Sécurité de l’application des Ports de Marseille:

Failles spécifiques

Le projet des Ports de Marseille de développer une application pour la gestion des ses ressources soulève le problème de sa sécurité et de ses failles spécifiques. Le but de cette application est de pouvoir faire une gestion précise aussi bien des ressources matérielles que des ressources humaines. Chaque employé doit pouvoir se connecter et avoir accès à son statut actuel, sa mission, le quai ou bateau auquel il est affecté. Pour les agents de quai, ils doivent pouvoir savoir l’état des chantiers s’il y en a, ainsi que pouvoir contrôler les entrées et sorties de bateau et le cas échéant connaître l’identité de ses propriétaires, compagnie ou plaisancier.

Cette application n’aura pas de partie réellement public, mais pourra permettre à tous les employés de se connecter et de connaître leur mission journalière. Le statut de la personne connecté au sein de l’entreprise lui permettra l’avoir accès à des informations spécifiques: ainsi les dockers n’auront accès qu’à leur profil là où le chef de quai aura lui accès à tous les profils: marins ou dockers.

Le déploiement de cette application entrainera obligatoirement un risque potentiel de connexion malveillante pouvant entrainer des perturbations ou des fuites d’informations confidentielles, et la sécurité n’est absolument pas à négliger et tout doit-être mis en oeuvre par les développeurs pour réduire au minimum ces risques.

Les différents styles d’attaques:

Attaque par force brute (Brute-force Attack)

Si la liste des mots de passe les plus utilisés en 2020 (https://nordpass.com/most-common-passwords-list/) est représentative de ceux pouvant être fourni par les utilisateurs de notre service, il est évident que des choix concernant le mot de passe doivent être faits. Il n’est pas possible d’avoir une sécurité infaillible, mais il est possible d’augmenter suffisamment le niveau de sécurité pour réduire les risques d’attaques. Ainsi les attaques par Force brute (trouver un mot de passe par entrées successives de toutes les combinaisons possibles) peuvent être ralenties facilement: imposer un temps d’attente de plus en plus long entre les essais ratés controlé par un compteur d’essai invisible à l’utilisateur. L’inconvénient majeur de cette méthode est celui de gêner l’utilisateur légitime de la plateforme/application et d’augmenter sa frustration à son égard.

Un autre choix possible et complémentaire est d'obliger l'utilisateur à choisir à son inscription un mot de passe remplissant plusieurs contraintes: un certains nombre de caractères qui inclus des lettres, chiffres et caractères spéciaux.

Si par exemple on laisse aux utilisateurs la possibilité de rentrer un code de 4 caractères numériques de 0 à 9 uniquement, il n'y aura que 10 000 possibilités (10 choix x 10 choix x 10 choix x 10 choix = 10 000 code possibles) et le « crackage » du code est presque instantané pour un processeur récent. Si toujours pour un code de 4 caractères il est possible de rentrer uniquement des lettres minuscules ou majuscules, là le gain en codes possibles est plus conséquent: 52^4 = 7 311 616 possibilités. Si nous demandons un mot de passe de 8 caractères incluant un chiffre, une majuscule, une minuscule et un caractère spécial au minimum, nous avons 94^8 = 6 095 690 385 410 816 possibilités. Malgré le gain évident de ses choix, il y a aussi le problème pour l'utilisateur de retenir ce mot de passe. Une balance entre ces deux possibilités peut représenter un bon compromis concernant la validation d'un mot de passe.

Si l'application avait une partie publique plus conséquente, la solution d'utiliser une authentification par un groupe comme Google, Microsoft, Apple, autres, serait une solution intéressante. La taille et la sécurité de ces infrastructures étant plus conséquente que celle de développeurs indépendants, le gain de sécurité en est proportionnel. Si l'application devait évoluer vers une partie publique plus importante, des changements sur l'authentification devraient avoir lieux.

Les facteurs bio-métriques sont une possibilité émergeante par l'empreinte digitale, le scan rétinien ou la morphologie faciale. Certaines applications bancaires ou constructeurs de PC portables par exemple l'utilisent actuellement.

Attaque par injection de code: Injection Attacks

L'attaque consiste a injecté du code par l'intermédiaire des invites de commande destinées aux utilisateurs soit pour avoir accès à des informations techniques ou confidentielles, trouver des failles, ou prendre le contrôle de l'application. Outre les entrées d'utilisateurs, il est aussi possible par l'utilisation de logiciels, de faire de requêtes aux serveurs par l'intermédiaire de ces logiciels et ainsi se faire passer pour un navigateur web, la finalité recherchée étant la même: obtenir des informations ou découvrir des vulnérabilités.

- Injection SQL

Le langage SQL étant la base de la communication pour les Systèmes de Gestion de Base de Données Relationnels (SGBDR) et de toutes les variations qui en ont découlé (MariaDB, MySQL, , injecter du code SQL peut permettre aussi d'endommager une base de donnée et la rendre inutilisable, soit pour obtenir des informations confidentielles. La fuite d'informations bancaires ou de mot de passe non-cryptés est un fait d'actualité récurrent.

Si le langage pour la partie back-end est php, l'utilisation de la classe PDO pour les connexions aux serveurs, permet de préparer les requêtes et ainsi éviter l'injection de SQL dans la base de données. Aussi, l'utilisation d'un framework récent et maintenu permet d'avoir un outil de travail prenant en compte les failles récentes de sécurité.

- XSS: Cross Site Scripting

Un script écrit par un utilisateur est accessible/incorporé dans la page d'un autre utilisateur. Par exemple un commentaire sur un blog incorpore un lien qui incorpore un lien qui capte les informations d'un autre utilisateur ainsi vulnérable.
Cette faille de sécurité peut être contré par un contrôle sur les inputs des utilisateurs et en supprimer toutes balises de script: nommé assainir (sanitize) les entrées utilisateurs, cette opération indispensable permet de supprimer ces risques.

Tester son code:

Une façon très simple est de réaliser un test avec du code Javascript élémentaire dans un input d'utilisateur, affiché sur une page par la suite ou stocké en Base de Donnée:

<script>alert('bonjour')</script>

Si un message d'alerte apparaît, alors l'input n'a pas été filtré convenablement.
Côté serveur, le développeur se doit au minimum de filtrer en utilisant la fonction htmlentities() ou htmlspecialchars(), avant tout enregistrement d'un input utilisateur.

Authentification par deux-facteurs

Choisi par Google par exemple, lorsqu'une nouvelle adresse IP est utilisé lors d'une authentification, elle envoi un code de vérification soit sur une adresse email fourni par l'utilisateur, soit par message sur son téléphone, avec une date d'expiration relativement courte.

Utilisation d'une stratégie de sécurité du contenu (CSP)

Une autre possibilité et celle d'utiliser une « stratégie de sécurité du contenu » .
Par le rajout d'un header spécifique et une d'une configuration des serveurs web, il est possible de proscrire l'exécution de script JavaScript outre là où celle est autorisé. Il est aussi plus simplement possible de rajouter une balise meta dans le head de la page pour arriver au même résultat.

Attaque par le DOM: DOM-Based Cross-Site Scripting Attacks

Dans le cadre de l'application des ports, l'affichage de la liste des marins peut-être longue et ainsi utiliser une manipulation de l'URL avec un fragment URI, qui permettra de charger par vague les profils à afficher. Cette fonction pratique pour charger progressivement la page peut là encore être une faille de sécurité. L'utilisation d'un framework récent est donc une obligation pour éviter ce risque et utiliser fonctions et méthodes pouvant échapper toute inclusion de code dans l'URL.

Falsification de requête inter-site (CSRF)

Cette attaque a pour but de « forcer » un utilisateur à accomplir des actions qu'il ne souhaite pas faire: changement d'adresse mail d'un compte, transfert de fond, etc… . Si le compte en question était celui d'une administrateur ou un poste à haute fonction, la sécurité du site pourrait être compromise.

Veille de sécurité

La fondation Owasp (https://owasp.org/) et leur dépôt relatif sur Github, permet d'avoir un profil complet des attaques ainsi que des solutions possibles. Le fil des actualités générales peut incorporer des faits en rapport avec la sécurité: leak de mot de passe massif, bases de données exposées. Ces actualités doivent être une piqure de rappel de l'importance d'une sécurité adaptée.

La lecture de livre sur le sujet de la sécurité peut être instructif mais malheureusement les livres se positionnent dans le temps long là où un site ou un fil d'actualité permet de se tenir au courant au jour le jour. Par contre sur les réseaux sociaux comme twitter, suivre les auteurs ou sites spécialisés comme Treatpost (https://threatpost.com/) , Hacker News (https://thehackernews.com/) ou CyberNews (https://cybernews.com/) permet de faire une veille plus efficace.