



POLYTECHNIQUE
MONTRÉAL

UNIVERSITÉ
D'INGÉnierie

INF4420: Éléments de Sécurité Informatique

Sécurité des applications web

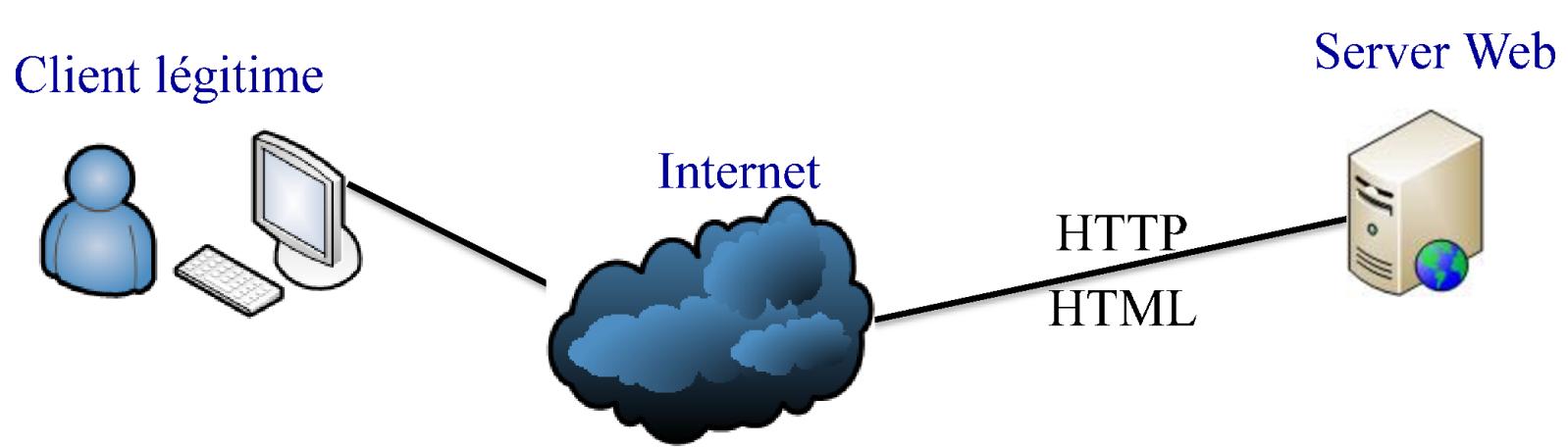


Contenu du cours

- Architecture des applications web
- Authentification
- SQL injection
- Cross site scripting
- Vérification des données usager
- Cross site request forgery
- Phishing (hameçonnage) et moralité de l'histoire



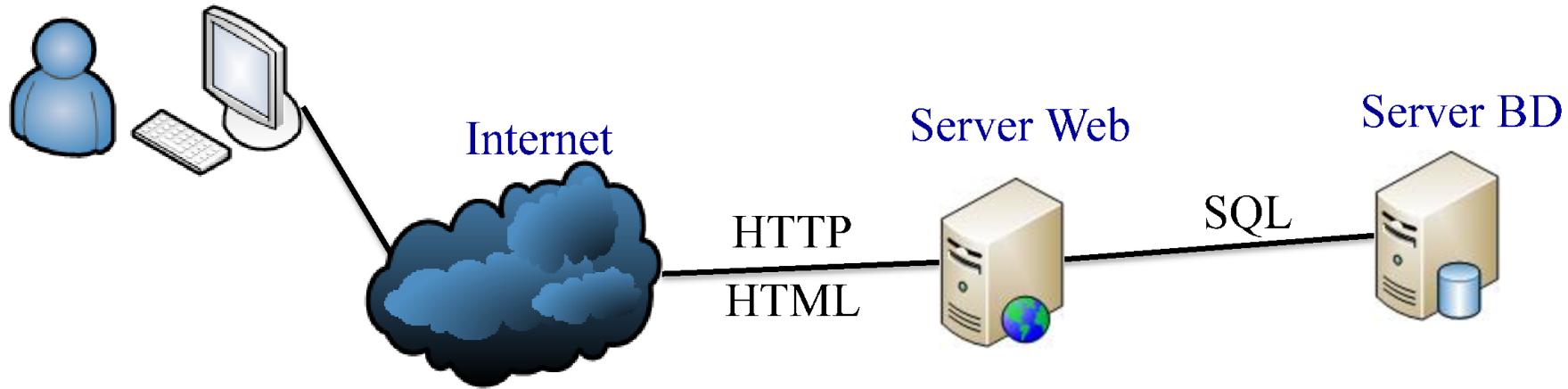
Architecture des applications web





Architecture des applications web

Client légitime

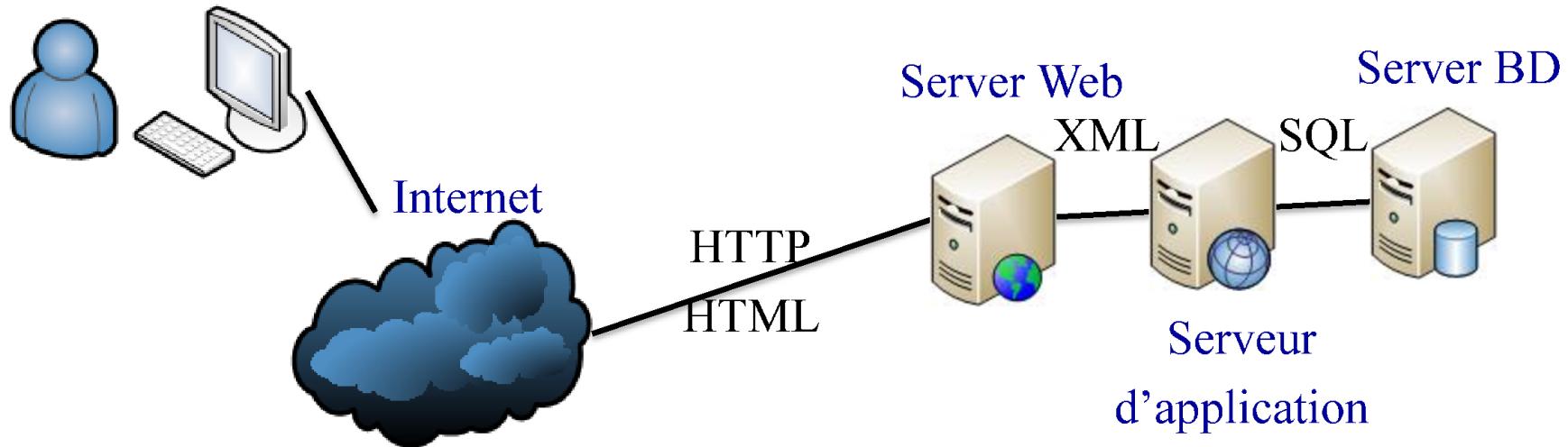


serveur DB pour ramener du contenu dynamique
sur le serveur web pour faire différents
types de transactions avec le client



Architecture des applications web

Client légitime



serveur application améliore l'interopérabilité aide
serveur web à gérer le contenu dynamique

serveur web servir du contenu statique comme
page html, image, texte stocké sur le système de
fichier du serveur.



Architecture des applications web

Client légitime



Internet



HTTP
HTML

Server Web



XML



SQL

Server BD



Serveur
d'application

Kerberos



LDAP

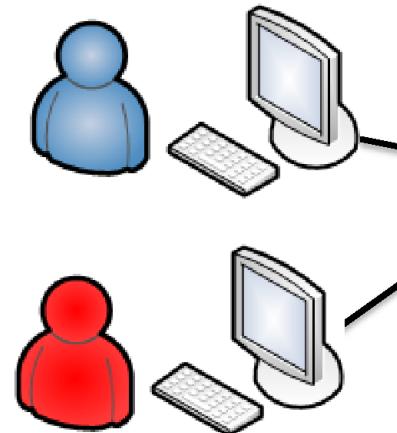
Serveur

d'authentification

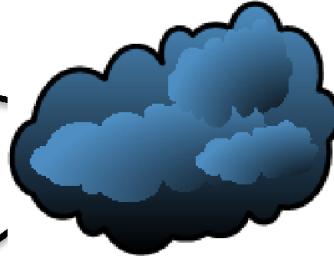
Serveur d'authentification s'occupe juste de l'authentification des utilisateurs avant connection au serveur web
protocole kerberos, protocole LDAP

Architecture des applications web

Client légitime



Internet



HTTP
HTML

Server Web



XML



SQL

Server BD



Client malveillant

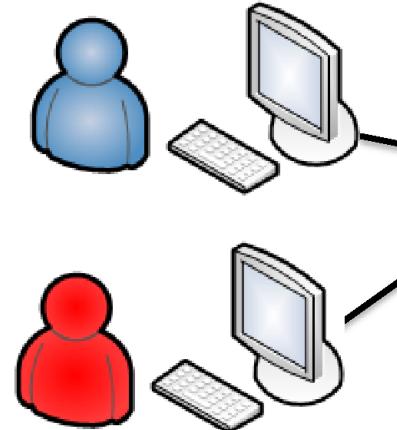
Kerberos
LDAP



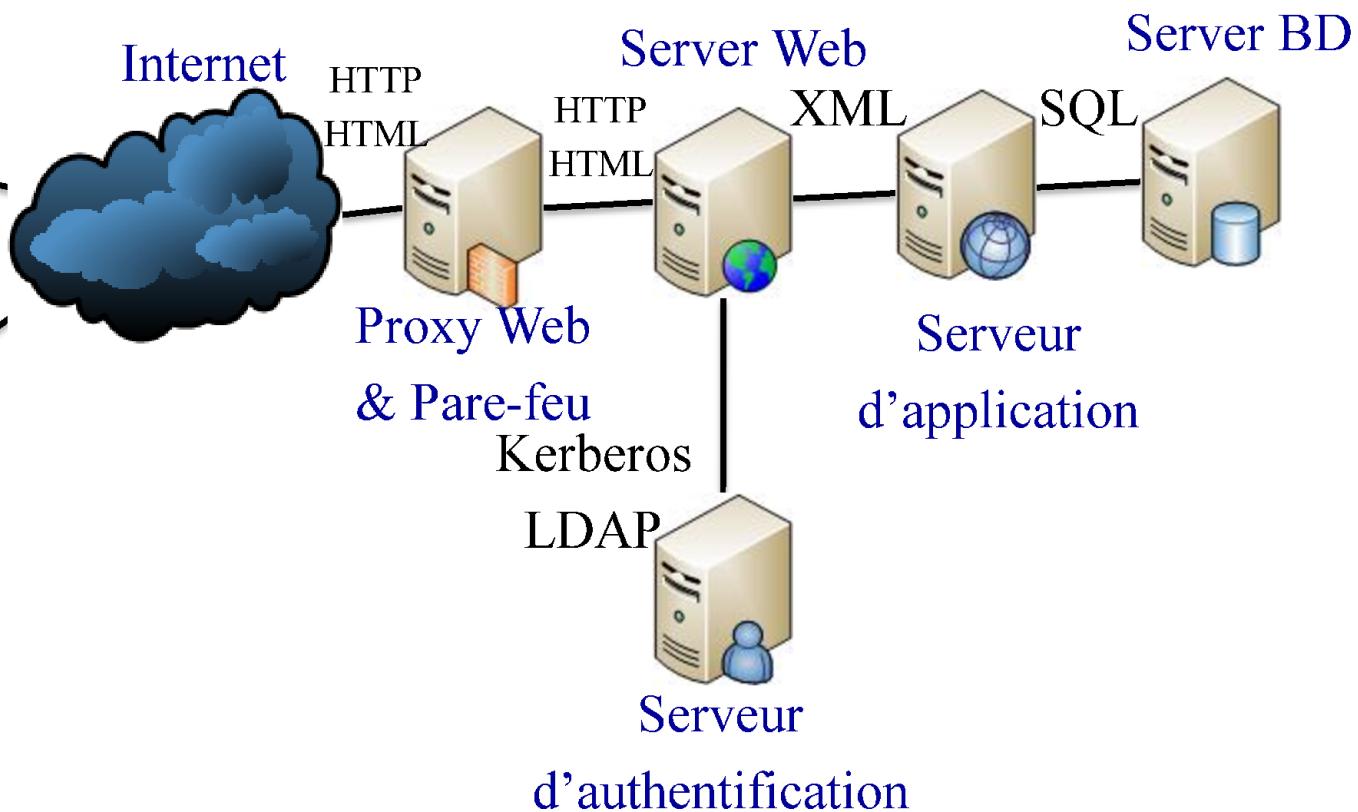
Serveur
d'authentification

Architecture des applications web

Client légitime



Client malveillant

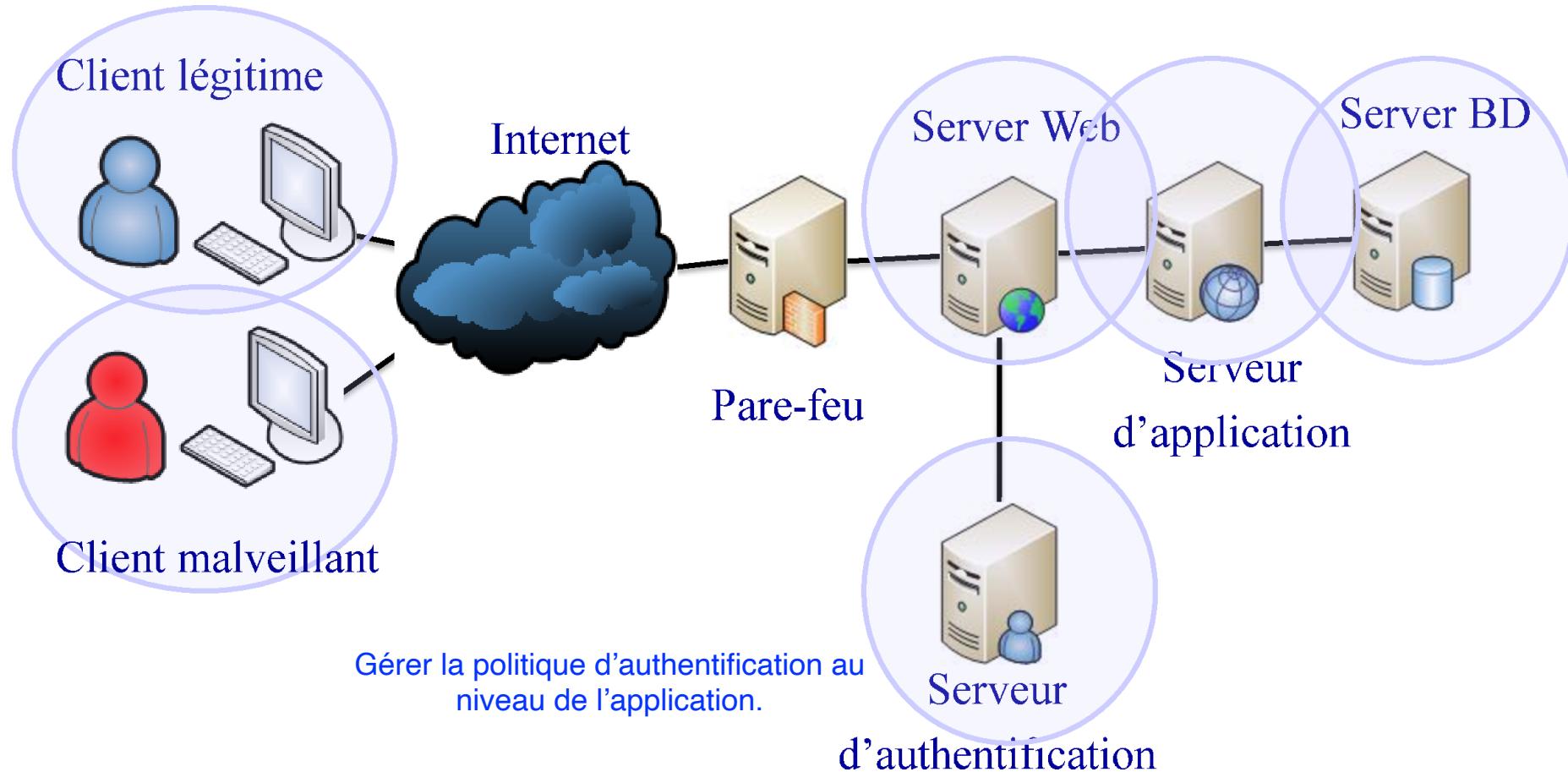




Authentification

- Composantes impliquées

authentification côté client pas bon
attaque sur client pour voler ses
données d'authentification



Authentification

- Canal de communication sécurisé ([https](https://www.polytechnique.mtl.ca))
 - Repose sur SSL-TLS (au-dessus de la couche 4)
- Authentifier le serveur
 - Certificat X509
 - Image personnalisée

HTTPS repose sur SSL-TLS

serveur a un certificat X509 délivré par une autorité de certification

Quand client veut se connecter le serveur commence par présenter son certificat au client

Serveur peut aussi présenter une image paramétrable en fonction de l'utilisateur
- Authentifier le client
 - Certificat X509 sur le poste client (optionnel)
 - Nom d'utilisateur + Mot de passe
 - Authentification deux facteurs

Suite à établissement d'une connection https

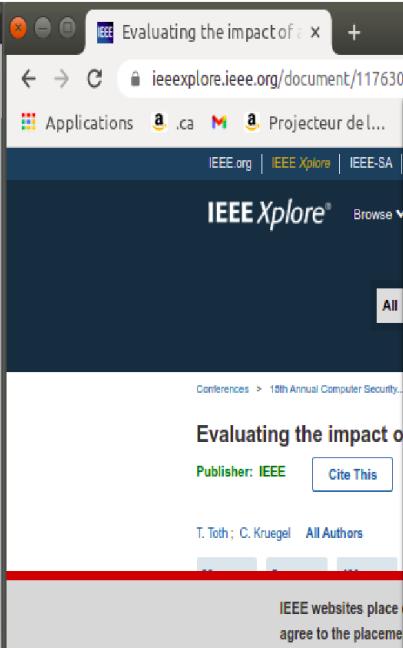
client va s'authentifier avec username et password

ou avec 2 facteurs pour plus sécurité
- Activation des priviléges
 - En fonction du profil d'authentification
 - Politique d'autorisation (ou de contrôle d'accès)

authentification permet activation des priviléges

Authentification

- Authentification du serveur
 - Certificat SSL



Gmail: Email from Google - Windows Internet Explorer
 https://www.google.com/accounts/ServiceLogin?service=mail

File Edit View Favorites Tools

Website Identification

VeriSign Class 3 Public Primary CA has identified this site as:
 www.google.com

This connection to the server is encrypted.

Lecteur du certificat : *.ieee.org

Général Détails

Ce certificat a été vérifié pour les utilisations suivantes :

Certificat du serveur SSL

Émis pour

Nom commun (CN)	*.ieee.org
Organisation (O)	THE INSTITUTE OF ELECTRICAL AND ELECTRONIC ENGINEERS, INC
Unité d'organisation (OU)	<Ne fait pas partie du certificat>

Émis par

Nom commun (CN)	DigiCert SHA2 Secure Server CA
Organisation (O)	DigiCert Inc
Unité d'organisation (OU)	<Ne fait pas partie du certificat>

IEEE

autorité de certification inclut des informations sur l'entité ou l'organisation pour laquelle le certificat a été émis, telles que son nom et son adresse.

La signature numérique valide le champ CN est la même que le nom du site courant visité



Authentification

- Authentification du serveur
 - Image personnalisée (par cookie persistant lié au navigateur)

image personnalisé généré aléatoirement par le serveur sauvegardé sur l'appareil du client en tant que cookie

Sign in to Yahoo!

Are you protected?
Create your sign-in seal.
[\(Why?\)](#)

Yahoo! ID:

Password:

Keep me signed in
for 2 weeks unless I sign out. [New!](#)
[Uncheck if on a shared computer]

[Forget your ID or password?](#) | [Help](#)

Don't have a Yahoo! ID?
Signing up is easy.

[Sign Up](#)

Sign in to Yahoo!



Yahoo! ID:

Password:

Keep me signed in
for 2 weeks unless I sign out. [New!](#)
[Uncheck if on a shared computer]

[Forget your ID or password?](#) | [Help](#)

Don't have a Yahoo! ID?
Signing up is easy.

[Sign Up](#)

protection contre attaque phishing où client entre vrai info sur faux site et attaquant utilise plus tard

protection contre man in the middle si paquets interceptées et hacker envoie paquet en tant que user image n'est pas présent



Authentification

- Comment renforcer l'authentification client-serveur ?
 - Voir cours sur l'authentification
 - Notamment... renforcer avec authentification à 2 facteurs
- Challenge – response
 - CHAP (Challenge-Handshake Authentication Protocol)
 - Kerberos
- Réauthentification à intervalles réguliers

redemander de s'authentifier à un intervalle régulier

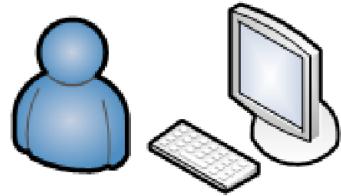
cookie peut être paramétré à authentifier à chaque ... ou
jamais
cookie n'est pas un mode d'authentification !!



SQL injection

- Injection SQL (SQL Injection)

Client légitime



Username:

Password:

Remember me on this computer.

cas authentification confié à un serveur de base de données
donc serveur à un accès direct à la base de données

Server BD



```
extract($_POST);
```

```
$req = "select mem_code from MEMBRES  
       where mem_login = '$login'  
       and mem_pwd = '$pass'";
```

```
$result = mysql_query($req) or  
die ("Error : the SQL request <br><br>".$req."<br><br> is not valid: ".mysql_error());  
list($mem_code) = mysql_fetch_array($result);  
if (empty($mem_code)) { //vérifier que la requête a retourné une réponse positive}
```

Server Web

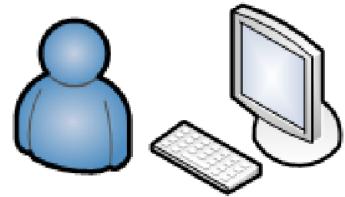




SQL injection

- Injection SQL (SQL Injection)

Client légitime



Username: **daniel**
Password: **Xa4!dfga**
 Remember me on this computer.
Sign in

Server BD



```
select mem_code from MEMBRES  
where mem_login = 'daniel'  
and mem_pwd = 'Xa4!dfga'
```

```
extract($_POST);
```

```
$req = "select mem_code from MEMBRES  
       where mem_login = '$login'  
       and mem_pwd = '$pass'";
```

```
$result = mysql_query($req) or  
die ("Error : the SQL request <br><br>".$req."<br><br> is not valid: ".mysql_error());  
list($mem_code) = mysql_fetch_array($result);  
if (empty($mem_code)) { // vérifier que la requête a retourné une réponse positive}
```

Server Web

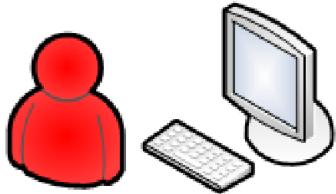


SQL injection

- Injection SQL (SQL Injection)

guillemet pour s'assurer que le format de la requête soit bon

Server BD



Client malveillant

Username:	daniel
Password:	' or '1'='1
<input type="checkbox"/> Remember me on this computer.	
Sign in	



```
select mem_code from MEMBRES
where mem_login = 'daniel'
and mem_pwd = '' or '1'='1'
or 1=1 donne toujours vrai
```

```
extract($_POST);
```

```
$req = "select mem_code from MEMBRES
       where mem_login = '$login'
       and mem_pwd = '$pass'";
```

```
$result = mysql_query($req) or die ("Error : the SQL request <br><br>".$req."<br><br> is not valid: ".mysql_error());
list($mem_code) = mysql_fetch_array($result); prend seulement la première variable dans tous
if (empty($mem_code)) { // vérifier que la requête a retourné une réponse positive}
```

va retourner tous les mem_code de la base de données

Server Web



utilisateur va être authentifié et en tant qu'administrateur (premier user dans table)



SQL injection

- Injection SQL (SQL Injection)



ferme la première requête

```
x'; INSERT INTO members
('email','passwd','login_id','full_name') VALUES
('steve@unixwiz.net','hello','steve','Steve Friedl');--
```

insert un faux membre qui peut le permettre de revenir dans la base de données



SQL injection

- Injection SQL (SQL Injection)



`x' ; exec(char(0x73687574646f776e))' ;`

commande qui arrête les mécanismes de détection de bd

`x' ; shutdown ;`

difficile de filtrer entrées utilisateurs

peut faire des choses pour éviter de se faire détecter

Remarque importante :

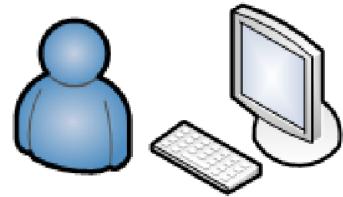
- La plupart des SGBD permettent d'exécuter des commandes shell depuis le SGBD
- Exemple Microsoft SQL Server)
- On peut en faire autant avec une SQL injection qu'avec un shell code !



Cross site scripting

- Cross site scripting (XSS) non persistant

Client légitime



Gagner de l'argent

Search

Server Web



```
extract($_POST);  
  
$req = "select * from POSTS  
       where title = '$stitle'"
```

Search results for Gagner de l'argent:

- Comment gagner de l'argent facile et des cadeaux sur internet...
- L' objectif du blog est de présenter toutes les idées qui permettent d' économiser ...

requete au serveur bd

bd revoie contenu

renvoie au client

```
<html>  
<head></head>  
<body>  
  
<h1>Search results for Gagner de l'argent :</h1>  
<itemize>  
    <item>Comment gagner de l'argent facile et  
des cadeaux sur internet...</item>  
    <item>L'objectif du blog est de présenter  
toutes les idées qui permettent d'économiser ...</item>  
</itemize>  
</body>  
</html>
```



Cross site scripting

- Cross site scripting (XSS) non persistent



script n'est pas sauvegarder, lien avec script en arrière envoyer à un utilisateur et quand l'utilisateur click dessus le script est lancé pour voler le nom utilisateur, le mot de passe, les cookies de sessions

```
<html>
<head></head>
<body>

<h1>Search results for <u>Super</u> :</h1>
No results found
</itemize>
</body>
</html>
```

```
extract($_POST);

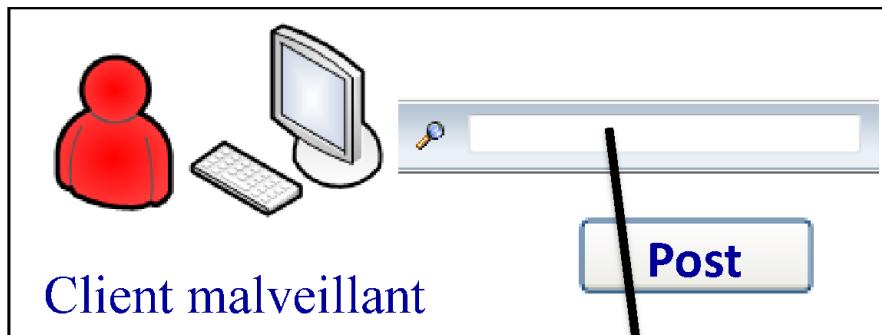
$req = "select * from POSTS
       where title = '$stitle'
```

code javascript en entré peut conduire à attaque contre serveur



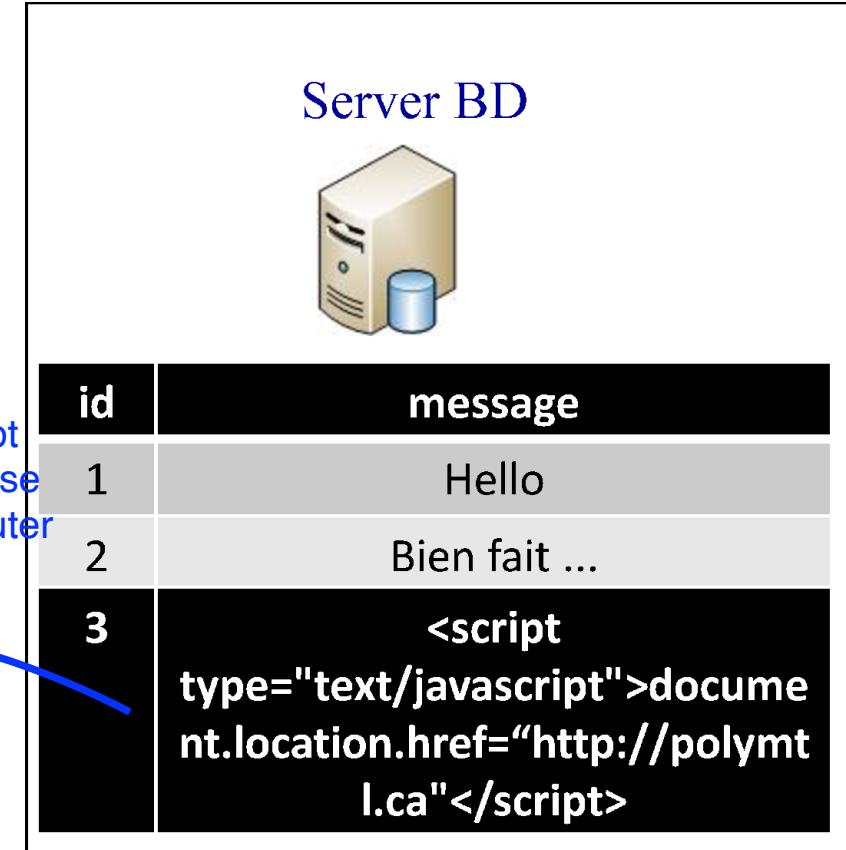
Cross site scripting

- Cross site scripting (XSS) persistent



L'attaquant va utiliser l'application et sauvegarder le script à chaque fois qu'une victime accède les données de la base de données où le script est sauvegardé, le script va s'exécuter

redirection vers le site du hacker



```
<script type="text/javascript">document.location.href="http://polymtl.ca"</script>
```

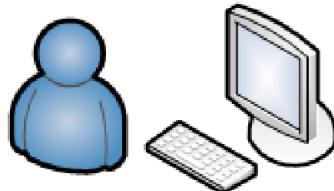
Your message has been posted



Cross site scripting

- Cross site scripting (XSS)

Client légitime



Guestbook messages:

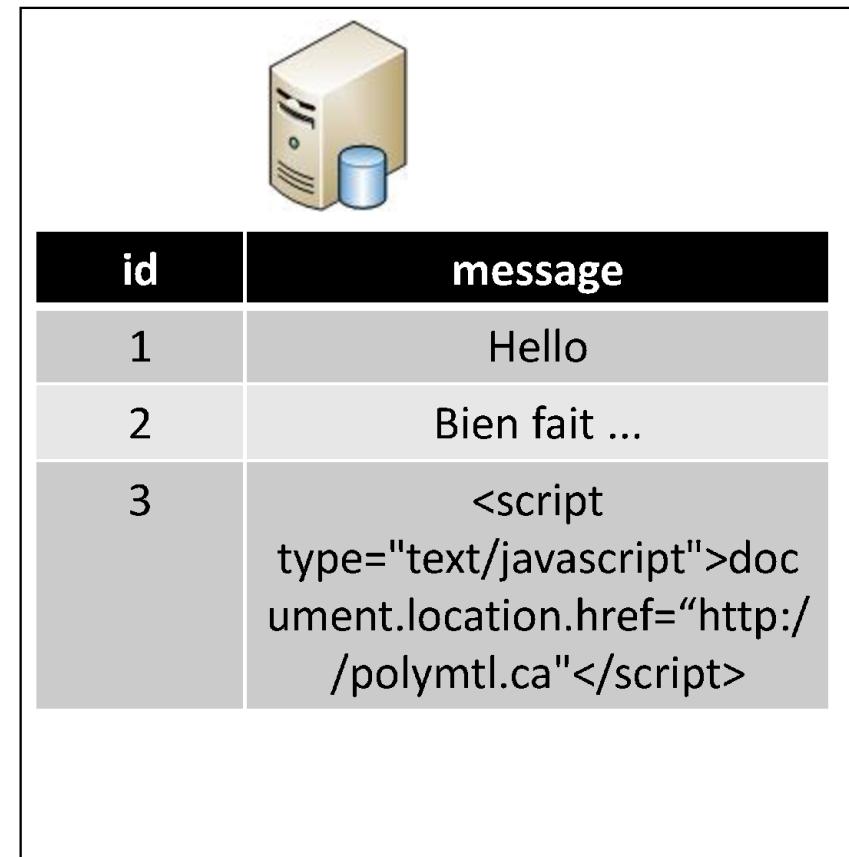
Hello

Bien fait ...



```
<h1>Guestbook messages:</h1>
Hello<br>
Bien fait<br>
<script
type="text/javascript">document.locatio
n.href="http://polymtl.ca"</script><br>
...
...
```

Server BD

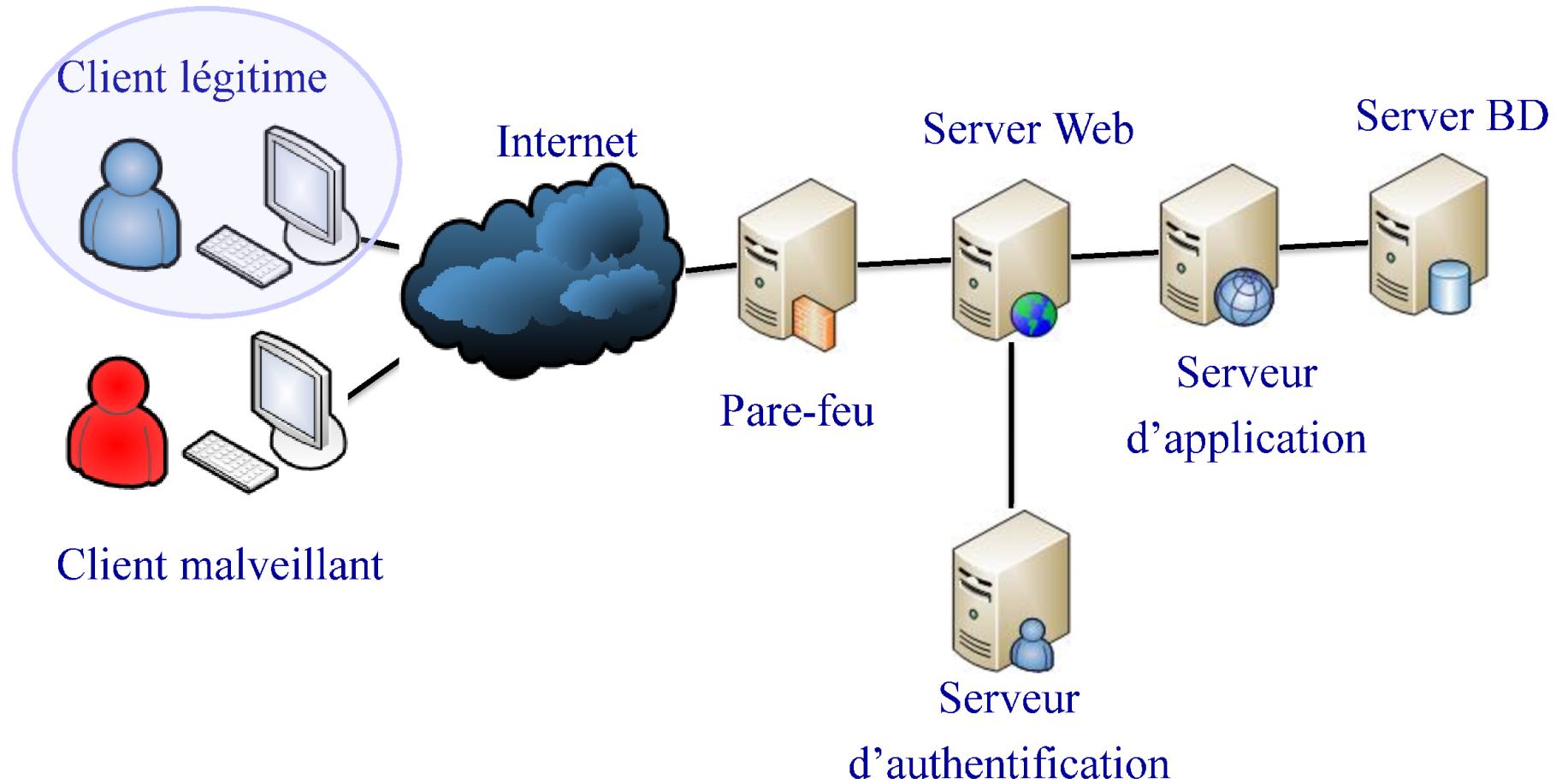




- Quel est le but de l'attaquant dans une attaque XSS ?
 - Rediriger le trafic du client vers le site de l'attaquant
- Pourquoi ?
 - Améliorer le référencement du site de l'attaquant
 - Faire de l'argent *si publicité paye site plus il y a utilisateurs plus d'agents fait*
 - L'attaquant se fait de l'argent en faisant cliquer le client sur son site
 - Infecter le site client
 - Exploitation d'une vulnérabilité du navigateur du client
 - Souvent, attaque par buffer overflow
- Remarque
 - Le XSS n'a pas de réel impact sur le serveur attaqué
 - Le serveur sert seulement à relayer le client vers le site de l'attaquant

Vérification des données usager (Input validation)

- Ce qu'on fait





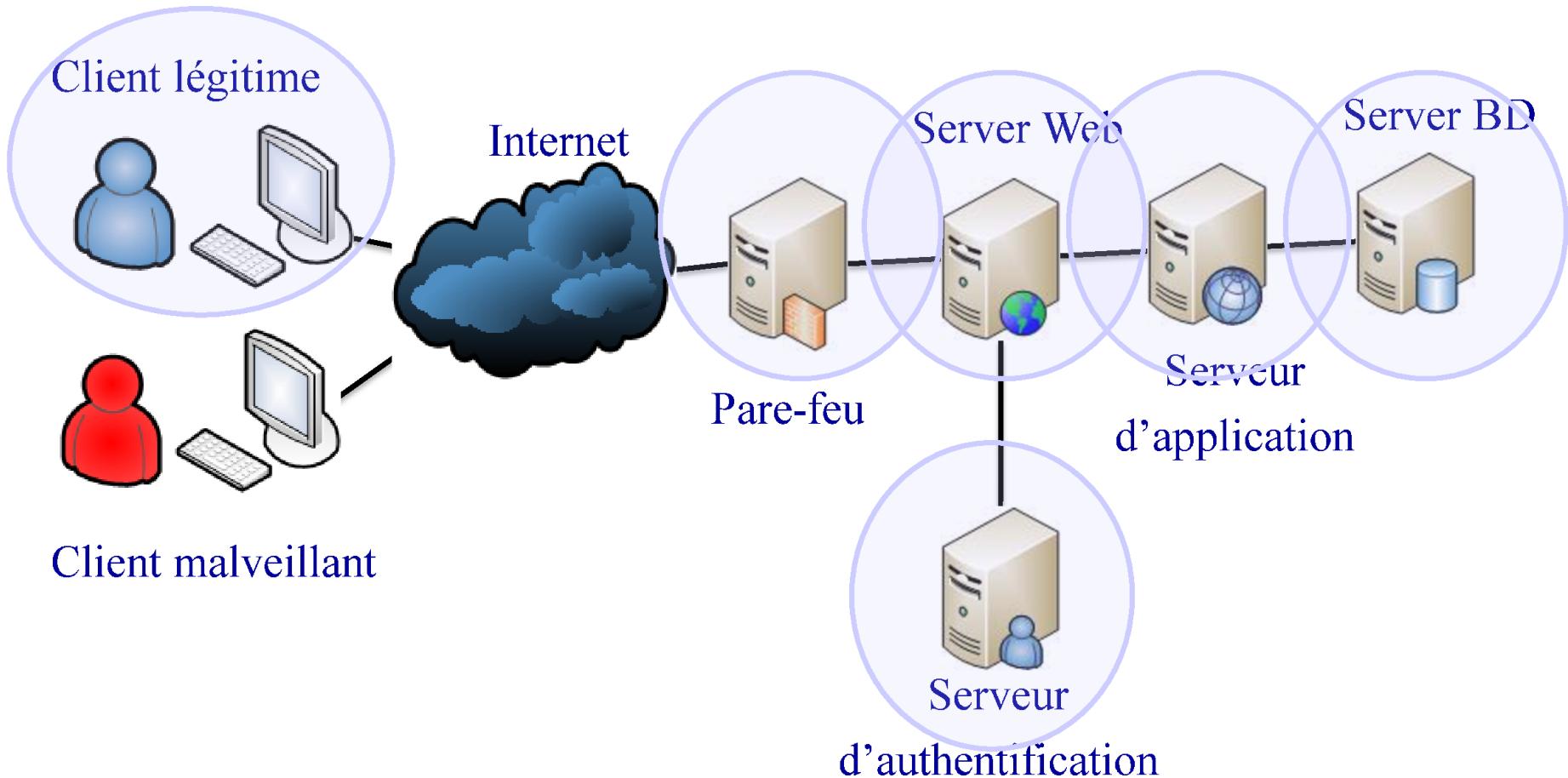
Vérification des données usager (Input validation)

- Code source html

```
<form action="mailto:yourname@yourdomain.com" method="post" onsubmit="return  
checkform(this);">  
  
<script language="JavaScript" type="text/javascript">  
<!--  
function checkform ( form )  
{  
    // see http://www.thesitewizard.com/archive/validation.shtml  
    // for an explanation of this script and how to use it on your  
    // own website  
  
    // ** START **  
    if (form.email.value == "") {  
        alert( "Please enter your email address." );  
        form.email.focus();  
        return false ;  
    }  
    // ** END **  
    return true ;  
}  
//-->  
</script>
```

Vérification des données usager (Input validation)

- Ce qu'on devrait faire





Vérification des données usager (Input validation)

- Valider les données de l'usager
- Où ?
 - sur le serveur Web
 - et/ou sur le serveur d'applications
- Comment ?
 - Exact Match (exemple : seulement « true » et « false » permis)
 - Whitelisting (exemple : seulement (a-zA-Z)+ permis)
 - Blacklisting (exemple: « SELECT » « JOINT » pas permis)
 - Encoding (exemple : mysqli_real_escape_string)
- Quoi d'autre ?
 - Limiter la taille de l'entrée

Vérification des données usager (Input validation)

- Utiliser les SQL Stored Procedures requêtes déjà fait qui prend des params
- Gérer les permissions sur la base de données
 - usagers, rôles, permissions control d'accès basé sur rôle donc qui a droit à quoi comme contenu
ex: bloquer quand injection sql veut ramener table de tous les membres
- Messages d'erreur
- Pare-feu applicatif
 - Software
 - ModSecurity
 - Appliance
 - Cisco, Fortinet, Checkpoint, etc.

Control de flux: pendant qu'un utilisateur consulte une table, empêche qu'il va consulter une autre.
contrôle de type MAC: faire une liste d'adresse MAC qui ont le droit et bloquer les autres

- Comment vérifier si un site est vulnérable ?
- Rien fait pour se protéger -> probablement vulnérable
- Développé sans gestion de projet -> probablement vulnérable
- Outils de scan automatique
 - Nikto
 - Acunetix (\$\$\$\$ mais gratuit pour test de XSS)
 - WebScarab
 - Autres (<http://sectools.org/web-scanners.html>)

rien mis en place => vulnérable

outil pour scanner application web pour trouver automatiquement les vulnérabilités



Vérification des données usager (Input validation)

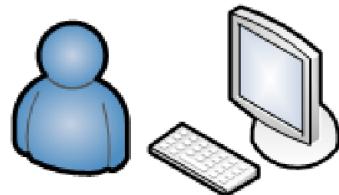
- Autres types d'attaques contre les applications Web
 - Cross Site Request Forgery (XSRT)
 - Remote File Inclusion
 - Variable tampering
 - Interface redressing (clickjacking)



Cross-Site Request Forgery

- Utilisation normale

Client légitime



CSRF

hypothèse: session légitime entre client authentifié et server existe

GET index.php

Server Web



Username:
Password:
 Remember me on this computer.

www.exemple.com

demande au client d'exécuter une action sur le serveur

POST login.php

hacker envoie une requête HTTP falsifiée au client comme si elle venait du vrai client. Il a l'air d'une source apparente. Le serveur va exécuter.

Redirect index.php

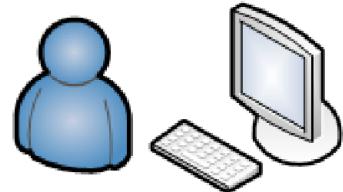
Set Cookie: PHP_SESSID=vxae9pa6nw408967a123...

après authentification, cookie renvoyé avec session id

Cross-Site Request Forgery

- Utilisation normale (2/2)

Client légitime



Server Web



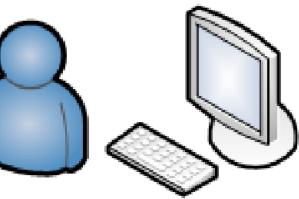
GET =index.php?action=acheter&id=1
Cookie PHP_SESSID=vwae9pa6nw408967a123...



Cross-Site Request Forgery

- Attaque

Client légitime



GET malicious.asp

page avec code malveillant

Server Web



www.attaque.com

**GET =index.php?action=acheter&id=1
Cookie PHP_SESSID=vwaegpa6nw408967a123...;**

Server Web



www.exemple.com

changer user et pass pour que hacker sign in et voit info sensible de compte de victim

lien qui amène la victime à un faux site et le code va auto exécuter sur le navigateur en envoyant l'imitation d'une requête à la victime qui va envoyer une vrai requête au vrai site pour exécuter des actions faire ce qu'il veut ne prétendant être l'usager



Protection contre le Cross-Site Request Forgery

- Comment se protéger ?

comme un one time pad attaché sur le form
envoyer au moment du login
stocké dans les variables de session du
serveur

- Token aléatoire

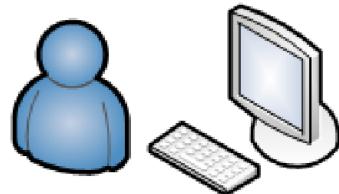
- Envoyé au moment du login
- Stocké dans les variables de session du coté serveur
- Ne pas stocké dans un cookie du coté client, mais
- Présent dans les liens de toutes les autres pages
- Vérifié par le serveur pour chaque page

Quand form est renvoyé du client au serveur,
serveur vérifie si la form a le token et que ce n'est pas une
form forgé par un hacker

Protection contre le Cross-Site Request Forgery

- Utilisation normale

Client légitime



GET index.php

Server Web



www.exemple.com



Username:

Password:

Remember me on this computer.

POST login.php

token généré

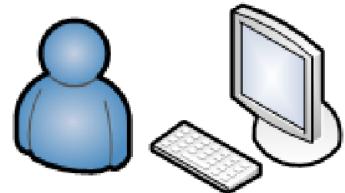
Redirect index.php?secureToken=431fwap8rawddf...
Set Cookie: PHP_SESSID=vxae9pa6nw408967a123...



Protection contre le Cross-Site Request Forgery

- Utilisation normale

Client légitime



Server Web



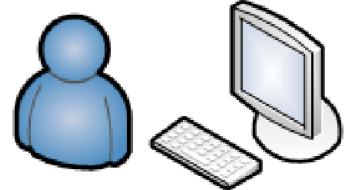
vérifie chaque fois que la form est bien celui que le serveur a renvoyé au client

GET =index.php?action=acheter&id=1&secureToken=431fwap8rawddf...
Cookie PHP_SESSID=vwae9pa6nw408967a123...

Protection contre le Cross-Site Request Forgery

- Attaque

Client légitime



GET malicious.asp

Server Web



www.attaque.com

```
<html>
<head></head>
<body>

</body>
</html>
```

**GET =index.php?action=acheter&id=1
Cookie PHP_SESSID=vwaegpa6nw408967a123...
www.attaque.com**

Server \



www.exemple.com

**Vérification de la variable
secureToken échouée.
Session fermée.**



Hameçonnage (Phishing)





Hameçonnage (Phishing)

- Comment se protéger ?
- Filtrer le spam
- Authentification du serveur
- Eduquer les utilisateurs

protection contre phishing
filtrer les faux messages
demander au serveur de fournir son certificat et
vérifier que le certificat est valide

Moralité de l'histoire

- Chaque attaque est différente
- Exploite la logique de l'application
- Difficile (impossible) à détecter par des outils automatiques
- Code review
- Exemples
 - Faire un don de -100\$
 - Créer un million d'usagers et écrire des messages dans un forum
 - Enlever le câble réseau au milieu d'une partie d'échec

- Attaques web très populaires
- Facile de créer une application vulnérable
- Validation des données usager
- Éducation des usagers
- Principe de sécurité de l'oignon (layered security)
- OWASP (Open Web Application Security Project) Top 10
 - http://www.owasp.org/index.php/Top_10_2007
- Clickjacking
 - <http://ha.ckers.org/blog/20081007/clickjacking-details/>
- SQL Injection Cheat Sheet
 - <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
- XSS Cheat Sheet
 - <http://ha.ckers.org/xss.html>

couche de protection

proxy —> couche authentification —> validation des entrées par le serveur —> politique d'autorisation

— <http://ha.ckers.org/xss.html>