Comprendre la sécurité

de Drupal et des applications web

Damien Tournoud

- CTO @ Commerce Guys ;
- Membre de l'équipe de sécurité de Drupal depuis 2008 ;
- Et deux trois autres trucs...

L'équipe de sécurité

- Créée en 2005 sous l'impulsion de Karoly Negashi (@chx).
- Coordonne les releases de sécurité:
  - de Drupal core (3 en 2013),
  - et des modules de contribution (98 en 2013)
- Assure veille, documentation et formation sur les sujets liés à la sécurité.

Pour les impatients

Règles de base pour les administrateurs

- Installez un minimum de modules, et choisissez les parmi les modules les plus populaires (évitez en particulier les modules n'ayant pas de version stable) ;
- Gardez vos modules à jour ;
- Portez attention aux roles et permissions ;
- Limitez au minimum les formats d'entrée ;
- Considerez les modules Security Review et Password Policy.

Règles de base pour les développeurs

Pour apprendre comment développer au
mieux vos modules Drupal, rendez vous sur
https://drupal.org/writing-secure-code

Trois types de vulnérabilités

- Contournement d'accès
- Détournement du navigateur
- Contournement de l'application

#1. Contournement d'accès

Un utilisateur peut faire ce qu'il ne **devrait pas** être autorisé à faire.

#2. Détournement du navigateur

Le plus évident: interception

Si la connection entre le navigateur et le serveur n'est pas chiffrée, elle peut être interceptée.

Depuis l'invention du Javascript (1995), les navigateurs sont devenus des machines virtuelles executant du code arbitraire.

HTML n'est pas un format de description de page, mais un conteneur de code.

(Prenez un peu de temps pour respirer.)

Le cross-site scripting (XSS)
L'injection de code

Si vous pouvez injecter du
contenu arbitraire dans une page,
c'est game over.

Il s'agit de la vulnérabilité la plus fréquente parce qu'elle *très* facile à introduire.

Un exemple minimal

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My title</title>
  </head>
  <body>
    <p class="test">Bonjour, Damien</p>
  </body>
</html>
```

Règle de la même origine

Chaque site ou "origine" (protocole, host, port) est isolé :

- Pas d'accès au contenu (DOM)
- Pas de requête HTTP

Exclus de la règle

- `<img>`, `<script>`, `<link>`, `<video>`, etc. ;
- Formulaires (`<form>`) ;
- Plugins tiers (Flash, Acrobat, Active X, ...).

Cross-site request forgery (CSRF)

Ou l'exploitation des limites de la règle de la même origine.

CSRF via GET

```
<img src="http://bank.example.com/withdraw?amount=1000000"/>
```

CSRF via <form>

```
<form method="post" action="http://bank.example.com/withdraw">
  <input type="hidden" name="amount" value="1000000" />
</form>
<script>$("form").submit()</script>
```

Le problème particulier des fichiers uploadés

#3. Contournement de l'application

Accès non sécurisé au serveur

- FTP (oui, en 2014)
- Mots de passe faibles
- ...

Injection de code

- Injections SQL
- Injection de code scripté (PHP)

Mais aussi…

- Sauvegardes de base de données
- Les problèmes de l'entête `Host`
- Les open-redirects

En résumé

- Un domaine constamment en mouvement ;
- Quelques règles simples couvrent 99% des problèmes.

Merci.