CLUB DE LA SECURITE DE L'INFORMATION FRANCAIS

Revue de code Sécurité ou Test d'Intrusion Applicatif

Quel est le plus efficace pour évaluer un niveau de sécurité applicatif ?

http://www.google.fr/#q=sebastien gioria

- Responsable de la branche Audit S.I et Sécurité au sein du cabinet Groupe Y
- OWASP France Leader & Founder - Evangéliste
- OWASP Global Education Comittee Member (sebastien.gioria@owasp.org)
- Responsable du Groupe Sécurité des Applications Web au CLUSIF

- +13 ans d'expérience en Sécurité des Systèmes d'Information
- Différents postes de manager SSI dans la banque, l'assurance et les télécoms
- Expertise Technique
  - Sécurité Applicative (Revue de code, tests d'intrusion, ...)
  - Sécurité du cycle de développement (amélioration, mise en place, ...)
  - Gestion du risque, Architectures fonctionnelles, Audits
  - Consulting et Formation en Réseaux et Sécurité

Agenda

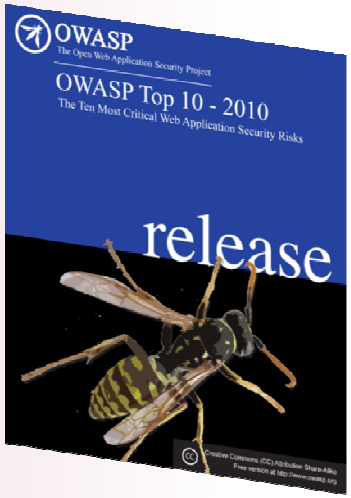
Introduction

Evaluer le niveau de sécurité d'une application ?

Efficience des différentes techniques sur des exemples

Conclusion

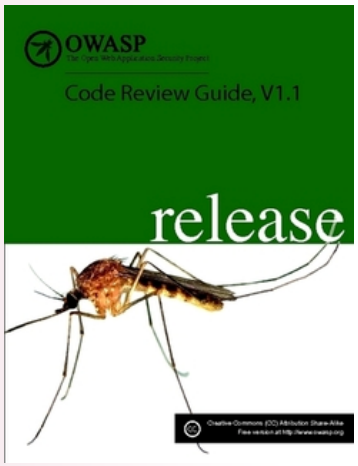
Apprendre



Contractualiser

Concevoir

Vérifier

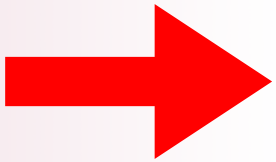


Tester

Améliorer

Pourquoi chercher des vulnérabilités ?

- ✓ Juste pour les trouver ?
- ✓ Pour savoir ou elles se trouvent exactement dans le code ?
- ✓ Pour s'assurer qu'elles ne sont pas dans notre application
- ✓ Pour se conformer à une exigence réglementaire ?



Quelle technique permet de répondre le mieux à l'une ou toutes ses questions ?

➡ Revue de code manuelle ?

➡ Test d'intrusion applicatif manuel ?

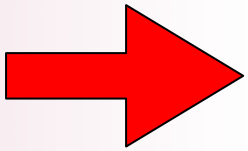
De quoi parle-t-on ?

Revue de code :

- Accès au code source
- Accès à la documentation fonctionnelle
- Accès à la configuration

Test d'intrusion applicatif :

- Accès via le réseau à l'application (protégée ou non par des éléments d'infrastructure)
- Temps limité
- Compétence du testeur limitée



L'utilisation d'outils permet d'aider la réalisation de la revue ou du test

Evaluer le niveau de sécurité d'une application ?

Référentiel :

- OWASP Top10 ?
  - L'un des plus connu, orienté Risques
- SANS Top25 ?
  - Plus orienté Code
- CWE ?
  - Un peu trop complexe ?
- OWASP ASVS ?
  - Plus orienté exigences fonctionnelles

Efficience des techniques ?

Efficience : Capacité d'un individu ou d'un système de travail d'obtenir de bonnes performances dans un type de tâche donné ; efficacité (définition du Larousse).

Métriques :

- Cout /Délais
- Faux Positifs
- Faux Négatifs
- Oublis

Echelle (de 1 à 5)

- 5 => simple
- 1 => difficile

Injection - Test

Injection - Code

```
if (numFisc == null)
{
    return false;
}
else
{
    query = "SELECT * FROM account WHERE numFisc = " + numFisc;
}
```

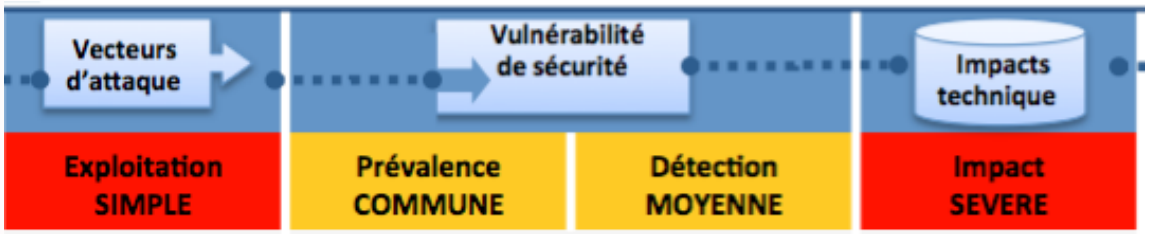
LDAP =>

```
ElementContainer ec = new ElementContainer();

searchlogin= "(&(uid="+user+")(userPassword={MD5}"+base64(pack("H*",md5(pass)))+"))";
```

```
try
{
    Connection connection = DatabaseUtilities.getConnection(s);

    ec.addElement(makeAccountLine(s));

    String query = "SELECT * FROM user_data WHERE last_name = '" + accountName +

    ec.addElement(new PRE(query));
```

Injection

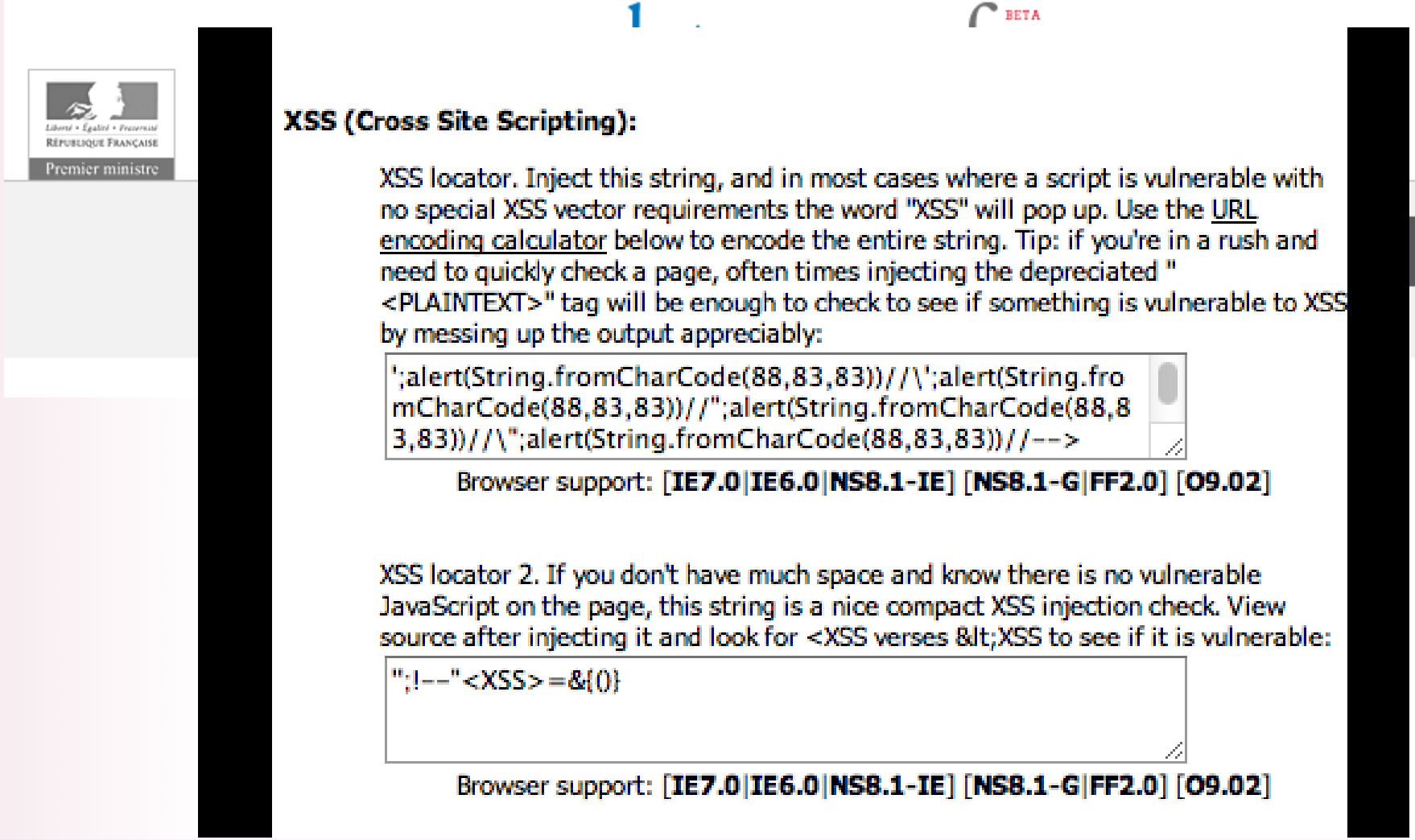
L'injection SQL fait beaucoup parler d'elle. Mais il existe d'autres formes d'injections : XML, XPath, LDAP, ORB(Hibernate), Commande



	Faux Positif	Faux Négatif	Oubli
Revue de code	1	1	1
Test	3	3	5

A voir : http://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

Cross Site Scripting (XSS) - Test



XSS (Cross Site Scripting):

XSS locator. Inject this string, and in most cases where a script is vulnerable with no special XSS vector requirements the word "XSS" will pop up. Use the URL encoding calculator below to encode the entire string. Tip: if you're in a rush and need to quickly check a page, often times injecting the depreciated "<PLAINTEXT>" tag will be enough to check to see if something is vulnerable to XSS by messing up the output appreciably:

```
';alert(String.fromCharCode(88,83,83))//\';alert(String.fro
mCharCode(88,83,83))//";alert(String.fromCharCode(88,8
3,83))//\";alert(String.fromCharCode(88,83,83))//-->
```

Browser support: [**IE7.0**|**IE6.0**|**NS8.1-IE**] [**NS8.1-G**|**FF2.0**] [**O9.02**]

XSS locator 2. If you don't have much space and know there is no vulnerable JavaScript on the page, this string is a nice compact XSS injection check. View source after injecting it and look for <XSS verses <XSS to see if it is vulnerable:

```
";!--"<XSS>=&{()}
```

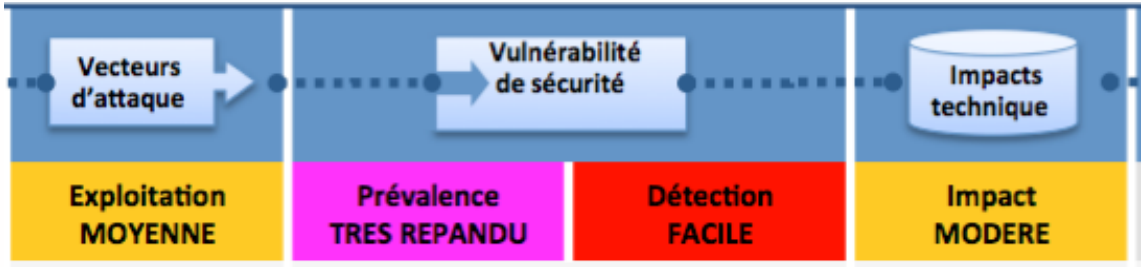
Browser support: [**IE7.0**|**IE6.0**|**NS8.1-IE**] [**NS8.1-G**|**FF2.0**] [**O9.02**]

Cross Site Scripting (XSS) - Revue

```
protected Element createContent(WebSession s)
{
    addMessage(s);

    ElementContainer ec = new ElementContainer();
    ec.addElement(makeInput(s));
    ec.addElement(new HR());
    ec.addElement(makeCurrent(s));
    ec.addElement(new HR());
    ec.addElement(makeList(s));

    return (ec);
}
```

Cross Site Scripting (XSS)

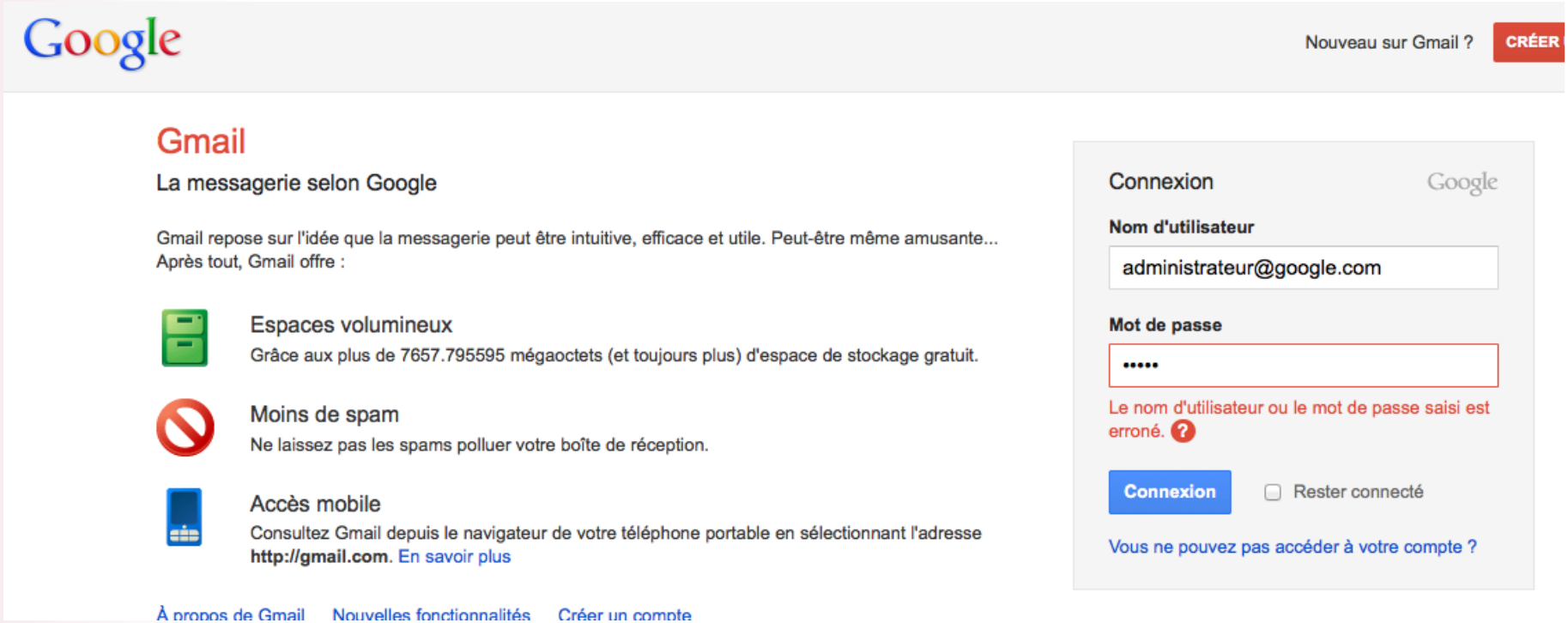
Le Cross Site Scripting est souvent mal considéré. Sa puissance peut aller jusqu'a la prise de contrôle sur le poste client...



	Faux Positif	Faux Négatif	Oubli
Revue de code	2	2	2
Test	5	3	1

A voir :
http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

Authentification



Authentification

Exigences	Test	Revue de code
Toutes les pages non publiques nécessitent un login		✓
Les mots de passes ne doivent pas être affichés dans la page Web	✓	
Les mots de passes sont créés de manière sécurisés	✓	
Les événements d'authentification sont loggués		✓
Les mots de passes sont stockés hashés		✓
Les sessions sont finies proprement lors du logout	✓	
Les IDs de session sont protégées par SSL	✓	
Les IDs de session ne sont jamais inclus dans l'URL	✓	
Les IDs de session sont aléatoires	?	?

Authentification

	Faux Positif	Faux Négatif	Oubli
Revue de code	3	3	?
Test	3	3	?

http://www.owasp.org/index.php/Authentication_Cheat_Sheet

Avantages

Test d'intrusion

✓Plus **facile** à effectuer
✓**Prouve** la vulnérabilité
✓Soumet **l'ensemble** de l'infrastructure au test
✓L'expertise nécessaire est **moins importante**

Revue de code

✓Permet d'avoir une vue **exhaustive**
✓Permet de découvrir **toutes les failles** d'une typologie
✓Vérifie que les **contrôles** sont **corrects**
✓Vérifie que **tous** les **contrôles** sont en **place**

Conclusion

Reprenons nos questions initiales :

✓Juste pour les trouver ?

➡ Aucun gagnant

✓Pour savoir ou elles se trouvent exactement dans le code ?

➡Avantage Revue de Code

✓Pour s'assurer qu'elles ne sont pas dans notre application

➡Avantage Revue de Code

✓Pour se conformer à une exigence réglementaire ?

➡Tout dépend de l'exigence :)

Conclusion

Découverte des vulnérabilités dans l'application en ligne

Découverte des vulnérabilités dans le code source