Universitatea Tehnică a Moldovei
Facultatea Calculatoare Informatică și Microelectronică
Catedra Tehnologia Informației

Laboratorul 3.
Programarea în rețea

A efectuat: studenta gr. FI-171 Ciocanu Carolina
A examinat: Buldumac Oleg

Chisinau, 2020

Condiții:

• Pentru acest laborator utilizați librării HTTP deja existente, nu este necesar de a utiliza Sockets API. Cine dorește poate să facă prin socket ca și la primul laborator.

• Clientul trebuie să facă cereri GET, POST, HEAD și OPTIONS

• Aplicația poate fi GUI sau consolă

• Nu sunteți limitați la funcțional, resura(pagina web) la care clientul o să facă cereri HTTP este la alegere.

• Vă recomand să folosiți proxy private și nu free: https://proxy-seller.com

• Aplicația elaborată trebuie să aibă o logică bine definită

Întrebări:

Cum este formatat corpul unei cereri HTTP pentru o cerere HTTP de tip POST ?

• De unde știe un client HTTP ce tip de conținut trimite serverul HTTP ?

• Cum decide un client dacă ar trebui să aibă încredere în certificatul unui server ?

• Care este problema principală cu certificatele autosemnate ?

• Conexiunea persistentă HTTP – care sunt principalele beneficii ?

• Ce este negocierea conținutului în HTTP și cum are loc ?

• Care sunt tipurile de negociere a conținutului HTTP ?

• Ce este un ETag în HTTP și cum funcționează ?

• Diferența dintre protocoalele fără stare și cele cu stare. Cărui tip îi aparține HTTP ?

• Avantajele cheie ale HTTP/2 în comparație cu HTTP/1.1

• Ce este un tip MIME, din ce constă și pentru ce se folosește ?

• Care este diferența dintre GET și POST ?

• Care este diferența dintre PUT și POST ?

• Care sunt metodele idempotente în HTTP și care sunt scopul lor.

• Cum sunt identificate resursele în protocolul HTTP ?

• Care sunt metodele sigure și nesigure în HTTP ?

• Pentru ce este nevoie de cURL ?

• Pentru ce este nevoie de HTTP Proxy?

• Diferența dintre autentificare și autorizare

• Care sunt metodele de autentificare HTTP ?

• Modalități de identificare a utilizatorilor în HTTP

• HTTP cookies – pentru ce se folosește ?

Răspunsuri:

Cum este formatat corpul unei cereri HTTP pentru o cerere HTTP de tip POST ?

În cazul unei cereri POST variabilele nu se află în URI, ci în partea body:

POST /wiki/special:Search HTTP/1.1

Host: ro.wikipedia.org

Content-Type: application/x-www-form-urlencoded

Content-Length: 24

search=pisici&go=articol

Serverul răspunde astfel :

HTTP/1.0 302 Moved Temporarily Date: Fri, 13 Jan 2008 15:32:43 GMT Location: http://ro.wikipedia.org/wiki/pisici

• De unde știe un client HTTP ce tip de conținut trimite serverul HTTP ?

Un client HTTP trimite un mesaj de solicitare către un server HTTP. La rândul său, serverul returnează un mesaj de răspuns. Cu alte cuvinte, HTTP este un protocol pull, clientul trage informații de pe server (în loc de server împinge informații în jos către client). Clientul HTTP și serverul comunică trimitând mesaje text. Clientul trimite un mesaj de solicitare către server. La rândul său, serverul returnează un mesaj de răspuns.

• Cum decide un client dacă ar trebui să aibă încredere în certificatul unui server ?

Un certificat digital este o acreditare electronică pe care o puteţi folosi pentru a vă demonstra identitatea pentru o tranzacţie electronică. Există un număr din ce în ce mai mare de modalităţi de folosire a certificatelor digitale, pentru a se asigura măsuri îmbunătăţite de securitate în reţea.

De exemplu, certificatele digitale sunt esenţiale pentru configurarea şi utilizarea SSL. Folosirea SSL vă permite să creaţi conexiuni sigure între utilizatori şi aplicaţii server peste o reţea ce nu este de încredere, cum ar fi Internet. SSL oferă una dintre cele mai bune soluţii pentru protecţia în Internet a caracterului privat al datelor sensibile, cum ar fi numele de utilizator şi parolele. Multe platforme System i şi aplicaţii, ca FTP, Telnet, server HTTP furnizează suport SSL pentru a asigura confidenţialitatea datelor.

Un utilizator care încearcă să se conecteze la un site web securizat utilizând Windows Internet Explorer poate primi următorul mesaj de avertizare:

Există o problemă cu certificatul de securitate al acestui site web. Certificatul de securitate prezentat de acest site web nu a fost emis de o autoritate de certificare de încredere. Probleme de certificat de securitate poate indica o încercare de a vă păcăli sau intercepta datele pe care le trimiteți la server. Vă recomandăm să închideți această pagină Web și să nu continuați cu acest site web.

• Care este problema principală cu certificatele autosemnate ?

În multe organizații, utilizarea certificatelor autofirmate este interzisă de politică. Organizațiile pot interzice utilizarea certificatelor autofirmate din mai multe motive: Este banal ușor să generezi perechea de chei a unui certificat fără o entropie rezonabilă, să nu protejezi în mod corespunzător cheia privată a perechii de chei pentru utilizarea sa, să validezi slab certificatul atunci când este utilizat și să folosești greșit un certificat auto-semnat atunci când autoritatea de certificare ar fi trebuit să fie utilizată în schimb. Cu toate acestea, atunci când este utilizat în mod corespunzător și adecvat, un certificat semnat de la sine asigură o securitate acceptabilă în anumite situații.

• Conexiunea persistentă HTTP – care sunt principalele beneficii ?

Conexiunile HTTP persistente prezintă o serie de avantaje: - Prin deschiderea și închiderea mai puține conexiuni TCP, timpul procesorului este economisit în routere și gazde (clienți, servere, proxy, gateway-uri, tunele sau memorii cache), iar memoria folosită pentru blocurile de control TCP poate fi salvat în gazde

Cererile și răspunsurile HTTP pot fi canalizate pe o conexiune. Pipelining permite unui client să facă mai multe cereri fără așteptând fiecare răspuns, permițând o conexiune TCP unică să fie utilizat mult mai eficient, cu un timp scurs mult mai mic.

Congestionarea rețelei este redusă prin reducerea numărului de pachete cauzate de TCP deschise și prin permiterea TCP timp suficient pentru a determina starea de congestionare a rețelei.

Întârzierea la solicitările ulterioare este redusă, deoarece nu există timp petrecut în strângerea de mână a conexiunii TCP.

HTTP poate evolua mai grațios, deoarece erorile pot fi raportate fără penalizarea închiderii conexiunii TCP. Clienții care folosesc versiuni viitoare de HTTP ar putea încerca în mod optim o

nouă caracteristică, dar dacă comunică cu un server mai vechi, încearcă să încerce cu semantică veche după ce este raportată o eroare.

• Ce este negocierea conținutului în HTTP și cum are loc ?

Negocierea de conținut se referă la mecanisme definite ca parte a HTTP care fac posibilă difuzarea diferitelor versiuni ale unui document (sau mai general, reprezentări ale unei resurse) la același URI, astfel încât agenții utilizatori pot specifica ce versiune se potrivește cel mai bine capabilităților lor. . O utilizare clasică a acestui mecanism este de a servi o imagine în format GIF sau PNG, astfel încât un browser care nu poate afișa imagini PNG (de exemplu, MS Internet Explorer 4) va fi difuzat versiunea GIF.

O resursă poate fi disponibilă în mai multe reprezentări diferite; de exemplu, acesta poate fi disponibil în diferite limbi sau tipuri de media diferite. Un mod de a selecta cea mai potrivită alegere este de a oferi utilizatorului o pagină de index și de a le permite să selecteze cea mai potrivită alegere; cu toate acestea, este adesea posibil să automatizați alegerea pe baza unor criterii de selecție.

• Care sunt tipurile de negociere a conținutului HTTP ?

Tipurile de negociere a conținutului HTTP:

Server-driven Negotiation - Dacă selecția celei mai bune reprezentări pentru un răspuns se face printr-un algoritm situat pe server, se numește negociere bazată pe server. Selecția se bazează pe reprezentările disponibile ale răspunsului (dimensiunile peste care acesta poate varia; de exemplu, limbă, codare conținut etc.) și conținutul anumitor câmpuri de antet din mesajul de solicitare sau pe alte informații referitoare la solicitare (cum ar fi ca adresă de rețea a clientului).

Agent-driven Negotiation - Cu o negociere bazată pe agent, selecția celei mai bune reprezentări pentru un răspuns este realizată de agentul utilizator după primirea unui răspuns inițial de la serverul de origine. Selecția se bazează pe o listă a reprezentărilor disponibile ale răspunsului incluse în câmpurile antet sau entitatea-corp a răspunsului inițial, cu fiecare reprezentare identificată de propriul său URI. Selecția dintre reprezentări poate fi efectuată automat (dacă agentul utilizator este capabil să facă acest lucru) sau manual prin selectarea utilizatorului dintr-un meniu generat (eventual hipertext).

Transparent Negotiation - Negocierea transparentă este o combinație atât de negociere condusă de server, cât și de agent. Când o memorie cache este furnizată de o formă a listei de reprezentări disponibile ale răspunsului (ca în cazul negocierii axate de agent) și dimensiunile variantei sunt înțelese complet de cache, atunci cache-ul devine capabil să efectueze o negociere bazată pe server. a serverului de origine pentru solicitări ulterioare pe acea resursă.

• Ce este un ETag în HTTP și cum funcționează ?

etagul sau eticheta entității face parte din HTTP, protocolul pentru World Wide Web. Este unul dintre mai multe mecanisme pe care le oferă HTTP pentru validarea cache-ului Web, care permite unui client să facă cereri condiționate. Acest lucru permite caşelor să fie mai eficiente și

economisește lățimea de bandă, deoarece un server Web nu trebuie să trimită un răspuns complet dacă conținutul nu s-a schimbat. ETag-urile pot fi, de asemenea, utilizate pentru controlul optim al concurentei, ca o modalitate de a ajuta la prevenirea actualizărilor simultane ale unei resurse de a se suprascrie reciproc.

Un ETag este un identificator opac atribuit de un server Web unei versiuni specifice a unei resurse găsite la o adresă URL. Dacă reprezentarea resurselor de la adresa URL se schimbă vreodată, se atribuie un ETag nou și diferit. Folosite în acest mod, ETag-urile sunt similare cu amprentele și pot fi comparate rapid pentru a determina dacă două reprezentări ale unei resurse sunt aceleași.

• Diferența dintre protocoalele fără stare și cele cu stare. Cărui tip îi aparține HTTP ?

Protocoalele fără stare

- nu necesită ca serverul să rețină informațiile despre server sau detaliile sesiunii.
- nu există o dependență strânsă între server și client.
- simplifică designul serverului.
- funcționează mai bine la momentul prăbușirii, deoarece nu există nicio stare care trebuie să fie restaurată, un server eșuat poate pur și simplu reporni după un crash.
- gestionați tranzacția foarte rapid.
- sunt ușor de implementat în Internet.

Protocoalele cu stare

- necesită serverul să salveze informațiile despre starea și sesiunea.
- există o dependență strânsă între server și client
- face ca designul serverului să fie foarte complex și greu.
- nu funcționează mai bine la momentul prăbușirii, deoarece serverul statistic trebuie să păstreze informațiile despre starea și detaliile sesiunii statelor interne.
- gestionează tranzacția foarte lent.
- sunt logic greu de implementat în Internet.

HTTP (Hypertext Transfer Protocol) face parte din protocoalele fără stare.

• Avantajele cheie ale HTTP/2 în comparație cu HTTP/1.1

Este întotdeauna de preferat să configurați serverele web cu HTTP 2.

Protocolul HTTP 2 a fost conceput pentru a rezolva multe probleme pe care protocolul HTTP 1.1 le are cu site-urile web moderne.

Când a fost introdus HTTP 1.1 în 1997, paginile web erau gestionabile doar cu marcaje și imagini HTML. Însă, paginile web din aceste zile conțin mai multe resurse cu imagini, fonturi, scripturi și multe altele. HTTP 1.1 nu a fost conceput cu siguranță pentru a gestiona aceste încărcături.

HTTP 2.0 poate comprima anteturile HTTP. Acesta permite apăsarea serverului, care poate trimite resurse browserului chiar înainte ca documentul HTML să fie complet analizat. De asemenea, permite multiplexarea, unde clientul și serverul pot procesa mai multe solicitări prin aceeași conexiune, prevenind astfel conexiuni HTTP multiple.

• Ce este un tip MIME, din ce constă și pentru ce se folosește ?

Un tip MIME este o etichetă folosită pentru identificarea unui tip de date. Este utilizat astfel încât software-ul să știe să gestioneze datele. Acesta servește același scop pe Internet, pe care îl fac extensiile de fișiere pe Microsoft Windows.

Cel mai frecvent le veți găsi în anteturile mesajelor HTTP (pentru a descrie conținutul la care răspunde un server HTTP sau formatarea datelor care sunt POSTATE într-o solicitare) și în anteturile de e-mail (pentru a descrie formatul mesajului și atașamente)

• Care este diferența dintre GET și POST ?

Metoda GET

GET este utilizat pentru a solicita date dintr-o resursă specificată.

GET este una dintre cele mai comune metode HTTP.

Metoda POST

POST este utilizat pentru a trimite date către un server pentru a crea / actualiza o resursă.

• Care este diferența dintre PUT și POST ?

Metoda PUT înlocuiește complet orice există în prezent la adresa URL țintă cu altceva. Cu această metodă, puteți crea o resursă nouă sau suprascrie una existentă, dat fiind că știți exact URI-ul de solicitare.

Metoda HTTP POST este utilizată pentru a trimite date generate de utilizator pe serverul web. De exemplu, o metodă POST se utilizează atunci când un utilizator comentează pe un forum sau dacă încarcă o poză de profil. De asemenea, trebuie utilizată o metodă POST dacă nu cunoașteți adresa URL specifică a locației în care ar trebui să se afle resursa dvs. recent creată.

• Care sunt metodele idempotente în HTTP și care sunt scopul lor.

O metodă de solicitare este considerată idempotentă dacă efectul prevăzut pe server de mai multe cereri identice cu acea metodă este același cu efectul pentru o singură cerere. Și merită menționat că idempotența este despre efectul produs asupra stării resursei pe server și nu despre codul de stare de răspuns primit de client.

Pentru a ilustra acest lucru, luați în considerare metoda DELETE, care este definită drept idempotentă. Acum, luați în considerare un client efectuează o solicitare DELETE pentru a șterge o resursă din server. Serverul procesează cererea, resursa este ștersă și serverul returnează

204. Apoi clientul repetă aceeași solicitare DELETE și, întrucât resursa a fost deja ștersă, serverul returnează 404.

În ciuda codului de stare diferit primit de client, efectul produs de o singură solicitare DELETE este același efect al mai multor solicitări DELETE către același URI.

În cele din urmă, cererile cu metode idempotente pot fi repetate automat dacă apare o defecțiune de comunicare înainte ca clientul să poată citi răspunsul serverului. Clientul știe că repetarea cererii va avea același efect prevăzut, chiar dacă cererea inițială a reușit, deși răspunsul ar putea fi diferit.

• Cum sunt identificate resursele în protocolul HTTP ?

Ținta unei solicitări HTTP se numește „resursă", a cărei natură nu este definită în continuare; poate fi un document, o fotografie sau orice altceva. Fiecare resursă este identificată de către un URI (Uniform Resource Identifier) utilizat în HTTP pentru identificarea resurselor.

Identitatea și locația resurselor de pe Web sunt date în mare parte de o singură adresă URL (Uniform Resource Locator, un fel de URI). Există, uneori, motive pentru care identitatea și locația nu sunt date de același URI: HTTP folosește un antet HTTP specific, Alt-Svc atunci când resursa solicitată dorește ca clientul să o acceseze în altă locație.

Care sunt metodele sigure și nesigure în HTTP ?

Metoda HTTP este sigură dacă nu schimbă starea serverului. Cu alte cuvinte, o metodă sigură efectuează operațiuni numai în citire. Mai multe dintre următoarele metode HTTP sunt sigure: GET, HEAD sau OPTIONS. Toate metodele sigure sunt, de asemenea, idempotente, ca unele altele, dar nesigure, cum ar fi PUT sau DELETE.

• Pentru ce este nevoie de cURL ?

curl este un instrument pentru a transfera date de la sau către un server, folosind unul dintre protocoalele acceptate (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, MQTT, POP3, POP3S, RTMP, RTMPS, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET și TFTP). Comanda este proiectată să funcționeze fără interacțiunea utilizatorului.

curl oferă o încărcătură bus de trucuri utile cum ar fi suport proxy, autentificare utilizator, încărcare FTP, postare HTTP, conexiuni SSL, cookie-uri, reluare transfer fișiere, Metalink și multe altele. După cum veți vedea mai jos, numărul de caracteristici vă va face capul să se rotească!

curl este alimentat de libcurl pentru toate caracteristicile legate de transfer

• Pentru ce este nevoie de HTTP Proxy?

Proxy HTTP servește practic ca o poartă de acces între utilizator și internet. Este folosit pentru a memora în cache fișiere și pagini web, ceea ce vă permite să le accesați ușor. Proxy HTTP are

două roluri pe care le joacă unul dintre ele, ca Client HTTP, iar celălalt ca server HTTP, care este pentru management, securitate și memorie în cache.

De asemenea, proxy-ul HTTP este folosit de numeroase browsere pentru a memora în cache site-urile dvs. favorite (site-uri web pe care le vizitați mai des), astfel încât să le puteți accesa fără a fi neapărat nevoie să reîmprospătați sau să descărcați din nou pagina. Știți când introduceți adresa URL pentru un site web pe care doriți să îl vizitați, apoi apare automat (dacă adresa site-ului este în cache) fără a aștepta să se descarce complet, adică serverul proxy HTTP la lucru.

• Diferența dintre autentificare și autorizare

Autentificare

Autentificarea se referă la validarea autentificărilor dvs. precum numele de utilizator / ID de utilizator și parola pentru a vă verifica identitatea. Sistemul stabilește dacă sunteți ceea ce spuneți că utilizați datele de acreditare. În rețelele publice și private, sistemul autentifică identitatea utilizatorului prin parolele de conectare. Autentificarea se face de obicei cu un nume de utilizator și o parolă și uneori în combinație cu factori de autentificare, care se referă la diferitele moduri de a fi autentificate.

Autorizare

Pe de altă parte, autorizarea are loc după ce identitatea dvs. este autentificată cu succes de sistem, ceea ce vă oferă în cele din urmă permisiunea completă pentru a accesa resurse precum informații, fișiere, baze de date, fonduri, locații, aproape orice. În termeni simpli, autorizarea determină capacitatea dvs. de a accesa sistemul și în ce măsură. După ce identitatea dvs. este verificată de către sistem după autentificarea cu succes, sunteți autorizat să accesați resursele sistemului.

• Care sunt metodele de autentificare HTTP ?

De bază

Numele de utilizator și parola sunt trimise ca un text codat bazat necriptat 64. Ar trebui să utilizați întotdeauna HTTPS, deoarece parola nu este criptată și poate fi ușor capturată și reutilizată dacă utilizați HTTP și nu HTTPS.

Digera

Acreditările sunt transmise serverului sub formă de hashed. Deși acreditările nu pot fi capturate prin HTTP, solicitarea poate fi redată folosind datele de autentificare.

NTML

Această metodă folosește un mecanism de provocare / răspuns sigur, care nu permite captarea parolei sau redarea atacurilor dacă utilizați HTTP. Funcționează numai cu conexiuni persistente HTTP / 1.1. Nu îl puteți folosi întotdeauna cu toate proxy-urile HTTP. De asemenea, nu ar trebui să utilizați această metodă dacă conexiunile sunt închise regulat de serverul dvs. web.

• Modalități de identificare a utilizatorilor în HTTP

Există mai multe moduri prin care un server Web vă poate identifica:

Anteturi de solicitare HTTP.

Adresa IP.

Adrese URL lungi.

Cookie-urile.

Informații de autentificare (autentificare)

• HTTP cookies – pentru ce se folosește ?

Cookie-urile pot fi utilizate pentru a aminti informații despre utilizator pentru a afișa în timp conținut relevant pentru utilizator. De exemplu, un server web ar putea trimite un cookie care conține numele de utilizator care a fost folosit ultima dată pentru a se conecta la un site web, astfel încât acesta să poată fi completat automat la următoarea dată când utilizatorul se conectează.