SSL/TLS
La protection HTTP

Stéphane Natkin

2006

SSL/TLS

Service

- SSLV3.0 et TLS offrent des connexions asymétrique et possédant toutes les fonctionnalités des connexions TCP. Elles assurent en outre :
- Une authentification forte d'un ou des deux parties en utilisant un système de certification basé sur le RSA, Diffie Hellman ou le DSA. Le protocole ne précise rien sur la gestion des certificats proprement dite.
- Une protection contre les attaques d'interception: une fois la connexion établie l'échange est garanti se dérouler entre les parties authentifiées.
- Une protection en intégrité des messages et du flux de messages, par utilisation conjointe d'un numérotation des messages et une fonction de hachage sécurisée (basée sur le MD5 ou SHA1).

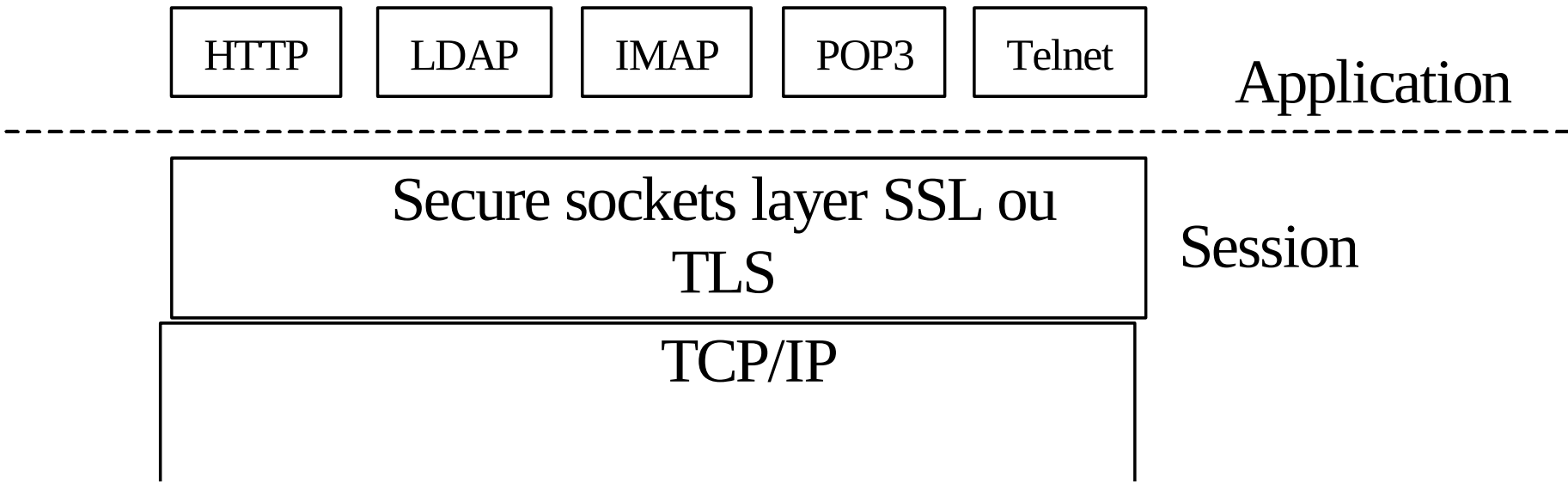
Service (2)

- Optionnellement la compression des données en utilisant tout algorithme de compression implanté de part et d'autre.

- Optionnellement une protection en confidentialité des données en utilisant tout algorithme cryptographique symétrique implanté de part et d'autre et une clef de session unique par connexion. La plus part des implantations supportent le DES, le 3DES (à clefs de 112 et 168 bits), le RC5.

- La suite cryptographique utilisée est négociée à l'ouverture de connexion.

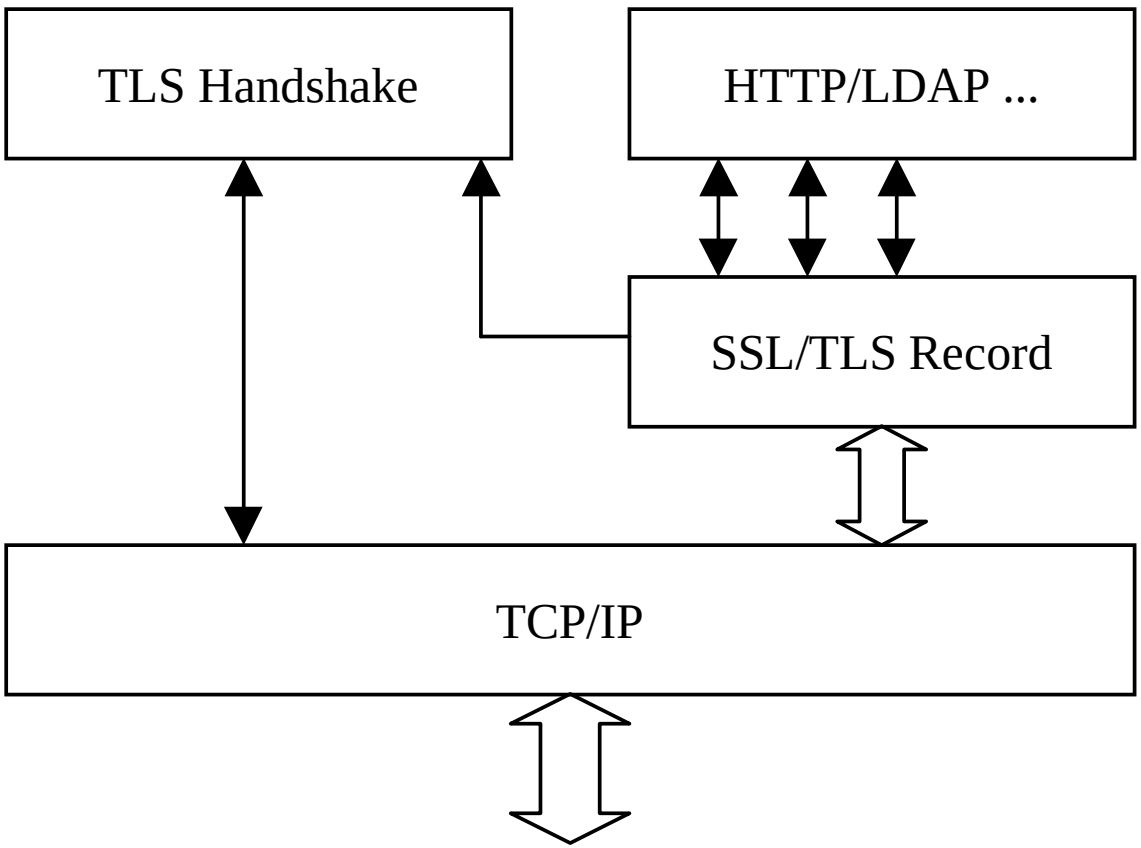
Positionnement architectural



Architecture

- 2 sous couches:
  - SSL record protocol réalise les fonctions de sécurité essentielles
  - Au dessus trois protocoles gèrent les paramètres et les erreurs:
    - Handshake, Change Cipher Spec, Alert
- Deux niveau de persistance
  - Connexion, correspond aux connexions TCP + Paramètres de sécurité propres
    - A transport with some service, associated with a session
  - Session
    - Créé par Handshake, gère les contextes communs à plusieurs connexions

Architecture



Contexte de session

- Un identifiant de session.
- Un certificat de l’entité distante. éventuellement nul.
- Une méthode de compression.
- Les spécifications du chiffrement. algorithme de chiffrement symétrique (nul, DES, etc.) algorithme de hachage (comme MD5 et SHA-1).
- La clé maître. Un secret de 48 octets partagé entre le client et le serveur.
- Un indicateur indiquant si la session peut couvrir plusieurs connexions TCP.

Contexte de connexion

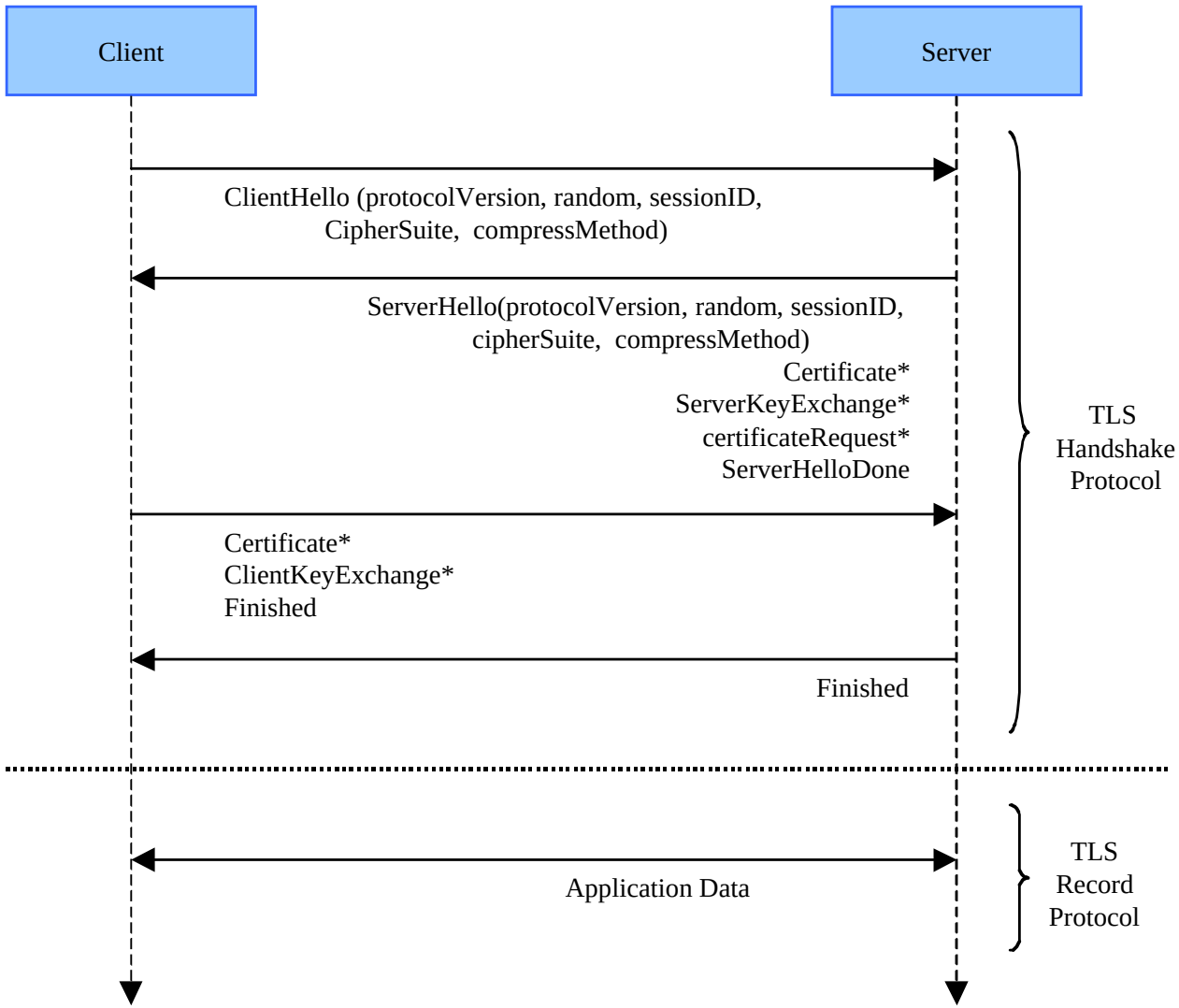
- Server_random et Client_random: nonces aléatoires de 32 octets, générés par le client et le serveur lors de chaque connexion.
- Server_MAC_write_secret: clé secrète utilisé par le serveur pour calculer les MACs
- Client_MAC_write_secret: clé secrète utilisé par le client pour calculer les MACs
- Server_write_key: clé symétrique utilisé par le serveur pour le chiffrement des données.
- Client_write_key: clé symétrique utilisé par le client pour le chiffrement des données.
- Initialization vectors: vecteur d'initialisation pour le chiffrement par bloc en mode CBC (Cipher Bloc Chaining), l'un du coté serveur et l'autre du coté client.
- Sequence number: chaque message est numéroté, l'un pour le serveur, l'autre par le client, et chacun codé sur 8 octets

TLS Handshake

Le protocole TLS Handshake consiste en une suite de trois sous protocoles qui sont utilisées par les entités communicantes pour s'authentifier entre elles, créer un contexte de sécurité négocié utilisé ensuite par TLS Record et gérer et signaler des conditions d'erreur:

- Négociation
- Changement des paramètres de chiffrement
- Alerte

MSC de Handshake et Record



Authentification du serveur

Suite à la requête d'un client, le serveur envoie son certificat au client et lui liste les algorithmes cryptographiques, qu'il souhaite négocier.

Le client vérifie la validité du certificat à l'aide de la clé publique du CA (Certificate Authority) contenue dans le navigateur.

Si le certificat est valide, le client génère un pré-master secret (PMS) de 48 octets qui servira à dériver le master secret (MS) de même taille.

Le PMS est chiffré avec la clé publique du serveur puis transmis à ce dernier. Les données échangées par la suite entre le client et le serveur sont chiffrées et authentifiées à l'aide de clés dérivées de la clé maître.

Authentification (optionnelle) du client

Le serveur (et seulement lui) peut demander au client de s'authentifier en lui demandant tout d'abord son certificat.

Le client réplique en envoyant ce certificat puis en signant un message avec sa clé privée :ce message contient des informations sur la session et le contenu de tous les échanges précédents.

Messages de HANDSHAKE

- **ClientHello** : ce message contient: la version: version du protocole SSL,client_random: nombre aléatoire, session_id:l'identificateur de session, cipher suite :la liste des suites de chiffrement choisies, et algo décompression: la liste des méthodes de compression.
- **ServerHello** : ce message contient: la version: version du protocole SSL,Server_random: nombre aléatoire, session_id:l'identificateur de session, cipher suite :la liste des suites de chiffrement choisies, et algo décompression: la liste des méthodes de compression.
- **Certificate** : ce message contient soit le certificat de serveur
- **ServerKeyExchange** : contient le certificat de signature
- **CertificateRequest** : le serveur réclame un certificat au client
- **ServerHelloDone** : la fin de l'envoi de message
- **ClientKeyExchange** : ce message contient le PreMastersecret crypté à l'aide PreMastersecret crypté à l'aide de la clé publique du serveur.
- **CertificateVerify** : vérification explicite du certificat du client
- **Finished** : fin du protocole Handshake et le début de l'émission des données

Suite cryptographique

Forme SSL_X_WITH_Y_Z où :

- X désigne l'algorithme utilisé pour l'échange de clés : RSA ou Diffie-Hellman avec signature DSS ou RSA.
- Y désigne l'algorithme de chiffrement
- Z l’algorithme de hachage

Exemple SSL_RSA_WITH_DES_CBC_MD5

Génération des paramètres cryptographiques

- Création du Master secret.
  - Le pre-master-secret est échangé dans un premier temps.
    - RSA, or Diffie-Hellman.
  - Les deux parties calculent le Master Secret à partir du pre-master-secret et des nonces.
- Génération des paramètres cryptographiques.
  - Clefs symétiques client/servuer d’intégrité (MAC) et de confidentialité, vecteur d’intialisation (IV) pour le CBC.

Master secret

- Client génère (RSA ou DH) un pre-master-secret de 48 octets s_p
- Master secret:
  - s_m=MD5(s_p|SHA('A'| s_p|r_c|r_s)) | MD5(s_p|SHA('BB'| s_p|r_c|r_s)) | MD5(s_p|SHA('CCC'| s_p|r_c|r_s))
  - ou $r_{c,s}$: nonce client, serveur

Clefs de chiffrement

Clef de session: même principe le master secret est utilisé à la place de s_p pour générer une suite binaire don't les éléments sont les clefs et IV.

SSL Record Protocol

- 3 services:
  - Confidentialité, intégrité des messages et du flot de messages, compression
- Opérations:
  - Fragmentation
  - Compression des données
  - Calcul d'un message authentication code (MAC) = h(s:*secret*|m:*message*)
  - Chiffre avec la clef client (cw) ou serveur (sw)
  - Transfère via TCP

PDU Record

- Intégrité: Données, MAC (16/20 octets)
- Confidentialité chiffrement par blocs: Données, MAC, Padding
- Confidentialité chiffrement en continu Données, MAC

Remarques

- L'authentification du client est facultative et au bon vouloir du serveur. Elle est en fait rarement utilisée
- L'authentification du réseau intervient dans tous les cas avant celle du client.
- Comme dans IPSec (IKE phase 1), l'authentification reprend les échanges précédents et valide ainsi tout le handshake.
- Seule la clé publique du serveur est utilisée pour faire du chiffrement; celle du client ne sert que pour de la signature.

ALERT

Génère des messages d'alerte suite aux erreurs que peuvent s'envoyer le client et le serveur.

Type d’erreur Fatal ou Warning.

Fatal, la session SSL est abandonnée:

- bad_record_mac: réception d'un MAC erroné
- decompression_failure: les données appliquées à la fonction de compression sont invalides
- handshake_failure: impossibilité de négocier les bons paramètres
- illegal_parameter: un paramètre échangé au cours du protocole Handshake ne correspondait pas avec les autres paramètres
- unexpected_message: message non reconnu.

ALERT (2)

Les warnings :

- bad_certificate: le certificat n'est pas bon
- certificate_expired: certificat périmé
- certificat_revoked: certificat révoqué
- certificat_unknown: certificat invalide pour des raisons précisés au dessus
- close_notify: la fin d'une connexion
- no_certificate: réponse négative à une demande de certificat
- unsupported_certificate: le certificat reçu n'est pas reconnu

Implantation

- Mode Proxy
- Mode intégré (navigateurs)
- Bibliothèques
- Nombreuses offrent serveur
  - » SSLeay
  - » Netscape Entreprise Server
  - » Apache
  - » Oracle Web Apllication Server
  - » Internet Information Server (IIS)
  - » Lotus Domino d'IBM
  - » Java Server de Sun Microsystems

Ports SSL

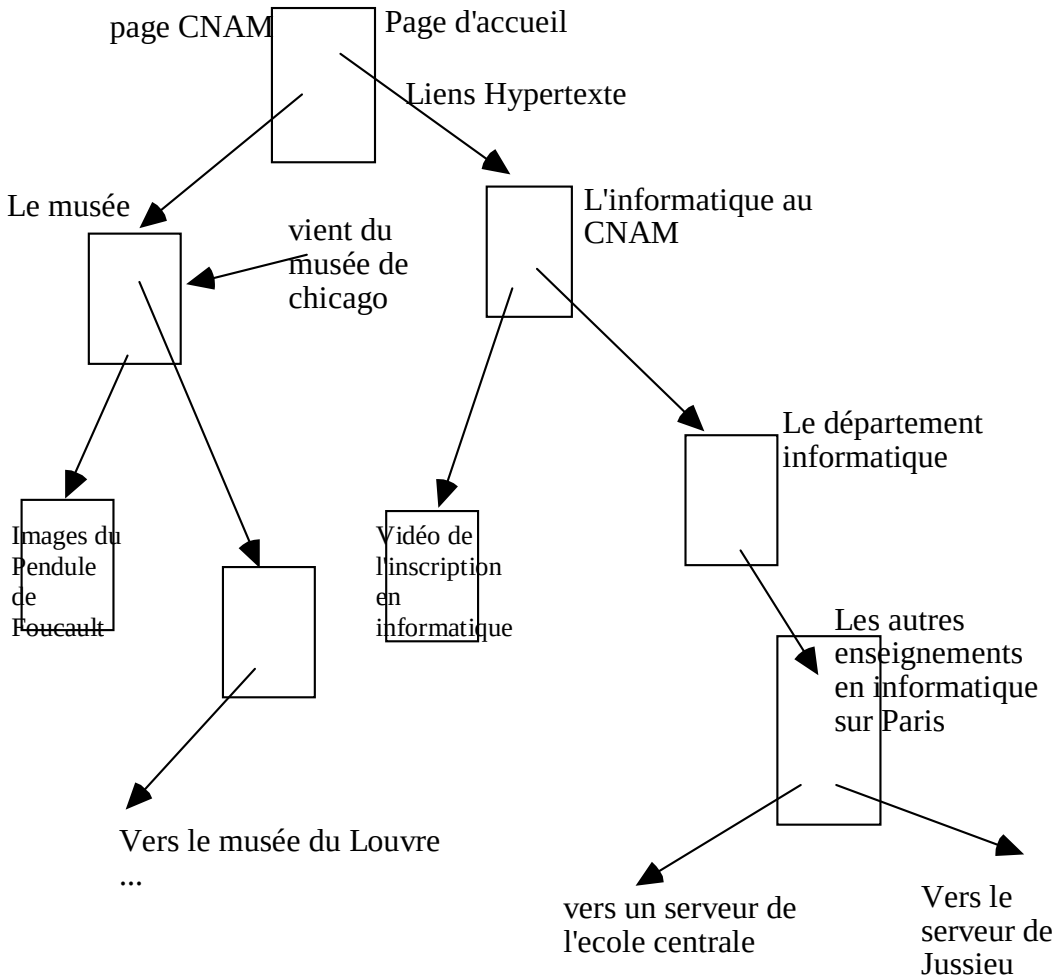
- HTTPS (HTTP en SSL) 443
- SMTPS (SMTP en SSL) 465
- NNTPS 563
- LDAPS (LDAP en SSL) 636
- POP3S 995
- IMAPS 995
- TELNETS992

HTTP

HTTP
terminologie du WWW

- **HTTP** :HyperText Transfert Protocol : protocole de transport de pages hypertexte/hypermédia
- **URL**: :Uniform Ressource Locator : adresse planétaire d'une ressource
- **HTML :** Langage Hypertexte qui permet de décrire les documents hypertextes gérés par le Web

Documents hypermédia



Format d ’un URL

protocole : chemin-d-acces

forme BNF :

```
nom-protocole:nom-serveur[:port]{/répertoire}*/resso
urce
```

Exemples d'URL :

```
http://www.cnam.fr/Marillion/Marillion.html
```

```
news:fr.rec.cuisine
```

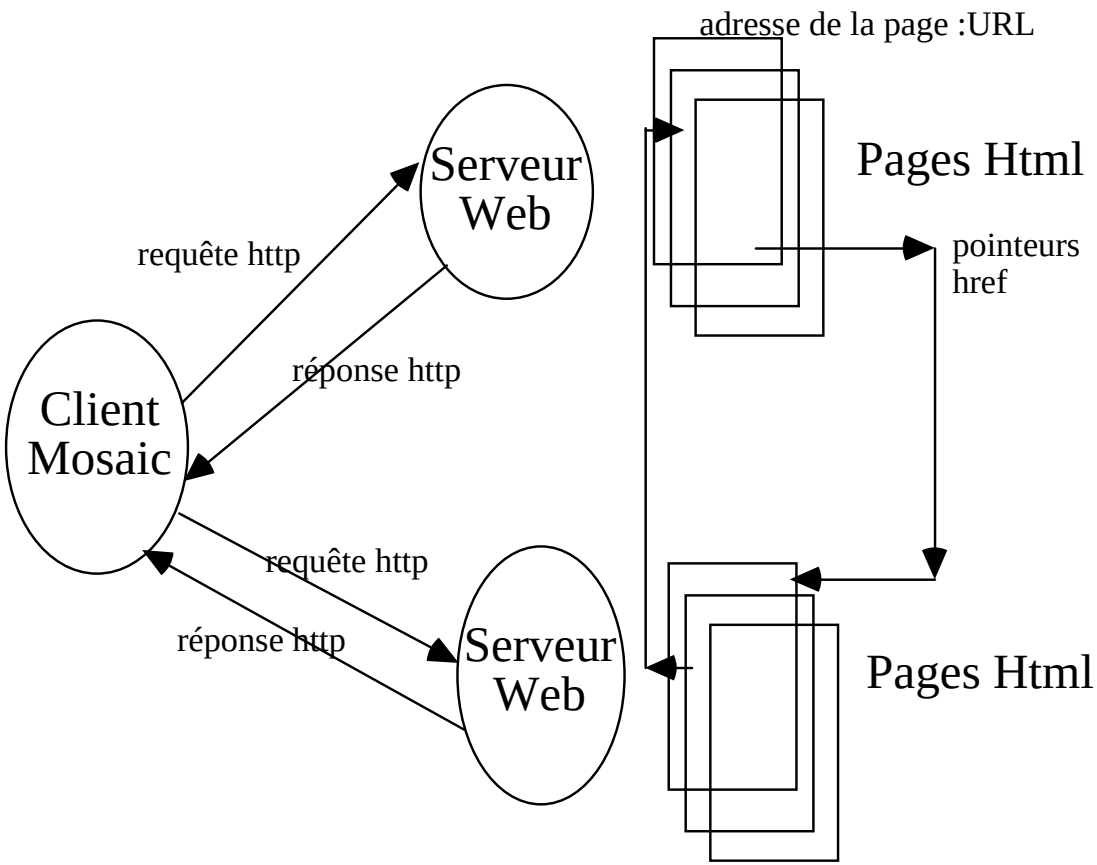
```
ftp://ftp.cnam.fr/pub/Atari/0new0
```

HTML

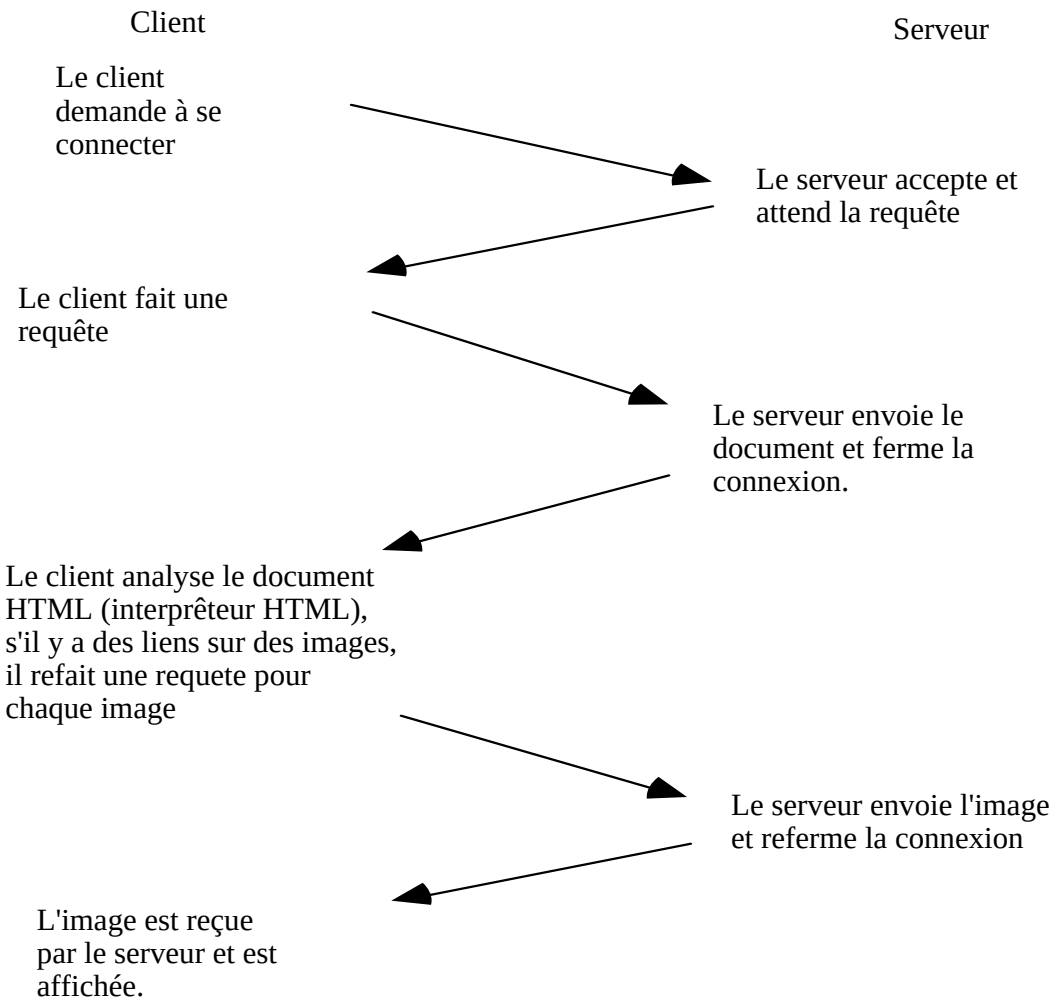
HyperText Markup Language

```
<TITLE>CNAM</TITLE>
<h1>Conservatoire National des Arts et Métiers, une tradition d'avance</h1>
<img src="/Images/cnam.gif">
<p>Yes, we do have a <a href="/index_english.html">home page in English</a>.
<p>
Bienvenue sur le serveur du CNAM, à;
<a href="http://www.cnam.fr/louvre/paris/">Paris</a>, France (vous ne
<a href="http://meteora.ucsd.edu/~norman/paris">connaissez pas Paris</a> ?).
Vous pouvez venir
<a href="http://metro.jussieu.fr:10001/bin/ext/french/france/paris/set1">en métro</a>.<p>
Vous avez des <a href="http://web2.cnam.fr/cnam/ACCUEUIL_CNAM.html">
informations pratiques</a> sur le CNAM et
des <a href="/Images/CNAM/">photos</a>. Vous pouvez aussi consulter le
<a href="http://web2.cnam.fr/vms/equipe/am3.html">catalogue de la
    biblioth&egrave;que</a>.<p>
<img src="/museum/ChapelleP1D.GIF"><p>
Visitez le <a href="/museum/">Musée
des Arts et Métiers</a>,
notre musée virtuel des techniques (photos et textes en français).<p>
```

Fonctionnement d ’un serveur Web



HTTP: principe de fonctionnement



HTTP
format des requêtes

Format d'une requête:

ligne requête
en-tête (0 ou plus)
<ligne blanche>
Corps (seulement sur la requête POST)

Le format de ***ligne requête*** est
requête URL HHTP-version

Types de requêtes

Les types de requêtes supportées en HTTP 1.1 sont :

- GET : accès en lecture à une URI,
- HEAD : accès en lecture à l'en tête d'une URI (permettant de connaître sa structure),
- POST : Passages de paramètres vers un URI dynamique (un script CGI par exemple)
- PUT : Créer ou modifier une URI
- DELETE : Détruire une URI
- TRACE : permet de tester l'accès par demande d'un retour de commande.

Format des réponses

état de ligne

en-tête (0 ou plus)

<ligne blanche>

Corps (seulement sur la requête POST)

Le format de état de la ligne est

HHTP-version code-de-réponse phrase de réponse

Exemples (1)

```
% telnet cedric.cnam.fr 80
Trying 163.173.136.10...
Connected to tulipe.cnam.fr.
Escape character is '^]'.
GET /index.html HTTP/1.0

HTTP/1.0 200 Document follows
Date: Fri, 25 Apr 1997 14:21:54 GMT
Server: NCSA/1.5.2
Last-modified: Fri, 29 Nov 1996 11:10:41 GMT
Content-type: text/html
Content-length: 2272

<title>
Centre d'Etude et de Recherche en Informatique du CNAM
</title>
<img src="LogoCedric.gif" alt="[CEDRIC}"><p>
<strong>Bienvenue au CEDRIC, le Centre d'Etude et de
  Recherche en Informatique
du <a href="http://www2.cnam.fr/">CNAM</a>!</strong><p>
Le laboratoire regroupe des
enseignants-chercheurs                du                <a
  href="http://deptinfo.cnam.fr/">DÈpartement d'Infor
matique du CNAM</a> ‡
<a
  href="http://meteora.ucsd.edu/~norman/paris/">Paris</a>,
....
```

Exemples (2)

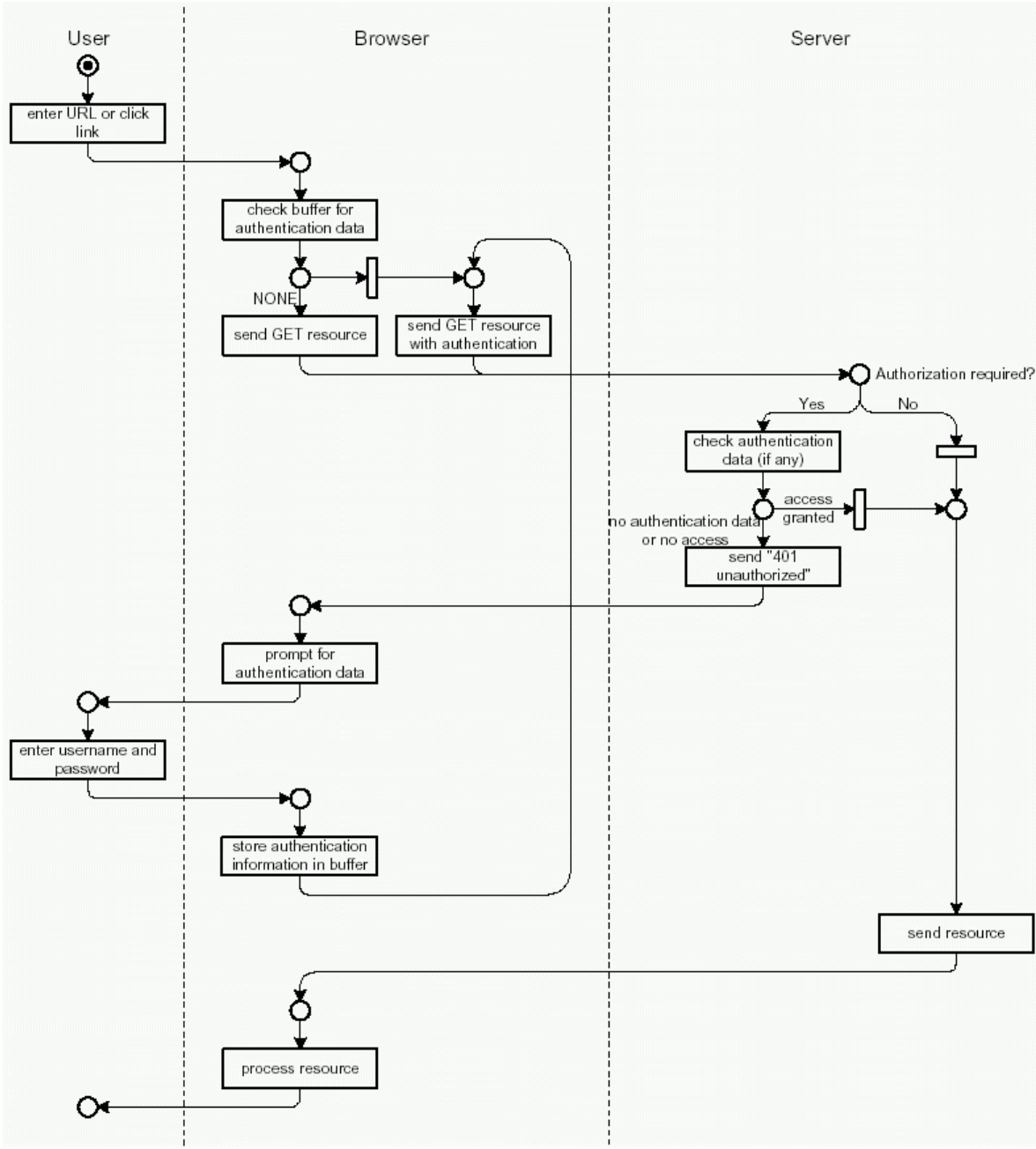
```
telnet cedric 80
Trying 163.173.136.10...
Connected to tulipe.cnam.fr.
Escape character is '^]'.
GET ind.html HTTP/1.0

HTTP/1.0 404 Not Found
Date: Fri, 25 Apr 1997 14:33:47 GMT
Server: NCSA/1.5.2
Content-type: text/html

<HEAD><TITLE>404 Not Found</TITLE></HEAD>
<BODY><H1>404 Not Found</H1>
The requested URL ind.html was not found on this server.
</BODY>
Connection closed by foreign host.
```

Asynchronisme du mode client/serveur HTTP

Le client n’attend pas (comme dans un RPC standard) la réception de réponse. Il continue à faire des requêtes en fonction de l ’analyse de la page. Sur un client standard il peut y a voir 4 à 8 requêtes simultanées en cours de traitement

En HTTP 1.1 il peut demander au serveur de ne pas fermer la connexion.

Authentification dans HTTP



Modes d’authentification du client

- En clair: très vulnérable
- Résumé (Hash) MD5: le serveur envoie une requête avec un nonce, le client hache son login le password et le nonce
- En théorie doit être refaite à chaque accès
- Vulnérable aux attaques par le milieu

Contrôle d’accès

Les serveurs utilisent ce mécanismes conjointement à des listes de contrôle d’accès. Les droits portent sur type de requête et la ressource (URI : Unified Ressource Locator) demandée.

Il est possible aussi d’attribuer des droits à un groupe. En l’absence d’authentification ce mécanisme permet d’autoriser ou d’interdire un type de requête à tout le monde.

HTTPS

HTTP au dessus de SSL/TLS

Accès par une URL

- https://webmail.cnam.fr/src/login.php?secure_login=yes
- Le serveur doit s'authentifier par certificat
- Le client peut soit s'authentifier par certificat soit par un autre mode, une fois la session SSL ouverte
- Par exemple en utilisant l'authentification HTTP

Conclusion

- HTTPS est en théorie une bonne solution si:

1. Les certificats étaient gérés et contrôlés correctement
2. Les Pwd étaient bien gérés
3. Les serveurs étaient bien protégés

Avec des si…