## ◌ Les failles humaines

Phishing :
Ex : Envoi d'un mail frauduleux sur une adresse de messagerie professionnelle éventuellement par le biais d'un email spoofing
La cible vient ensuite à communiquer des données compromettantes soit dans le mail, soit via un faux site web.
Prévention : Messagerie interne à l'entreprise fermée. Faire de la prévention parmi les salariés.

Propagation d'un virus :

Ex : Un employé a été infecté par un virus silencieux à activation de type worm, trojan, backdoor etc. Celui-ci se réplique sur sa clé USB ou sur son ordinateur de travail.
Par la suite, lorsque le particulier sera connecté au réseau de l'entreprise, celui-ci se propagera sur tous les appareils disponibles, pouvant mener par exemple à l'applicaton d'un ransomware.
Prévention : Limiter les outils de travail à une utilisation interne stricte, interdire l'utilisation de périphériques extérieurs à l'entreprise. Faire de la prévention parmi les salariés.

Social engineering :

Ex : Similaire au phishing mais dans un cadre plus vaste, permettrait par exemple pour un utilisateur malveillant connaissant la victime ( animal de compagnie, famille, activités etc ) de modifier un mot de passe via un outil de récupération de mot de passe intégré au site / à l'application proposant des questions secrètes.
Prévention : Ajouter une étape supplémentaire aux questions secrètes comme une vérification IP. Faire de la prévention sur

les données sensibles et les réseaux sociaux, et les questions secrètes.

- **Les failles serveur**

Attaque DDoS :

Ex : Un utilisateur malveillant obtient l'adresse IP du serveur hébergeant le site web cible. Il peut via l'utilisation de bots/machines zombies ou dans des cas spécifiques par l'envoie de fichiers volumineux réaliser une attaque par déni de service permettant de ralentir considérablement la navigation voir même de couper le serveur.

- **Les failles techniques**

Injections SQL :

Ex : Lors d'une requête de connection où la condition serait "WHERE `login` = $login AND `password` = $password", un utilisateur malveillant pourrait rentrer comme $password '0 OR 1=1' validant la condition WHERE.
Prévention : Utiliser les requêtes préparées.

Les failles XSS :

Ex : Un utilisateur malveillant rentre du code <script> dans un formulaire mal encadré ou directement dans l'url, menant à l'exécution de celui-ci ( immédiatement ou via un stockage en bdd lors de son appel ).
Prévention : Filtrer les inputs utilisateur, nombreuses méthodes possibles comme htmlspecialchars, pregmatch, quote, etc.

Les attaques Bruteforce :

Ex : Un utilisateur malveillant rentre des des milliers de combinaisons login/password par le biais d'un script ou d'une application, et l'utilisation de caractères aléatoires, de "dictionnaires" ( contenant des mots de passes communs ) ou de "rainbow tables" ( permettant de reproduire un hash plutôt qu'un mot de passe ).
Prévention : Interdire les mots de passe faibles. Recommander les bonnes pratiques aux utilisateurs ( mots de passe uniques ). Limiter le nombre de tentatives.

Les sessions hijacking / cookie stealing

Ex : Par l'utilisation d'une faille XSS ou d'un réseau wifi public, un utilisateur malveillant récupère un cookie d'authentification utilisateur ou sa session. Il peut alors agir comme s'il était l'utilisateur ciblé.
Prévention : Informer les utilisateurs, faire de la prévention sur les wifi publics. Ne jamais limiter l'authentification à la seule présence du cookie, mais ajouter une seconde vérification comme l'adresse IP.

Les attaques CSRF :

Ex : Un site propose la possibilité de transférer de l'argent entre ses utilisateurs par le biai d'un formulaire de type POST, avec un champs "somme", "id_sender", "id_receiver". Un utilisateur malveillant pourrait envoyer ce formulaire dissimulé dans un bouton via un mail frauduleux à un autre utilisateur de ce site, actionnant le POST par son clic.
Prévention : De bonnes pratiques de code, de vérification, et de sensibilisation aux mails frauduleux.

Clickjacking :

Ex : Un utilisateur malveillant créé un faux site pour remporter le dernier iPhone. Ce site est situé une couche au-dessus d'un iframe contenant la page paypal de l'utilisateur, ou de son compte en banque. Le positionnement du faux site par dessus l'autre est de telle sorte que le bouton "Demander mon iPhone" est situé à l'endroit exact du bouton "Envoyer de l'argent".
Prévention : Interdire la possibilité d'afficher son site dans un iframe en configurant le header du htaccess.

Les attaques Directory traversal :

Ex : Un site utilisant des includes prend en paramètre un $_GET de l'url, et affiche la page correspondante. Un utilisateur malveillant peut ainsi "scanner" un site mal configuré et accéder à des dossiers qu'il ne devrait normalement pas grâce à des suites de '../'.
Prévention : Définir les routes ( deny all sauf les dossiers publics ), vérifier si l'url correspond à un fichier .php, interdire l'affichage des dossiers, etc ...

Le Remote File Inclusion :

Ex : Même système qu'une attaque Directory traversal, sauf qu'ici le $_GET pointe vers l'exécution d'un fichier ou d'un script distant.
Mêmes préventions.

Les brèches de données :

Ex : Suite à une faille humaine, serveur ou technique, tout ou

partie de la base de donnée est volée ou détruite par un utilisateur malveillant.
Prévention : Si malgré tout un utilisateur réussit à s'attaquer à la base de données, pour prévenir tout dommage immédiat ou collatéral, il faut au préalable chiffrer toutes les données sensibles, avec des systèmes de cryptographie mis à jour et sécurisés ( cas par exemple d'un service médical ). Eventuellement limiter l'accès à certaines parties de la base de données à des utilisateurs spécifiques identifiés. Ensuite, il faut toujours conserver une copie locale de cette base de données, et effectuer des sauvegardes régulières par exemple chaque nuit.