Sécurité Applicative

M1 WEB - Introduction

Me. 26 Juin 2019 - PHELIZOT Yvan

```
var b64img = window.location.hash.substr(1);
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    var reader = new FileReader();
    reader.onloadend = function() {
      document.write(`
    <a href="${b64img}" alt="${atob(b64img)}">
     <img src="${reader.result}">
    </a>`);
    }
    reader.readAsDataURL(this.response);
  }
};
xhttp.responseType = 'blob';
xhttp.open("GET", b64img, true);
xhttp.send();
```

Où est la faille?
Comment l'exploiter?
Comment la détecter?
Comment la corriger?

Programme

- Rappels (0.5j.)
  - HTTP
  - HTML
  - Javascript
- OWASP Top 10 (1j.)
- Secure Coding (0.5j.)

Download **WebGoat 7**

https://bit.ly/2Fnfc8p

Qui suis-je?

- PHELIZOT Yvan
- Coach Sécurité chez Arolla
- yvan.phelizot@arolla.fr
- Secure Coding/Secure by (DD-)Design
- Meetup OWASP France à Paris

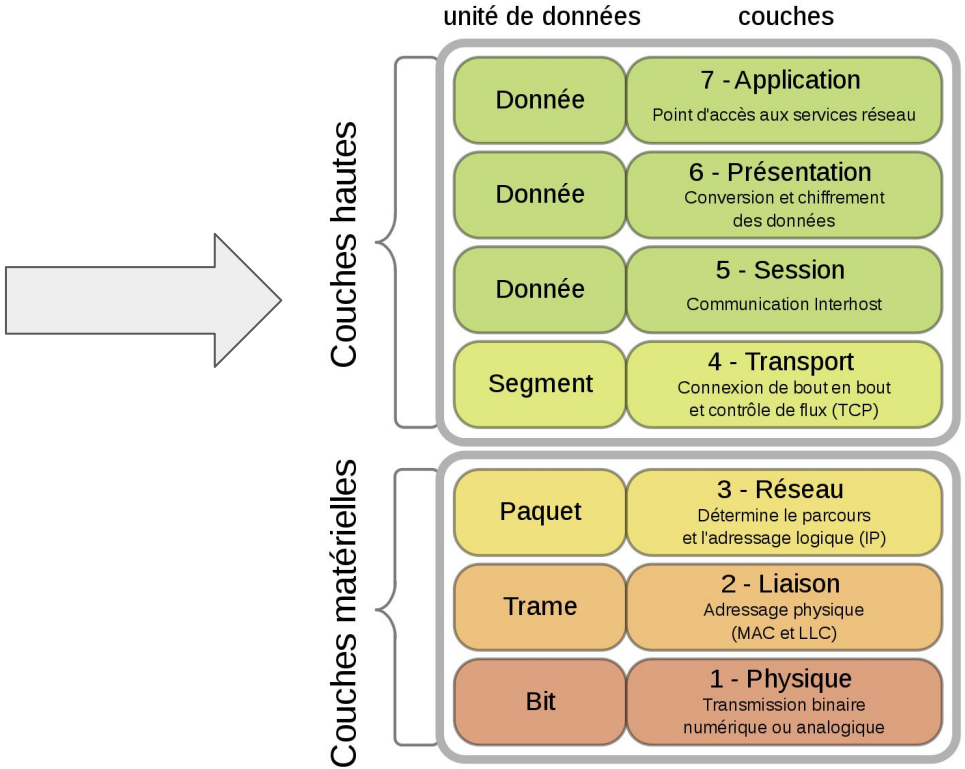
Rappels

HTTP

- HyperText Transfer Protocol
- Protocole sans état : requête autosuffisante
- Crée en 1990
- Version actuelle : HTTP 1.1
- Version future : HTTP/2

Modèle OSI

HTTP



Source: https://fr.wikipedia.org/wiki/Mod%C3%A8le_OSI

URL

- Uniform Resource Locator
- Identifiant pour accéder à une ressource

http://login:pwd@www.here.com:8888/chemin/d/acc%C3%A8s.php?q=req&q2=req2#s

- http ⇒ protocole
- login:pwd ⇒ login & password
- www.here.com ⇒ domaine & sous-domaine
- 8888 ⇒ port
- Échappement des caractères : %C3
- q=req : paramètres
- # : fragment

Protocoles

- http/https
- file
- ftp
- gopher, shttp, ...
- javascript:alert(1)
-

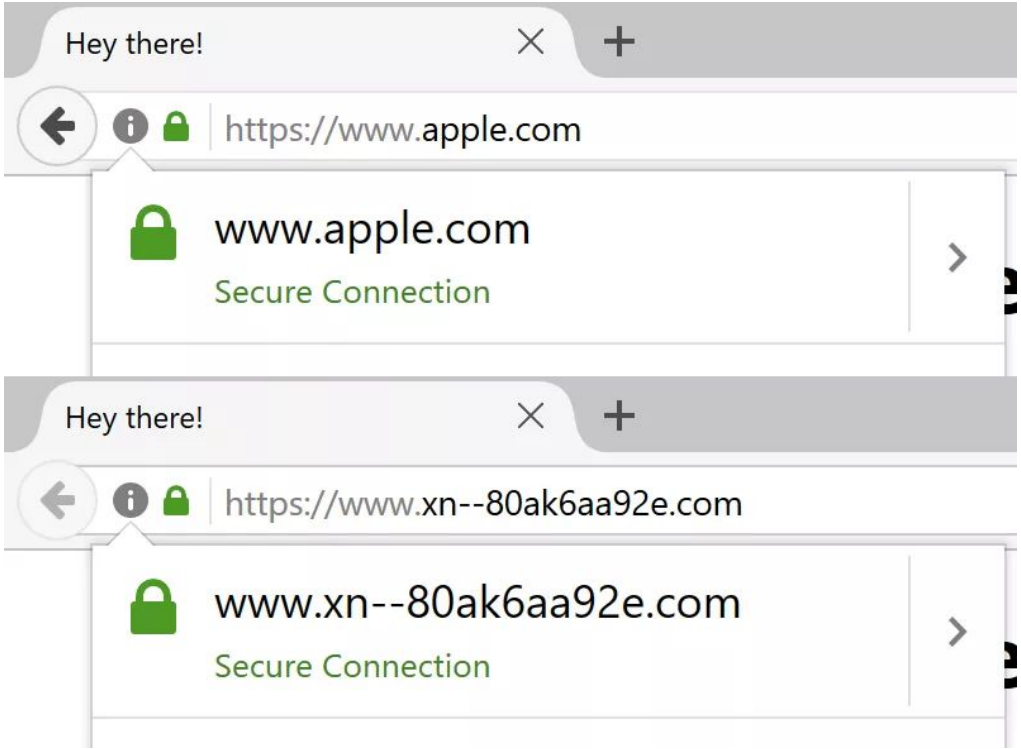
Domaine: valide ou non?

1. http://google.com/
2. http://www.google.com/
3. https://www.google.com/
4. https://216.58.213.164/
5. https://[2a00:1450:4007:808::2004]/

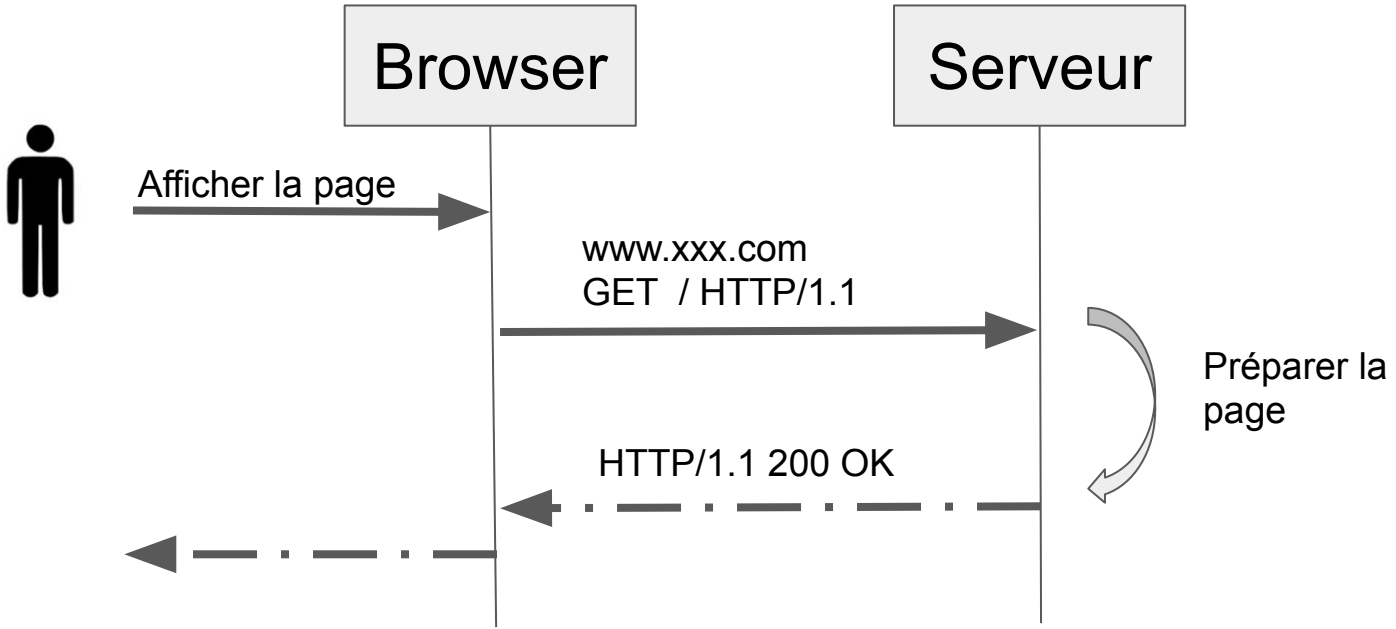
1. http://0xd83ad5a4
2. http://0x7f.1/
3. http://033016552644
4. //0xd83ad5a4

Homograph attack

- URL phising attack
- Exemple : https://www.xn--80ak6aa92e.com/ sous Firefox



Requêtes/Réponses



HTTP Request/Démo

curl -v http://www.lemonde.fr

> GET / HTTP/1.1

> Host: www.lemonde.fr

> User-Agent: curl/7.55.1

> Accept: */*

HTTP Response/Démo

< HTTP/1.1 301 Moved Permanently
< Location: https://www.lemonde.fr/
< Content-Length: 0
< Accept-Ranges: bytes
< Date: Sun, 30 Sep 2018 07:08:15 GMT
< Via: 1.1 varnish
< Age: 27
< Connection: keep-alive
< X-Served-By: cache-cdg20729-CDG
< Set-Cookie: prog-deploy=23; expires=Fri, 29 Mar 2019 07:08:15 GMT; path=/; domain=.lemonde.fr;
< Set-Cookie: prog-deploy2=78; expires=Fri, 29 Mar 2019 07:08:15 GMT; path=/; domain=.lemonde.fr;

Request Headers

- User-Agent
- Referer
- Content-Length
- Host
- Cookie
- DNT

Response Headers

- Content-Length
- Content-Type
- X-Content-Type-Options
- Set-Cookie
- X-XSS-Protection

- Access-Control-*
- X-Frame-Options
- Strict-Transport-Security
- Content-Security-Policy
- Expect-CT

HTTP Parameter Pollution

- http://example.com/action?do=xxx&do=yyyy
- Which one is used?

Cache

- Cache-Control
- Proxy
- Cache Poisoning

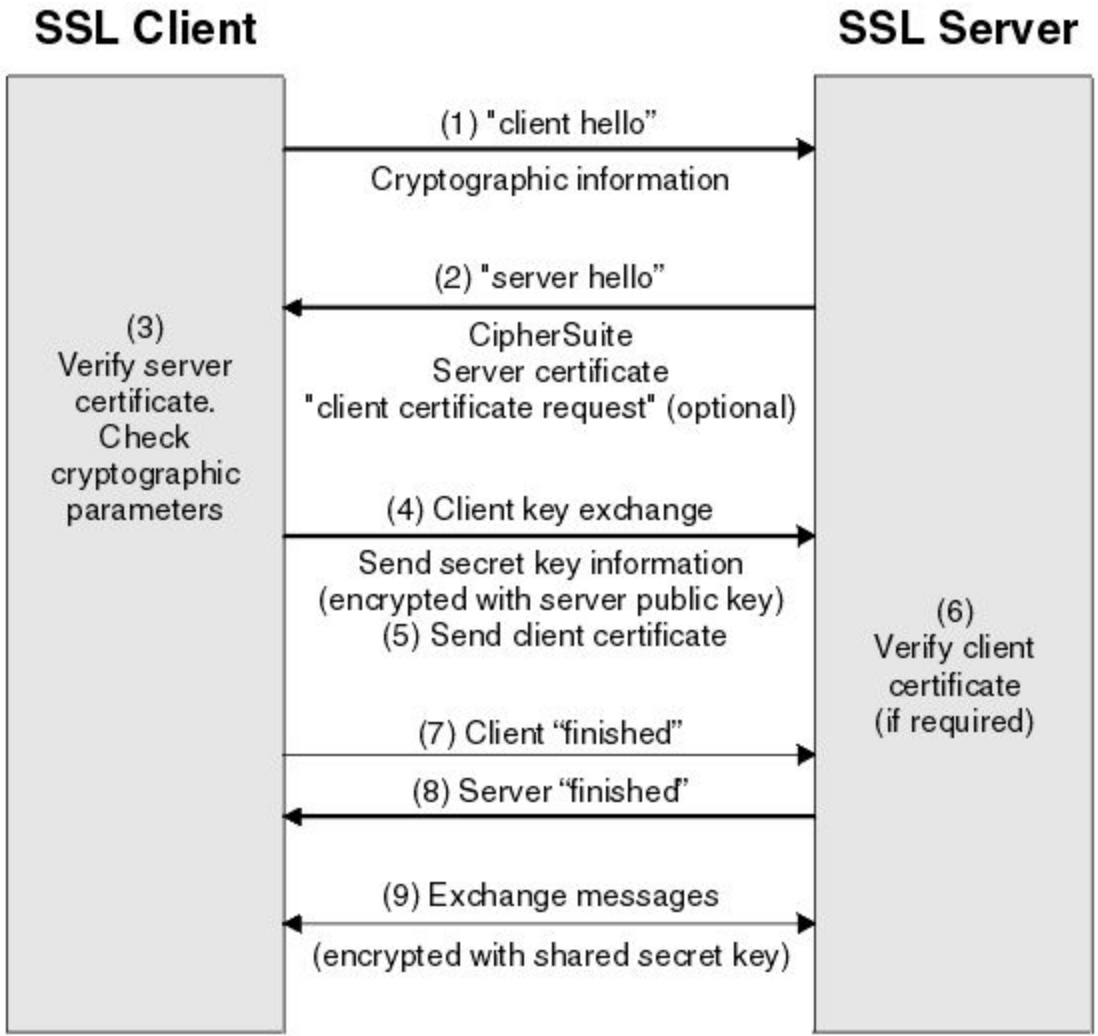
HTTPS

- HTTP for Secure communication
- Chiffrer les communications sur HTTP
- Histoire
  - 1994: SSL1.0
  - 1998 : TLS.10
  - SSL 3.0 (Deprecated)
  - TLS : 1.3 (Current)

HTTPS

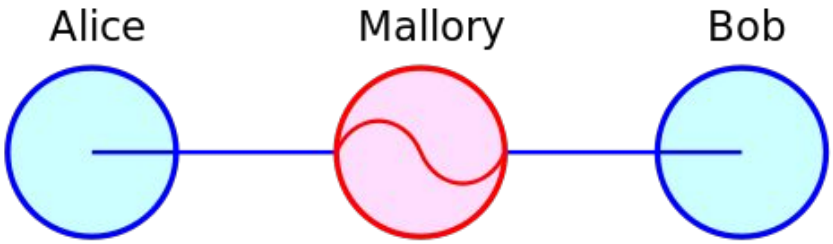
- Chiffrement & authentification du serveur ⇒ Certificat
- Chaine de validation
- Niveau de validation (Extended Validation Cert)
- Let’s Encrypt
- Google : importance

Regardons un certificat

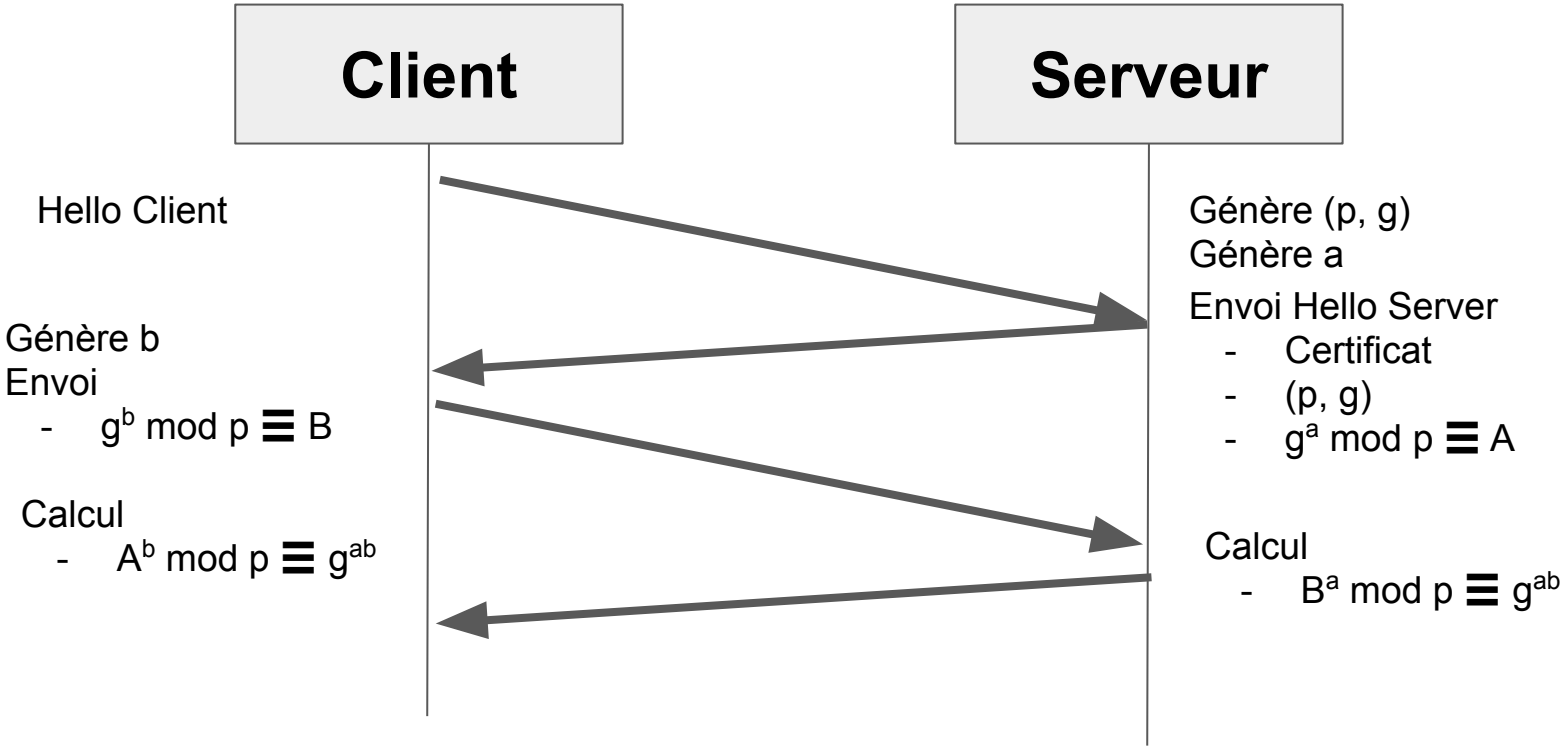

SSL Client
SSL Server
(1) "client hello"
Cryptographic information
(2) "server hello"
CipherSuite
Server certificate
"client certificate request" (optional)
(3)
Verify server
certificate.
Check
cryptographic
parameters
(4) Client key exchange
Send secret key information
(encrypted with server public key)
(5) Send client certificate
(6)
Verify client
certificate
(if required)
(7) Client "finished"
(8) Server "finished"
(9) Exchange messages
(encrypted with shared secret key)

Imaginons...

- Interception (ex: Man in the Middle) d'une communication HTTPS
- Deux ans plus tard, le certificat est volé
- Comment protéger la communication?



Perfect Forward Secrecy



Wireshark

HTTPS

Avantages

- Confiance dans les échanges
- Protection de la communication (privacy)

Désavantages

- Confiance excessive dans la chaîne
- Pas de cache possible
- Performances

Attaques HTTPS

- Protocol negotiation
- Brute-force
- Insecure conception
  - SSL 1.0
  - by design (https://en.wikipedia.org/wiki/Dual_EC_DRBG)
- Vol de certificat

Protection

- Redirect to HTTPS
- Durée de vie (Let’s encrypt)
- Révocation
- HSTS
- Certificate Transparency
- Public-key pinning (HPKP)

Session

- HTTP is stateless
- Solution : Session
- Permet à un utilisateur d’avoir accès à l’application sans devoir rentrer son mot de passe à chaque requête

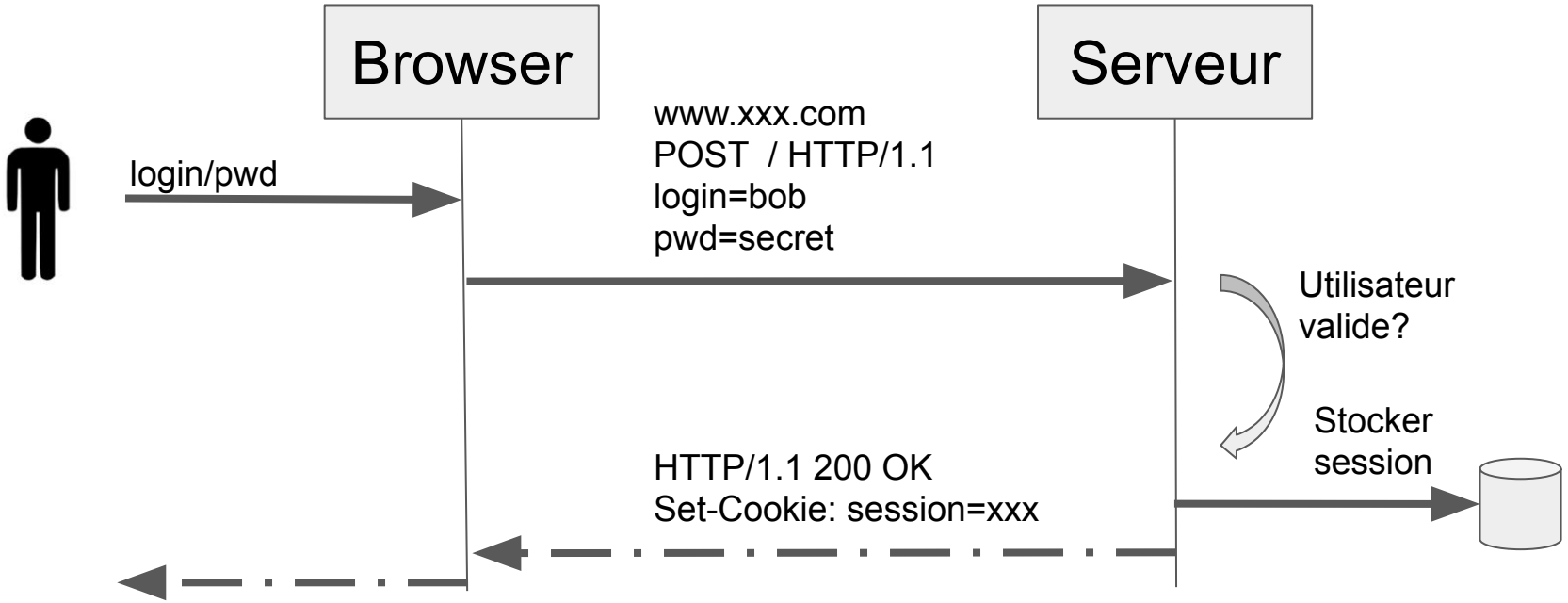
Où stocker le numéro de session?

http://www.monsite.com/page.php?id=45&PHPSESSID=0c6ca5b447035bbb274830f1ad7695bc

Solutions trouvées

- Cookie
  - Fichier texte stocké côté client
- Connexion : Token de session
- Informations dans les requêtes
  - Paramètres
  - Cookies

Connexion


Browser
Serveur
login/pwd
www.xxx.com
POST / HTTP/1.1
login=bob
pwd=secret
Utilisateur
valide?
Stocker
session
HTTP/1.1 200 OK
Set-Cookie: session=xxx

Lecture

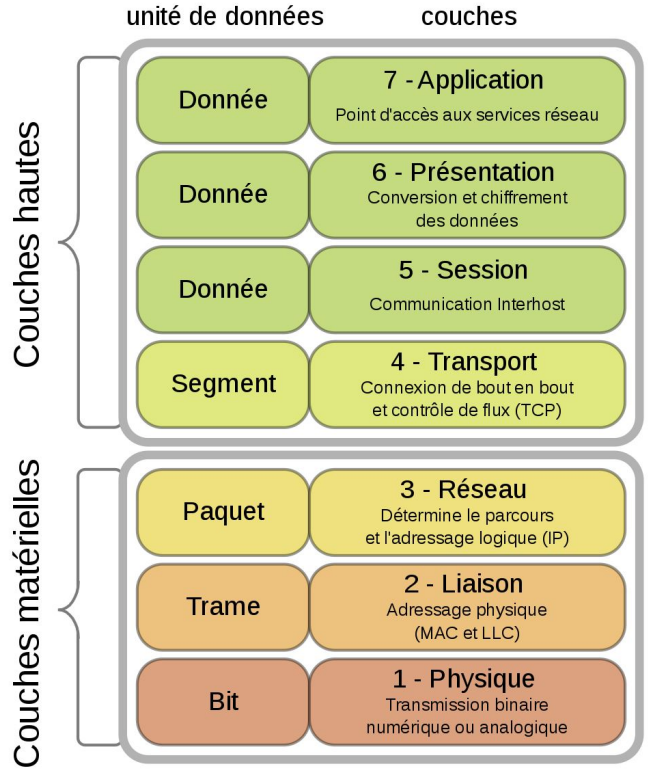

Browser
Serveur
Afficher page
www.xxx.com
GET / HTTP/1.1
Cookie: session=xxx
Récupérer
session
Session
valide?
HTTP/1.1 200 OK

Cookie

- Stocké côté client
  - Peut être changé
  - Chiffrement par le serveur
- Secure
- HttpOnly
- Domain
- Path
- Expires

Modèle OSI

HTML



Source: https://fr.wikipedia.org/wiki/Mod%C3%A8le_OSI

HTML

HTML

- Hyper Text Markup Language
- Structurer sémantiquement les pages
- Inclusions de multiples sources (images, …) de multiples origines
- Variable selon les navigateurs

HTML Encoding

- UTF-7, 8, 16, …?
- Notations:
  - A
  - &#x41
  - &#0000065
- https://codepen.io/anon/pen/yWjQEg

HTML - FRAME

- Inclure une page dans une autre page
- Accéder à des ressources protégées d’une autre page?
- Accéder à des ressources protégées d’un serveur?

HTML - SOP

- **Same Origin Policy** : restreint la manière dont un document ou un script chargé depuis une origine peut interagir avec une autre ressource chargée depuis une autre origine.
- Une iframe a une origine ⇒ **isolation**
- Une iframe peut avoir accès aux données d'une iframe de même origine (accès réseaux, accès DOM, cookies, …)
- Mais..

HTML - SOP

- Comment est calculé l’origine? ⇒ non consistante entre navigateurs
- N’empêche pas certaines requêtes (GET, POST)
- Nécessité de **"relaxer"** ces contraintes (CORS)
- Quid quand je download une page?

Javascript

Javascript

- Langage de programmation
- Inventé par Netscape en 1995
- Pages web dynamiques
- Node

Javascript

Valide ou non valide? Lequel produit une popup avec 1?

- alert(‘1’)
- alert(1)
- alert(/1/.source)
- alert(‘\u0031’)
- alert(`${0+1}`)
- top[8680439..toString(30)](1)

Cookie

Quid si j’arrive à mettre dans une page:

<img src=x
onerror="&#0000106&#0000097&#0000118&#0000097&#00
00115&#0000099&#0000114&#0000105&#0000112&#00001
16&#0000058&#0000097&#0000108&#0000101&#0000114&
#0000116&#0000040&#0000039&#0000088&#0000083&#00
00083&#0000039&#0000041">

- document.cookie

Session

- Interception: Token stealing/Session fixation
- Token prévisible
  - PRNG
  - Token ++

Protection HTTP Headers

- Cookie
  - httpOnly : cookie non accessible
  - Secure, Expires, Domain, Path, …
- X-Frame-Options
- X-XSS-Protection
- Access-Control-*
- CSP ...

⇒ Mozilla Observatory

Protection

- WAF : Web Application Firewall
- Application agnostic
- Blacklist filtering
- NGinx/Apache: mod_security
- Not a silver bullet