JBoss / WildFly
Sécurisation des applications

Sécurisation d'une application Web

- Pour cet exemple nous allons utiliser
  - le login-module de base du fichier de configuration
    - donc le realm *ApplicationRealm*
  - une authentification de type BASIC
    - transmission de user:password sous forme Base64

Sécurisation d'une application Web

- Dans le fichier *web.xml* de l'application, il faut
  - mettre en place la contrainte de sécurité sur les ressources à protéger
  - définir la méthode d’authentification
  - déclarer les rôles

Sécurisation d'une application Web

- Mise en place de la contrainte de sécurité

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>administration</web-resource-name>
        <url-pattern>/admin/*</url-pattern>
        <http-method>GET</http-method>
        <http-method>POST</http-method>
    </web-resource-collection>
    <auth-constraint>
        <role-name>administrateur</role-name>
    </auth-constraint>
</security-constraint>
```

Sécurisation d'une application Web

- Définition de la méthode d'authentification

```
<login-config>
    <auth-method>BASIC</auth-method>
</login-config>
```

- Déclaration des rôles

```
<security-role>
    <role-name>administrateur</role-name>
</security-role>
```

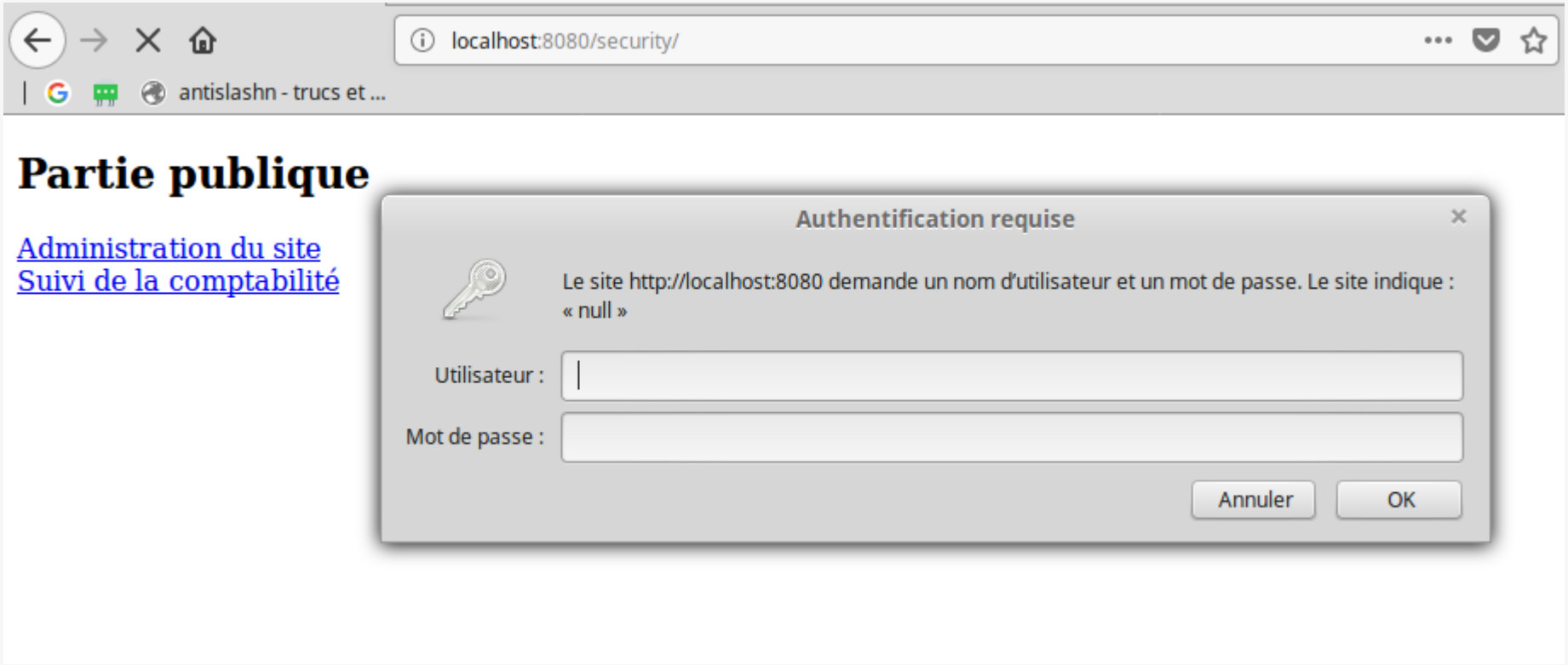
Sécurisation d'une application Web

- Ajouter un fichier *jboss-web.xml* dans le répertoire *WEB-INF* de l'application

```
<jboss-web>
    <security-domain>java:/jaas/other</security-domain>
</jboss-web>
```

- Côté serveur il faut ajouter un utilisateur
  - *./add-user.sh*
  - type d'utilisateur : Application User
  - gaston / gastonpw / administrateur

Sécurisation d'une application Web



Sécurisation d'une application Web

- Mécanismes d’authentifications
  - mécanismes HTTP
    - BASIC
    - DIGEST
    - CLIENT-CERT
      - nécessite HTTPS
  - mécanisme supplémentaire Java EE
    - FORM
      - le développeur fournit la page de login et d'erreur
      - le serveur envoie automatiquement les pages