Servere web

Utilizarea Sistemelor de Operare

Paul Irofti

Universitatea din București
Facultatea de Matematică și Informatică
Department de Informatică
Email: paul.irofti@fmi.unibuc.ro

Modul de funcționare

Serverul Web

- comunică prin protocolul HyperText Transfer Protocol (HTTP)
- reprezintă fundația World Wide Web (WWW)
- servește pagini web, documente și alte resurse clienților
- paginile web: scrise cu HyperText Markup Language (HTML)
- navigare prin intermediul hyperlink-urilor din paginile web
- un hyperlink poate indica către o pagină a serverului web actual sau al altuia
- adresele urmează formatul Uniform Resource Locator (URL)
- `http://www.example.com/index.html`
  - `http` – protocolul
  - `www` – subdomeniu
  - `example.com` – domeniu, host, server
  - `.com` – top level domain (TLD)
  - `index.html` – fișierul ce conține pagina web

HTTP

- funcționează pe modelul cerere-răspuns
- exemplu: clientul cere o pagină web, iar server-ul întoarce pagina în format HTML
- comunicația presupune existența protocolui IP pentru adresare și TCP pentru transportul datelor între server și client
- porturile implicite sunt: 80 și 443 pentru comunicare encriptată
- HTTP/1.0: o singură cerere-răspuns per conexiune
- HTTP/1.1: conexiunea stabilește o sesiune cu oricâte cereri-răspuns
- oferă autentificare de tip utilizator-parolă
- clientul mai este denumit și identificat ca *User Agent*

Cereri HTTP

- OPTIONS – afișează cererile HTTP acceptate de server
- GET – cere o resursă: pagină web, document etc.
- HEAD – la fel ca GET dar fără a prelua resursa, doar datele despre ea
- PUT – transmite date noi și le stochează la URL cerut
- POST – transmite date noi cu care inițializează o acțiune pe server
- DELETE – șterge resursa de la URL-ul transmis
- PATCH – aplică modificări resursei de la URL-ul transmis
- CONNECT – crează un tunel TCP/IP; pachete encriptate prin conexiune HTTP
- TRACE – funcție de tip ecou folosită la depanare; în general blocată

GET

- obține paginile web pentru *user agent*
- format simplu: cale către fișier și numele host-ului

```
GET /index.html HTTP/1.1
host: fmi.unibuc.ro

HTTP/1.1 200 OK
Content-Type: text/html
Accept-Ranges: bytes
ETag: "2065617787"
Last-Modified: Mon, 25 May 2015 15:34:17 GMT
Content-Length: 5940
Date: Thu, 19 Apr 2018 09:35:53 GMT
Server: lighttpd/1.4.26

[content]
```

PUT

- acceptă datele primite și le stochează la URL-ul cerut
- funcționalitate de tip upload `ftp` sau `sftp`
- de oricâte ori este apelat PUT rezultatul este același
- `Content-type` – tipul conținutului; trebuie înțeles de server
- `Content-length` – mărimea conținutului în bytes

```
PUT /new.html HTTP/1.1
Host: example.com
Content-type: text/html
Content-length: 16

<p>New File</p>
```

POST

- transmite date noi și inițializează o acțiune pe server
- acțiunea depinde de server și conținut
- poate fi redusă la o operațiune tip PUT
- poate crea o resursă la altă adresă
- apelat repetat, poate avea efecte diferite asupra conținutului serverului
- este folosit pentru a scrie pe forumuri

```
POST / HTTP/1.1
Host: example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 21

subj=Hello&msg=World
```

Coduri de eroare

- prima linie din răspuns la cerere; format din 3 cifre
- exemplu: `HTTP/1.1 200 OK`
- anunță ce tip de răspuns urmează
- 1xx – mesaj informativ
- 2xx – operația a decurs cu succes
- 3xx – redirectare (ex. pagina a fost mutată în alt director)
- 4xx – eroare cilent (ex. pagina cerută nu există)
- 5xx – eroare server (ex. cererea a expus un defect în server)
- coduri des întâlnite
  - 404 – pagina nu a fost găsită
  - 403 – acces interzis
  - 502 – server-ul funcționa ca gateway și nu s-a putut comunica cu un server extern
  - 301 – site-ul a fost mutat permanent la adresa cuprinsă în răspuns

User Agent

Software care acționează în numele utilizatorului

- implicit reprezintă browser-ul și conține date legate de el
- exemplu date: nume, versiune, sistem de operare etc.
- Firefox: `Mozilla/[version] ([system and browser information]) [platform] ([platform details]) [extensions]`
- concret: `Mozilla/5.0 (X11; OpenBSD amd64; rv:59.0) Gecko/20100101 Firefox/59.0`
- poate fi și un automat: robot, crawler, bot
- agenții automați trebuie să respecte regulile din fișierul `robots.txt`

robots.txt

- dictează ce directoare și fișiere pot fi accesate de automate
- servește un singur (sub)domeniu
- exemplu: separat pentru `fmi.unibuc.ro` și `unibuc.ro`
- directive
  - User-Agent – specifcă pentru ce agenți se aplică regulile
  - Disallow – căi și fișiere interzise
  - Allow – căi permise (ex. interzic un director întreg, dar permit un fișier anume din acel director)
  - Crawl-delay – perioada minimă în secunde între vizite
  - Host – specifică ce *mirror* să folosească pentru acces

Exemplu `http://fmi.unibuc.ro/robots.txt`

```
User-agent: *
Disallow: /devel
Disallow: /stest
Disallow: /vechi
Disallow: /rectorat
Disallow: /phpmyadmin
```

Autentificare

- limitarea accesului la anumite fișiere sau directoare
- permisiune prin autentificare cu utilizator și parolă
- metode interne incluse în protocolul HTTP
  - basic access
  - digest access
- există metode externe mai bune, puternice criptografic
  - autentificare prin cheie publică
  - kerberos: autentificare reciprocă client-server
  - cookie sessions: identificarea comunicației printr-o singură autentificare la început

Basic Access

- cea mai simplă formă de autentificare
- datele sunt trimise în clar în cerere
- nu folosește criptografie, cookie-uri sau alte mecanisme
- folosit împreună cu HTTPS pentru securitate

Digest Access

- autentificare `username:realm:password`
- realm: setul de pagini care folosesc același utilizator și parolă
- funcție de hashing (MD5) pentru a transmite acreditările
- poate fi folosit cu HTTP, deși nu este cea mai sigură metodă
- fișierul `.htdigest` conține datele generate cu `htdigest(1)`:
  `alex:fmi:5ea41921c65387d904834f8403185412`

Cookie

- comunicația HTTP nu reține contextul dialogului (*stateless*)
- de multe ori avem nevoie să reținem acțiuni trecute (*stateful*)
- cookie-urile sunt fișiere text mici pe care le reține browser-ul
- exemple: date de autentificare, istoric, ce buton a fost apăsat
- în funcție de scop, există mai multe tipuri
  - session – ținute în memorie temporară, expiră după ce utilizatorul părăsește site-ul
  - persistent – este salvată pe disk cu o dată de expirare
  - advertisment – folosite pentru reclame, conțin istoricul, sunt de tip persistent
- conține un nume, o valoare și zero sau mai multe atribute
- atributele au o structură cheie-valoare
- construcție
  - server: emite `Set-Cookie:` cu nume, valoare
  - client: la următoarele cereri pune cookie-urile în `Cookie:`

Cookie: exemplu comunicare

Clientul cere o pagină

```
GET /index.html HTTP/1.1
Host: fmi.unibuc.ro
```

Serverul răspunde și setează două cookie-uri

```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: lang=ro
Set-Cookie: sessionToken=deadbeef;
            Expires=Fri, 03 Jun 2018 12:00:00 GMT+2
```

Clientul cere o altă pagină și anunță ce cookie-uri folosește

```
GET /examene.html HTTP/1.1
Host: fmi.unibuc.ro
Cookie: lang=ro; sessionToken=deadbeef
```

HTTPS prin TLS

- comunicația HTTP se face în mod normal prin text
- oricine din rețea poate vedea traficul
- soluție: transmiterea pachetelor HTTP criptat
- protocolul HTTP rămâne același
- peste este adăugat protcolul Transport Layer Security (TLS)
- TLS este proiectat exact pentru astfel de operați
- alte exemple de protocol necriptat învelit în TLS: FTP, SMTP, IMAP
- în URL protocolul devine `https` și portul implicit 443
- criptografia este bazată pe chei publice certificate de o autoritate
- certificarea garantează identitatea posesorului cheii private

Certificate electronice

Un certificat conține

- informații legate de cheia asimetrică
- informații legate de identitatea posesorului cheii private
- posesorul mai este denumit și subiect (*subject*)
- semnătura digitală a unui terț care a verificat conținutul certificatului
- acest terț mai este numit emițător (*issuer*)

Cheia publică va fi folosită de client dacă

- semnătura este validă
- programul care verifică certificatul are încredere în emițător
- rezultă un lanț al slăbiciunilor (*chain of trust*)

Din 2016 certificare gratuită prin Let's Encrypt!

HTML

- conținutul oferit de server este în format HTML
- la baza limbajului: `<tag>content</tag>`
- tag-ul decide cum este afișat conținutul
  - `<html>` – demarcă începutul și sfârșitul documentului
  - `<head>` – conține descrierea paginii
    - `<title>` – titlul paginii
    - `<meta>` – cuvinte cheie folosite la căutare
    - `<base>` – calea absolută; baza link-urilor relative din pagină
  - `<body>` – demarcă începutul și sfârșitul documentului
  - `<header>`,`<h1-6>` – titlu de mare (1) la mic (6)
  - `<div>`,`<p>`,`<br>` – secțiune, paragraf, linie nouă
  - `<b>`,`<i>`,`<u>` – bold, italic, underline
  - `<a>`, `<img>` – hyperlink, imagine
- tag-urile pot avea și atribute stabilite la început
  - `<div style="background-color:lightblue">...</div>`
  - `<a href="experience.html">Experience</a>`
  - `<img src="images/email.png" alt="[E-mail]" />`

Common Gateway Interface (CGI)

- protocol prin care serverele web execută programe normale
- scopul acestor programe este de a genera pagini HTML
- implementarea serverului decide modul de execuție
- programele primesc o formă brută HTML prin varibile de mediu (ca `$PATH`)
- majoritatea variabilelor de mediu sunt prefixate cu `$HTML_`
- acestea conțin datele de intrare pentru program
- în general folosit în câmpuri `<form>`: butonul de *submit* după ce a fost tastat un mesaj pe forum
- când butonul este apăsat, conținutul este trimis unui program care execută mai multe operații
  - vaildează datele de intrae
  - înlocuiește formule speciale în text cu imagini sau cod HTML
  - adaugă mesajul în baza de date și actualizează profilul utilizatorului care a făcut operația

Limbaje ce folosesc protocolul CGI

- programele se pun de obicei în directorul `cgi-bin`
- istoric: programe specializate pentru site scrise în C
- limbajele de scripting existente profită și oferă programe generice CGI care oferă funcționalitate completă
- modul de funcționare a unui program CGI pentru un limbaj
  1. programul CGI primește datele de intrare
  2. datele sunt transformate într-o formă acceptată de compilator
  3. datele sunt trimise printr-un socket compilatorului
  4. procesarea și trimiterea datelor rezultate de către compilator
  5. aplică încă o operație de transformare a datelor din formatul primit de la compilator în cod HTML
  6. trimite serverului pagina web
  7. serverul trimite pagina clientului
- exemple: Perl, Ruby, Python