**Contexte**

Les applications Web occupant une place, de plus en plus importante au sein des organisations. La plupart des organisations dépendent d’elles en partant de la productivité de leurs employés au renforcement de leur domaine d’activités en passant par l’analyse des données qu’elles fournissent. De nos jours, les attaquants comprennent ô combien critiques ces applications sont, et tentent de plus en plus à les compromettre.

Cependant, il est possible de ces prévenir contre ces attaques mais les organisations n’arrivent pas très souvent à prendre en charge la sécurité de leurs applications ou n’en font pas une priorité. Cela est principalement dû aux raisons suivantes :

* La nécessité d’un développement et d’un déploiement rapide de ces applications
* La non-application de bonnes pratiques de sécurité lors du développement des applications
* Entre autres raisons

Ainsi, la plupart des applications développées aujourd’hui plusieurs failles de sécurité.

1. **SQL Injection**
   1. Definition

Une attaque par injection SQL cible les applications utilisant des systèmes de Gestion de base de données.

Lors d’une attaque par injection SQL, l’attaquant fournit un/des segments malicieux de code SQL en tant qu’input transformant la requête SQL initiale en une autre souvent dangereuse. Les attaques par injection SQL sont le plus souvent utilisées pour dérober des informations sensibles de la base de données.

Par une injection SQL, l’attaquant, selon ses capacités peut obtenir des informations confidentielles ou même arriver à modifier ces informations.

* 1. Causes

La cause principale d’une injection SQL est le fait qu’une application accepte des données provenant de sources non sures comme l’utilisateur par exemple (input) ou le navigateur de l’utilisateur (cookies ou headers des requêtes http)

* 1. Vecteurs principaux
* HTTP POST
* HTTP GET
* Cookies
  1. Préventions

Principales :

* Input Validation
* Requêtes préparées

Supplémentaires :

* Réduire la surface d’attaques en donnant le moins de privilèges
* Crypter toutes les données sensibles
* Se méfier des headers http

1. **XSS**
   1. Definition

Le Cross-Site Scripting abrégé XSS est une faille de sécurité des applications utilisant un navigateur Web permettant à un attaquant d’injecter du code malveillant exécutable par ce navigateur. Il s’agit donc d’un cas d’injection de code à un interpréteur qui, à la différence des autres cas d’injection de code qui cible le serveur, cible le navigateur de l’utilisateur.

Le code injecté consiste généralement en du script JavaScript exécuté par le navigateur victime qui le prend pour du code légitime provenant de la même origine que le contenu html affiché. Ce script peut ainsi faire de nombreuses choses, allant d’une simple redirection à une usurpation de l’identité d’un utilisateur par vol des cookies de session. Et dans ce dernier cas, si l’utilisateur en question a des droits d’administrateur sur une certaine application, l’attaquant peut compromettre entièrement ladite application.

On note trois types d’attaques XSS :

* Reflected XSS
* Stored XSS
* DOM XSS

Lors d’une attaque par injection SQL, l’attaquant fournit un/des segments malicieux de code SQL en tant qu’input transformant la requête SQL initiale en une autre souvent dangereuse. Les attaques par injection SQL sont utilisées pour dérober des informations sensibles de la base de données.

Par une injection SQL, l’attaquant, selon ses capacités peut obtenir des informations confidentielles ou même arriver à modifier ces informations.

* 1. Causes

Étant donné que les sites Web dynamiques reposent sur l'entrée de l'utilisateur, un utilisateur malveillant peut entrer un script malveillant dans la page en le masquant dans des demandes légitimes.

Le cross site scripting (XSS) affecte essentiellement des sites qui acceptent des données en provenance de l'utilisateur sans en contrôler correctement la conformité

La cause principale d’une injection SQL est le fait qu’une application accepte des données provenant de sources non sures comme l’utilisateur par exemple (input) ou le navigateur de l’utilisateur (cookies ou headers des requêtes http)

* 1. Vecteurs principaux
* http POST
* http GET
* Cookies
  1. Préventions

Principales :