Quelles sont les bonnes pratiques de programmation dans la sécurité des applications web?

Table des matières

Présentation de l'auteur de ce document....3
Introduction....3
Organisation du document....3
I. Les failles de sécurité courantes dans le code....3
1) Cross-Site Scripting (XSS)....4
a) Stored XSS....4
Via un attribut HTML....4
Via AJAX....4
b) File upload XSS....4
c) Reflected XSS....4
2) Injections....4
a) Injections SQL....4
b) Injections de code....4
c) Injections dans le DOM (Document Object Model)....4
d) Injections XML....4
e) Injections JSON....4
3) Déni de service....4
4) Path traversal / Attaque par traversée de répertoires....4
5) Divulgation d'informations (information disclosure)....4
6) Altération de données (data tampering)....4
7) Obtention de privilèges (changer ce titre)....4
a) URLs non protégées....4
8) Attaque par cookies....5
9) Cross-site request forgery (XSRF)....5
10) Cross Site Script Inclusion (XSSI)....5
II. Les bonnes pratiques de programmation....5
1) Vérifier toutes les entrées....5
2) Séparer très clairement la présentation de la logique métier....5
3) D'autres à venir ….....5
III. L'utilisation des scanneurs de vulnérabilités....5
1) Test sur une application vulnérable....5
2) Résultats....6
3) Analyse....6
4) Conclusion....6
Conclusion....6
Annexes....6
Bibliographie....6

Quelles sont les bonnes pratiques de programmation dans la sécurité des applications web?

Présentation de l'auteur de ce document

Guillaume Humbert, étudiant en Master 2 MIAGE par apprentissage à l'université de Paris Ouest Nanterre La Défense. Je travaille en tant qu'apprenti analyse développeur depuis deux ans dans l'entreprise SNEF (Société Nouvelle Electric Flux) à Saint-Denis (93).

Introduction

De nos jours, les applications web sont de plus en plus utilisées. Elles s'appuient sur des technologies de plus en plus nombreuses: HTTP, HTML, JavaScript, XML, JSON, bases de données SQL, … Dues aux contraintes du marché et à la concurrence, le développement de telles applications doit se faire de plus en plus rapidement, et cela sans compromis sur la qualité. D'après mon expérience en entreprise et ce que j'ai pu apprendre au cours de ma formation, lorsque l'on parle de qualité, on parle surtout en termes de fonctionnels (est-ce que le logiciel marche?), en termes de performances, et en termes de gestion du projet (organisation des équipes, qualité du code, utilisation d'outils et de méthodes agiles, …). En revanche, et cela est surprenant, l'accent est peu mis sur l'aspect sécurité au niveau du code. Par exemple, cela ne choque personne de voir encore des concaténations de chaînes sans vérification pour exécuter des requêtes SQL.

L'enjeu est considérable. Le travail dans le secteur tertiaire est maintenant quasiment complètement dépendant des systèmes informatisés. Si les applications ne fonctionnent plus, plus personne ne peut travailler; si les données sont corrompues, de mauvaises décisions seront prises. Cela peut engendrer des pertes catastrophiques.

Il est donc primordial de s'imprégner de bonnes pratiques de programmation afin d'obtenir du code sécurisé en production.

Organisation du document

Ce document est organisé en trois grandes parties.

Dans la première, nous nous pencherons sur des erreurs courantes de programmation qui ouvrent des failles. Nous détaillerons ces erreurs avec des exemples de code en Java, et nous verrons comment les corriger.

Ensuite, nous predrons du recul sur la partie précédente. Nous verrons qu'il y a des problèmes qui se ressemblent, qui ne sont pas forcément liés à un langage ou une technologie particulière. A partir de ce point nous établirons une liste des bonnes pratiques à adopter dans la programmation d'une application web.

Dans un troisième temps, nous allons étudier les résultats d'une expérimentation. J'ai implémenté toutes les mauvaises pratiques énoncées précédemment dans une application web Java. Cette application sera soumise à des scanners de vulnérabilités. Nous tenterons de savoir quelle genre d'erreurs de programmation peuvent être détetées automatiquement, et nous nous poserons la question de l'utilité et des limites de ces outils.

I. Les failles de sécurité courantes dans le code

(texte d'introduction)

1) Cross-Site Scripting (XSS)

a) Stored XSS

Via un attribut HTML

Via AJAX

b) File upload XSS

c) Reflected XSS

2) Injections

a) Injections SQL

b) Injections de code

c) Injections dans le DOM (Document Object Model)

d) Injections XML

e) Injections JSON

3) Déni de service

4) Path traversal / Attaque par traversée de répertoires

5) Divulgation d'informations (information disclosure)

6) Altération de données (data tampering)

En uploadant des fichiers bizarres (un fichier qui s'appelle ../../toto.txt par exemple), on peut remplacer le vrai ../../toto.txt du serveur.

7) Obtention de privilèges (changer ce titre)

a) URLs non protégées

Il faut protéger les URLs du type /admin.jsp en ne laissant l'accès qu'aux administrateurs.

Quelles sont les bonnes pratiques de programmation dans la sécurité des applications web?

8) Attaque par cookies

Cookies: les données sont enregistrées côté client. Ils sont dont vulnérables, car ils peuvent être modifiés.

9) Cross-site request forgery (XSRF)

10) Cross Site Script Inclusion (XSSI)

II. Les bonnes pratiques de programmation

1) Vérifier toutes les entrées

Vérifier toutes les entrées:

- les saisies utilisateur
- les paramètres envoyés dans les requêtes HTTP, mêmes ceux censés être générés automatiquement par un navigeteur
- les données d'une base de données
- les données lues dans les fichiers

2) Séparer très clairement la présentation de la logique métier

Pas de vérification métier côté client (en JavaScript par exemple).

3) D'autres à venir …

III. L'utilisation des scanneurs de vulnérabilités

1) Test sur une application vulnérable

Présentation de l'application, outils utilisés, contexte, …

2) Résultats

3) Analyse

4) Conclusion

Conclusion

Annexes

Bibliographie