SOMMAIRE

---

1) Différentes façon de tester ses applications:

Vérification manuelle:

- Formulaire (mettre en place une vérification pour les valeurs rentrées -> vérifications des extensions dans le cas d'upload d'un fichier ou par exemple pour une adresse email vérifier le fait qu'il y'a un "@" et un nom de domaine entre autre (le pattern) , les tester en rentrant des valeurs invalides ou valeurs vides et ensuite correctes).

- Essayer d'accéder à différentes pages avec différents types de comptes-> système de redirection de page (admin, modérateur ou un utilisateur sans rôle, ou même une session sans utilisateur). Par exemple: un utilisateur ayant un compte sans rôle ne peut accéder au panel admin et un utilisateur ne pourra pas consulter la page modifier le mot de passe d'un autre utilisateur.
-

Vérification avec outils:

- On peut utiliser plusieurs types d'outils pour s'assurer que notre application web ne dispose d'aucune faille majeure. Par exemple on peut utiliser Netsparker ou Acunetix Web Vulnerability Scanner qui vont analyser si l'application possède d'éventuelles vulnérabilités et pour effectuer des tests authentifiés, on peut utiliser la Burp Suite composée d'outils comme un serveur proxy (Burp Proxy), robot d'indexation (Burp Spider), un outil

d'intrusion (Burp Intruder), un scanner de vulnérabilités (Burp Scanner) et un répéteur HTTP (Burp Repeater).

Attention: ces outils ne permettent pas d'éliminer tous les risques de subir une cyber-attaque mais préviennent de failles majeures.

2) Faille web connue

Le phishing est une technique utilisée par les hackeurs pour obtenir des renseignements personnels.

Le Cross site scripting correspond au XSS soit l'injection de code HTML dans une page, ce qui provoquent des actions non désirées sur une page Web.

Broken Access Control correspond aux failles de sécurité sur les droits des utilisateurs authentifiés. Les attaquants peuvent exploiter ces défauts pour accéder à d'autres utilisateurs.

Broken Authentification and Session Management : correspond au risque de casser ou de contourner la gestion de l'authentification et de la session. Il comprend notamment le vol de session ou la récupération de mots de passe.

Security misconfiguration correspond aux failles liées à une mauvaise configuration des serveurs Web, applications, base de données ou framework.

Sensitive Data Exposure correspond aux failles de sécurité exposant des données sensibles comme les mots de passe, les numéros de carte de paiement ou encore les données personnelles et la nécessité de chiffrer ces données.

3) Les bonnes pratiques à suivre pour sécuriser une application:

Pour éviter les infections SQL, on peut préparer les requêtes et ou utiliser htmlentities/htmlspecialchar.

Pour éviter le bruteforce (test de différente combinaison une à une), on peut mettre un Captcha, qui va permettre d'éviter le test en masse de compte (souvent grâce à des combo list : email:password ou username:password).
On peut aussi en limiter l'accès à certains répertoires grâce à un fichier .htaccess. Il faut tester les fonctionnalités du site, de manière à ce que les fonctionnalités soient utilisables uniquement à quoi elles sont destinées.
Il faut aussi chiffrer les données sensibles dans la base de données, par exemple les mots de passe.(password_hash() fonction PHP)

Pour éviter d'afficher des erreurs, qui pourrait afficher des données sensibles pour les utilisateurs il faut utiliser : display_error=off et log_errors=on. Le "log_errors=on" va répertorier les erreurs dans fichier log non-visible par tous les utilisateurs et pour repérer les attaques.

Un certificat SSL / protocole HTTPS (Secure Socket Layer) est aussi très important, car ce dernier chiffre toutes les requêtes émises par les clients et donc sécurise le transport des données. Le certificat SSL apporte une garantie de sécurité pour la navigation de tous les utilisateurs.

Pour éviter les attaque DDOS (attaque par dénis de service) (envoie de plusieurs requests en même temps sur la même IP, ce qui rendre le serveur indisponible) on peut mettre des anti DDOS, il y a certains hébergements proposent ce service, mais on peut utiliser le service cloudflare, ce système aura pour but de rediriger toutes les requêtes vers un autre serveur.

Il faut aussi maintenir à jour la version de PHP et les extensions installés, pour éviter toute potentielle faille exploitable. on peut vérifier la version de PHP avec PHP en utilisant phpinfo();

Les identifiants de session sont enregistrés dans un répertoire temporaire. Par défaut, le paramètre session.save_path vaut /tmp est accessible en lecture à tous. Il est donc plus sécurisé d'indiquer le chemin d'un répertoire situé ailleurs sur le système, et dont les droits auront été limités. Cette modification peut aussi se faire à partir du fichier de configuration d'Apache, à l'aide de la variable php_value session.save_path.

Cette variable indique si les identifiants de session doivent être utilisés seulement avec des cookies. Par défaut, cette variable est à 0, elle est désactivée et autorise d'autres modes de lecture, par exemple avec les éléments GET ou POST des requêtes HTTP. En mettant session.use_only_cookies à 1, le système lit les informations d'identifiant uniquement à partir des cookies

Il est impératif que son mot de passe pour accéder aux données sensibles (hébergement, CPANEL, PLESK, phpMyAdmin, VPS, etc…) soit complexe, et bien sécurisé.

Lors d'une validation d'un formulaire, on peut vérifier cela en JS, mais il est impératif de faire les vérifications aussi côté serveur, pour pas que l'utilisateur puisse sauvegarder des données pas bien formatée.

4) Les sources à suivre pour rester informé sur les nouvelles failles

Pour se tenir à jour des nouvelles normes en terme de sécurité:
-https://www.iso.org/fr/isoiec-27001-information-security.html
Pour se tenir à jour des nouvelles failles de sécurités:
-https://www.cert.ssi.gouv.fr/
Pour se tenir à jour des nouvelles recommandations de sécurités:
https://www.ssi.gouv.fr/guide/recommandations-pour-la-securisation-des-sites-web/