SECURITE PHP

1) attaque XSS -Cross-Site Scripting (JS dans un champ input)

Exemple dans un champ input on saisit un script JS
Ce nom s(inscrit dans la BDD
Lorsque l'on affiche ce nom le script , s'il n'est pas "échappé" c'est à dire que les caractères spéciaux ne sont pas protégés , ce script va s'exécuter.

Pour se protéger de ce risque

```
function secure_input($data) {
      $data = trim($data);
      // supprime les espaces en début et fin de chaine
      $data = stripslashes($data); // supprime les "/"
      $data = htmlspecialchars($data); // échappe caractère spéciaux genre <>
      return $data;
}

htmlentities(). // transforme tous caractères transformable en entité
html les caractères.
```

Vous pouvez l'exécuter avant d'écrire dans la BDD et/ou avant l'affichage.

2) Injection SQL

si je saisi un email et un password pour me connecter il y a des chances que quelque part dans mon application j'ai la requête SQL:

SELECT * FROM users WHERE login ="monlogin" AND password = "monpassword"

Type d'attaques:

login: " or ""="
password: " or ""="

le hacker va considérer qu'il y aura une requête du genre

SELECT * FROM users WHERE login ="" or ""="" AND password ="" or ""=""

renverra toute la table.

soit le champ input de id_produit

id_produit: 105; DROP TABLE Suppliers

SELECT * FROM produits WHERE id_produit = 105; DROP TABLE Suppliers;

supprimera la table Suppliers
Destruction de ta table suppliers

PDO

Utiliser les paramètres nommés avec la fonction prepare de l'objet PDO pour vérifier qu'on a le nombre de paramètres requis et aussi le type peut être vérifié. (voir PDO)

Exemple de prepare avec paramètre nommés

```
$nom = "Flo";
$prenom = "Dechand";
$adresse = "Rue des Moulins";
$ville = "Marseille";
$cp = 13001;
$pays = "France";
$mail = "flodc@gmail.com";

//$sth appartient à la classe PDOStatement
$sth = $dbco->prepare("
INSERT INTO Clients(Nom, Prenom, Adresse,  Ville,  Cp,  Pays,  Mail)
           VALUES (:nom,:prenom,:adresse, :ville, :cp, :pays, :mail)
```

```
            ");

    $sth->execute(array(
                                    ':nom' => $nom,
                                    ':prenom' => $prenom,
                                    ':adresse' => $adresse,
                                    ':ville' => $ville,
                                    ':cp' => $cp,
                                    ':pays' => $pays,
                                    ':mail' => $mail));
            echo "Entrée ajoutée dans la table";
        }
```

SECURITE NODE/EXPRESS

1) attaque XSS

utiliser le module helmet.js qui protège des attaques dans l'entête et qui échappe les caractères spéciaux .

helmet protege des attaques XSS ainsi que les attaques liées aux informations de l'entête http

```
const express = require("express");

const helmet = require("helmet");

const app = express();

app.use(helmet());
```

2) Injection SQL

Pour le module mysql2 utiliser le prepare ainsi

```
connection.execute(
  'SELECT * FROM person WHERE name = ? AND age > ?',
  ['Rick', 53],
  function(err, results) {
    console.log(results);
  }
);
```

Cross Site Request Forgery - XSRF (ou CSRF)

Définition

L'attaque XSRF (ou CSRF pour Cross Site Request Forgery) est une attaque silencieuse mais redoutable. Quand elle survient, la victime ne s'en rend souvent pas compte, et ses dégâts ne se font sentir que plus tard.

Le principe de l'attaque consiste à faire exécuter une opération qui demande des privilèges spéciaux par une personne authentifiée et autorisée sans qu'elle ne s'en rende compte.

Sur le back-office d'un site web par exemple, l'administrateur et ses modérateurs ont des privilèges qui leur permettent de gérer, à leur guise, le contenu présenté en front-office. Ils peuvent alors créer, modifier, supprimer... depuis un tableau de bord facile à utiliser.

Si par exemple, l'administrateur souhaite supprimer une entrée d'une base de données, il va tout simplement cliquer sur le bouton (ou lien) approprié. Ce lien enverra les paramètres qui permettent d'identifier l'enregistrement à supprimer, et puisque l'administrateur est authentifié et dispose des privilèges nécessaires pour supprimer l'entrée, alors l'opération se passe sans encombre.

L'attaque XSRF consiste donc à forger le lien de la suppression (dans ce cas) et l'envoyer à l'administrateur, et c'est lui qui l'exécutera à son insu.

Outre la suppression, le XSRF peut causer d'autres dommages au contenu du site Web comme:

- L'ajout ou la modification de contenu d'une manière non autorisée.
- Exécution des commandes systèmes via l'interface Web qui peuvent avoir des conséquences désastreuses.

Exploitation

Imaginons que le lien de suppression d'un cours sur ce site est **admin/cours/supp.php?num_cours=120**. La page **supp.php** est faite de telle façon à s'exécuter uniquement si l'utilisateur est authentifié et autorisé (je vous conseille d'aller voir l'exercice sur les sessions dans le cours de PHP). Par conséquent, si l'administrateur exécute ce lien, il s'exécutera normalement et la suppression sera effectuée.

Le pirate quant-à-lui, ne dispose pas des privilèges nécessaires pour exécuter le lien, mais pour une raison quelconque, il connait le lien de la suppression ci-dessus. Tout ce qu'il a à faire, c'est d'envoyer ce lien à l'administrateur (en guise de

message) en utilisant le menu contact par exemple. Mais l'astuce c'est encapsuler ce lien dans une balise image **<img>** comme ceci:

****

Si la vulnérabilité existe, il suffit que l'administrateur ouvre le message pour que le lien qui constitue la source s'exécute avec ses privilèges et la suppression sera donc faite sans faire de bruit.

Comment s'en protéger?

Au niveau du code PHP

On peut se prémunir contre les XSRF de plusieurs manières différentes:

Filtrer les entrées de l'utilisateur:

Comme pour les vulnérabilités précédentes, l'attaque XSRF se base sur l'insertion de certains caractères spéciaux dans les entrées d'un site Web (formulaire, URL...). Comme ces données là ne sont pas fiables alors il faut les filtrer en limitant leur longueur, en échappant les caractères spéciaux et en éliminant les balises à l'aide de la fonction **strip_tags()**.
ou utiliser la fonction secure_input page 1.

Faire confirmer toutes les actions irréversibles avant de les exécuter:

Avant de supprimer ou modifier une entrée, il convient de demander confirmation auprès de l'utilisateur, car des fois, même ce dernier peut appuyez accidentellement sur le bouton de suppression. D'autant plus, si le site était victime d'une attaque XSRF, un message de confirmation de l'action serait affiché devant l'administrateur qui se douterait alors que quelque chose d'anormale est en train de se passer.

Aussi l'utilisation des jsonwebtoken protège d'une attaque CSRF.

HTTPS

CF diaporama keynote.
https.key