa do wykonywania plików CGI przez Apache'a poprzez *Options +ExecCGI .*

Utwórzmy teraz w folderze /var/www/ plik test1.cgi o zawartości :

#!/usr/bin/perl

print "Content-type: text/html\n\n";

print "Hello, World.";

Jednak zanim przystąpimy to testowania naszego skryptu, nie zapomnijmy o nadaniu prawa do uruchamiania test1.cgi na poziomie systemu operacyjnego. W tym celu wykonajmy :

chmod a+x /var/www/test1.cgi

Teraz możemy zaobserwować niezbyt zaskakujące działanie skryptu, wywołując go w przeglądarce. Proszę przeanalizować także w jaki sposób http realizuje przesyłanie i wykonywanie skryptu (korzystając z Wireshark)

4.6. Uwierzytelnianie

Plik .htaccess umieszczony w danym katalogu może zawierać opcje allow, deny zezwalające/zabraniające na dostęp do katalogu(tego, w którym się znajduje). Może też w połączeniu z plikiem .htpasswd określać użytkowników mających dostęp do katalogu po podaniu odpowiedniego hasła.

Przykładowy plik *.htaccess:*

AuthName „Podaj haslo”

AuthType Basic

AuthUserFile sciezka_do_.htpasswd

Require valid-user

Plik *.htpasswd* przechowuje nazwy użytkowników i odpowiadające im odpowiednio zakodowane (chyba, że świadomie to kodowanie wyłączymy) hasła w formie:

Nazwa_uzytkownika:zakodowane_haslo

Plik taki można wygenerować z konsoli za pomocą polecenia:

htpasswd -c /sciezka_do_pliku/nazwa_pliku nazwa_uzytkownika

Następnie wystarczy podać hasło, które automatycznie zostanie zapisane w pliku w swojej zakodowanej wersji. Proszę zbadać tą funkcjonalność – podglądając także proces uwierzytelniania w Wireshark.

4.7. Ciasteczka (cookies)

Cookies - to mała porcja informacji wysyłana przez serwer do przeglądarki użytkownika, przechowywana lokalnie na jego komputerze. Przyczyną, dla której niezbędne okazało się wprowadzenie nowej technologii do istniejącego środowiska WWW, była bez wątpienia chęć identyfikacji anonimowych do tej pory użytkowników Internetu. Sam protokół http nie przewidywał ani nawiązywania "sesji" pomiędzy klientem a serwerem, ani też rejestracji odwiedzin użytkownika.

Naszym interfejsem przy obsłudze ciasteczek będzie język PHP.

Utwórz w katalogu /var/www/ plik o nazwie : setcookies.php i zawartości :

```
<?php
$value='sieci_komputerowe';
setcookie("testCookie", $value);
?>
```

Spod systemu Windows (za pomocą telnetu) wykonaj zapytanie do serwera :

```
GET /setcookies.php /HTTP1.1
```

Proszę sprawdzić w Wireshark'u czy serwer ustawił „ciasteczka” . (Nagłówek set-cookie).

Do odczytywania naszych Cookies użyjemy skryptu :

```
<?php $_COOKIE["TestCookie"];
echo $HTTP_COOKIE_VARS["TestCookie"];
cookiesprint_r($_COOKIE);
?>
```

Proszę sprawdzić, jakie nagłówki związane z Cookies są przekazywane do serwera po wykonaniu skryptu na maszynie Windows.

4.8. Formularze *(dla ambitnych)*

Dzięki formularzom możemy w bardzo prosty sposób zaobserwować działanie metody POST. W tym celu proszę do pliku strony głównej *index.htm* dodać prosty formularz zawierający jedno pole *select* z dwoma opcjami oraz pola *input* typu *text i submit.* Ustawić *method=POST* i *action=1.php*. Stworzyć w tym samym miejscu plik *1.php* zawierający kod:

```
<?php echo „wysłano”; ?>
```

Zaobserwować w programie Wireshark co dzieje się po naciśnięciu przycisku *submit* zarówno po stronie klienta jak i serwera.

4.9. Zadanie dla bardzo ambitnych *(sprawdź swoją wiedzę – także z DNS)*

Należy skonfigurować, serwer Apache'a w taki sposób, aby był gotowy do uruchomienia dwóch oddzielnych domen na jednej maszynie:

1. **pk.edu.pl**

Domena będzie w zawierać dwie strony użytkowników

pk.edu.pl/~nowak/

pk.edu.pl/~kowalski/

2. **admin.pk.edu.pl**

Do domeny admin.pk.edu.pl będzie miała dostęp wyłącznie osoba z komputera o adresie IP : 10.0.2.2 , z uwierzytelnieniem na:

login : admin

haslo : http

Połączenie ma być szyfrowane (klucz oraz certyfikat ma być umieszczony w folderze /etc/apache2/), proszę także umożliwić wykonywanie skryptów CGI.

5. Wnioski i zakończenie

W sprawozdaniu powinien znaleźć się opis realizacji zadań, wraz z uzyskanymi wynikami oraz wnioskami. Proszę też wymyślić jedno zadanie związane z tematem ćwiczenia i podać jego rozwiązanie oraz podać słabe strony/błędy w instrukcji (co pozwoli dopracować ją na przyszłość). Sprawozdanie może stanowić również rozwiązanie zadania z części **4.9** wraz z opisem jego wykonania i potwierdzeniem funkcjonowania.

Dodatek A – program Telnet

Wśród rozlicznych usług, które mogą być realizowane w sieciach opartych na protokole TCP/IP, jedną z najbardziej elementarnych i podstawowych jest telnet. Technicznie jest to najprostsza z usług sieciowych: służy do nawiązania interaktywnego połączenia terminalowego ze wskazanym komputerem w sieci (a dokładniej z programem - serwerem telnetu, pracującym na tym komputerze) – na wybranym przez użytkownika porcie. Po nawiązaniu takiego połączenia znaki wpisywane na klawiaturze naszego komputera przesyłane są poprzez sieć do maszyny, z którą jesteśmy połączeni, a przesyłane w odwrotną stronę odpowiedzi wyświetlane są na ekranie.

W Windows : uruchom konsolę i wpisz *telnet*

o ip port – otwiera połączenie z komputerem o danym IP na danym porcie

Po tej komendzie możemy wydawać polecenia. Nas będą szczególnie interesowały zapytania do serwera typu: *GET /index.html /HTTP1.1*

Literatura dodatkowa:

1. Clinton Wong - *HTTP Leksykon kieszonkowy*
2. Rich Bowen - *Apache : podręcznik administratora*
3. *http://www.apache.org/*

Instrukcja opracowana przez:

Michał Budziło
Bartosz Ryndak

Opieka merytoryczna:

dr inż. Piotr Andrzej Kowalski, dr inż. Szymon Łukasik, mgr inż. Sławomir Żak