Sécurité des sites WEB

Philippe Mathieu & Guillaume Dufrene

IUT-A Lille
http://www.iut-a.univ-lille.fr
prenom.nom@univ-lille.fr

Le problème

Université de Lille

- De nombreux sites WEB permettent à l'utilisateur de publier de l'information
  - les blogs
  - les réseaux sociaux
  - les sites contenant des informations personnelles

  Il faut s'assurer qu'un utilisateur malveillant ne puisse empêcher le fonctionnement normal du site en saisissant des données
- De nombreux sites WEB necessitent de s'authentifier

  Il faut s'assurer qu'un utilisateur malveillant ne puisse se connecter sans y être autorisé

L'injection XSS

Université de Lille

- Technique d'injection de code utilisée pour "corrompre" un site web et empêcher son fonctionnement normal
- Envoyer du HTML ou du Javascript dans les champs de saisie pour changer l'affichage
- Injection interprétée coté Client

DANGER

A chaque fois que des données issues de l'utilisateur sont affichées

Interprétation d'une page HTML

Université de Lille

- HTML n'est jamais que du texte "interprété" par le navigateur
- Certains caractères ont une signification particulière en HTML, avec une incidence forte sur le formatage de la page
- Notamment : <, >, " et &
- Taper `<h1>` dans une page HTML ne donne pas `<h1>` à l'affichage !
- Pour ces caractères spéciaux, HTML préconise un encodage spécifique

Car	Code Iso UTF-8	Code HTML
"	&#34 ;	&quot ;
&	&#38 ;	&amp ;
<	&#8249 ;	&lsaquo ;
>	&#8250 ;	&rsaquo ;

Perturber l'affichage

Université de Lille

- Quand on passe en paramètre d'une requete HTTP une chaine destinée à être affichée, elle sera interprétée lors de l'affichage !
- `<h1>Hello</h1>` donnera **Hello**
- `\&lsaquo;h1&rsaquo;Hello&lsaquo;/h1&rsaquo;` donnera `<h1>Hello</h1>`
- Lorsque qu'un formulaire offre la possibilité de saisir une chaine qui sera par la suite affichée dans une page il est donc impératif d'encoder les caractères spéciaux pour éviter une erreur d'affichage dans le navigateur.

XSS par injection d'HTML

Université de Lille

Le principe de l'injection HTML consiste à saisir dans des champs de formulaires des chaines de caractères dont l'interprétation perturbe l'affichage normal

Il saisit :

- `<h1>HELLO WORLD</h1>`
- `<font size=7 color=red>HELLO WORLD</font>`
- `<form>...</form>`

XSS par injection de Javascript

Université de Lille

- `<script>alert('hack')</script>`
- `<script>location='http://www.libe.fr'</script>`
- `<script>window.open("monscript.php")</script>`
- `<script>alert(document.cookie)</script>`

Pour éviter l'injection XSS

Université de Lille

Traduire les `<h1>Hello</h1>` en
`\&lsaquo;h1&rsaquo;Hello&lsaquo;/h1&rsaquo;`

- Au minimum : `replaceAll("[\<.*?>]","");` sur toutes les chaines saisies
  inconvénient : enlève des caractères parfois nécessaires
- Idealement : API Apache `commons-text` avec la classe `StringEscapeUtils` et sa méthode `escapeHtml4(String)`
  Avantage : conserve tous les caractères et les encode HTML

L'injection SQL

Université de Lille

- Technique d'injection de code utilisée pour "attaquer" la BDD associée à un site web
- Envoyer du SQL dans les champs de saisie pour détourner les requêtes SQL
- Injection interprétée coté Serveur

Danger

A chaque fois qu'une requête est exécutée à partir de données saisies par l'utilisateur
(notamment quand des requêtes sont construites par concaténation)

Exemple 1 : selection d'un nom valide

Université de Lille

- Si la requête dans les pages du site est

```
query = "select * from users "+
        "where login ='" + nom + "';" ;
```

- et que l'utilisateur entre `' or '1'='1`
- On exécutera

```
select * from users where login ='' or '1'='1';
```

- Requête qui donne toujours un succès et selectionne un login (en général le 1er de la table, souvent l'admin !)

Exemple 2 : injection de commentaires

Université de Lille

- Si la requête dans les pages du site est

```
query = "select * from users "+
        "where login ='" + nom + "' and mdp =...;" ;
```

- et que l'utilisateur entre `' or '1'='1' --` ou autre commentaire
- On exécutera

```
select * from users  where login ='' or '1'='1' -- ;
```

- La requête est toujours vraie, la fin est invalidée !

Exemple 3

Université de Lille

Détruire des données

- Si la requête dans les pages du site est

```
query = "select * from users "+
        "where phone ='" + tel + "';" ;
```

- et que l'utilisateur entre

```
a'; drop table commandes;
```

- On exécutera

```
select * from users where login ='a' ;
drop table commandes;
```

- La table `commandes` est alors détruite

Mauvaise solution

Université de Lille

Principe :

- Mettre des `\` devant les caractères spéciaux.
- la saisie de `' or '1'='1` devient `\' or \'1\'=\'1`
- Usage de `replaceAll` pour éviter certains mots

Une fausse "bonne solution" !

- Complexe à mettre en oeuvre (difficile de ne rien oublier)
- Insuffisant pour contrer l'ensemble des attaques

Bonne solution : Les requêtes pré-compilées

Université de Lille

Principe :

- On pré-compile dans le SGBD la requête sans ses paramètres
- L'arbre relationnel ne peut plus être modifié à l'exécution
- On utilise les "setters" pour fixer les paramètres

```
PreparedStatement ps =
  con.prepareStatement("SELECT * FROM users "+
                       " WHERE login=? and pwd=? ");
...
ps.setString(1,nom);
ps.setString(2,password);
...
ResultSet rs = ps.executeQuery();
```

Bien d'autres écueils subsistent

Université de Lille

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]	∪	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]	∪	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.]

issu de www.owasp.org

En résumé

Université de Lille

- Deux grandes techniques de piratage s'appuient sur de l'injection de code
  - L'injection XSS, coté client
    Perturbe l'affichage souhaité
  - L'injection SQL, coté serveur
    Permet d'attaquer la base de données
- De bonnes pratiques de programmation permettent de les éviter
  - Encoder tout paramètre en HTML pour XSS
  - Usage de requêtes précompilées pour SQL