20/01/2022

STI-Projet 2

Ryan Sauge & Michael Ruckstuhl
HEIG-VD

Table des matières

1 Introduction

Ce document décrit une modélisation des menaces réalisées sur l'application STI-PRO-01disponible à l'adresse suivante : https://github.com/rya-sge/STI-PRO-01

2 Descriptions du système

2.1 Objectif du système

Il s'agit d'une application mettant en œuvre une messagerie électronique au sein d'une entreprise.

- Les utilisateurs peuvent envoyer et de recevoir des messages d'autres utilisateurs.

-Les utilisateurs (rôle, état du compte) sont gérés par un ou plusieurs administrateurs

2.2 Hypothèse de sécurité

- Réseau interne et administrateurs de confiance
- Système d'exploitation et serveur web de confiance

2.3 Exigence du système :

- Le site web doit être accessible et fonctionnelle
- Les informations des utilisateurs doivent être scrupuleusement protégé(privacy)
- Les données (ex : messages) doivent être conservées et ne pas pouvoir être altérées

2.4 Eléments du système

- Un serveur web apache
- Une base de données SQLite
- Des utilisateurs dont un utilisateur admin

2.4.1 Rôle des utilisateurs

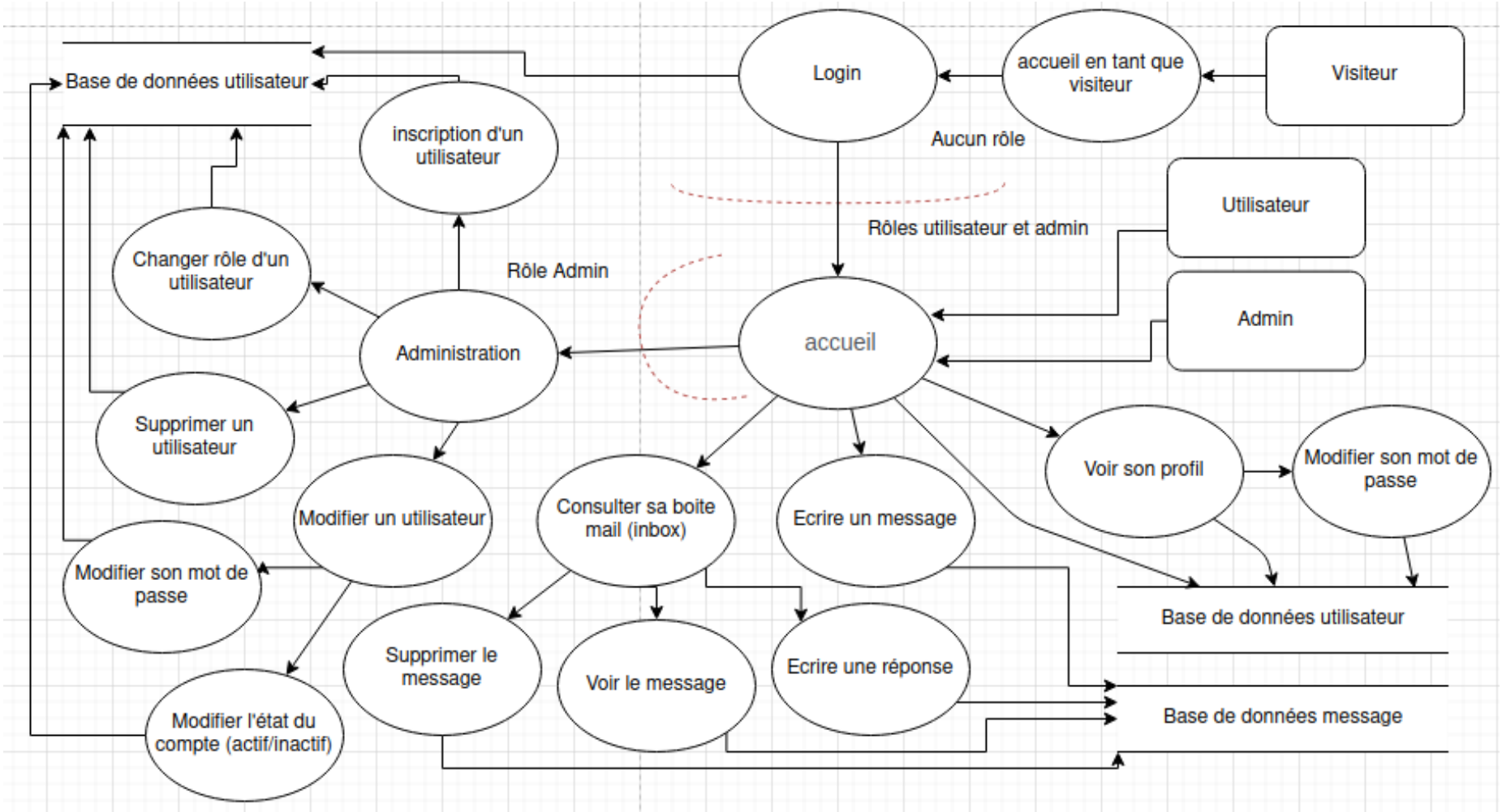
- Utilisateur
- Administrateur
- Rien (Visiteur)

2.4.2 Actifs à hautes valeurs

Actif	Description
Base de données utilisateurs	Les mots de passes (même hashés) sont confidentiels. Un accident nuirait à la réputation du site
Base de données messages	Les messages échangés sont confidentiels Un accident nuirait à la réputation du site
Base de données des logs	Les données contenues dans les logs du serveur apache.
Infrastructure	Intégrité, disponibilité Un incident serait critique et nuirait à la disponibilité/réputation

2.5 DFD

Ce DFD décrit les principaux composants et acteurs du systèmes avec les flux les reliant


Base de données utilisateur
Login
accueil en tant que visiteur
Visiteur
inscription d'un utilisateur
Aucun rôle
Utilisateur
Rôles utilisateur et admin
Changer rôle d'un utilisateur
Rôle Admin
Admin
Administration
accueil
Supprimer un utilisateur
Voir son profil
Modifier son mot de passe
Modifier un utilisateur
Consulter sa boite mail (inbox)
Ecrire un message
Modifier son mot de passe
Base de données utilisateur
Supprimer le message
Voir le message
Ecrire une réponse
Modifier l'état du compte (actif/inactif)
Base de données message

3 Identifier les sources de menaces

Chaque source de menace possible est caractérisée par 4 propriétés : Acteurs, ses motivations, sa ou ses cibles principales en fonction de ses motivations ainsi que sa potentialité.

Acteurs	Motivation	Cible	Potentialité
hackers/scripts kiddies	S'amuser, gloire, curiosité	N'importe quels éléments, Les fonctionnalités " écrire des nouveaux messages et répondre à un email " pourraient être une cible attirantes"	Potentialité : haute
Cybercrime (Spam, maliciels)	Financière	vol de crédenciales du client, spam des clients, modification d'informations	haute
Activiste	Montrer que l'application doit être plus sécurisée. Démontrer que la société fait des choses peu éthiques. Conviction idéologique ou politiques en désaccord avec l'entreprise.	Lire les messages, prendre les accès d'un compte pour usurper une un utilisateur et écrire des messages. Dénis de service. Supprimer message et utilisateurs (pour saboter l'entreprise).	Faible à moyenne (dépend de l'éthique de l'entreprise qu'on ne connait pas car celle-ci est fictive)
Nos concurrents	Ternir la réputation, voler du code, voler des informations confidentielles	Liste des messages reçus et envoyés, prendre l'identité d'un autre employé pour demander des informations (usurpation de session)	Moyenne
Concurrent d'un utilisateur	Obtenir des informations sur leur concurrents : interlocuteur, finance, technologie	Liste des messages reçus et envoyés, prendre l'identité d'un autre employé pour demander des informations (usurpation de session)	Moyenne
Utilisateurs malins	Rendre actif un compte inactif	Sa page de profil et les fonctionnalités vue du point de vue d'un	Moyenne voir faible, hormis pour contacter une personne en particulier, l'intérêt

		administrateur + page de création de compte	pour un utilisateur moyen est faible (pas d'accès payant par exemple + il y a suffisamment de services de messagerie concurrents)

4 Identifier les Scenarios d'attaques

- Cette analyse représente des hypothèses émises
- Les scénarios d'attaques n'ont pas étés testés sur le site

4.1 Stride

Source : https://owasp.org/www-pdf-archive/STRIDE_Reference_Sheets.pdf

4.1.1 Spoofing

Une personne

Cas 1 : S'authentifier à l'application avec un mot de passe volé et/ou cassé

Elément du système attaqué

- Fonctionnalité du mot de passe
- Base de données (essayer d'obtenir les hash puis les mots de passe en clairs)

Motivations :

- Usurper l'identité pour avoir les droits d'une autre personne
- Lire ses messages ou en écrire en se faisant passer pour une autre personne

Contre-mesure :

- Vol : HTTPS, authentification 2FA
- Se protéger contre les injections SQL
- Cassage du mot de passe : politique de sécurité du mot de passe, captcha, code en temps constant (Contre timing attack), temps d'attente entre chaque mot de passe

Cas 2 : Modifier le destinataire ou l'émetteur d'un message pour y mettre son propre compte

Elément du système attaqué

- Requête à la base de données
- Paramètre des URL
- Objet envoyé lors de la communication

Motivations

- Cacher un message
- Voler un message
- Réceptionner un message à la place d'un autre utilisateur
- Se faire passer pour quelqu'un d'autre

Contre-mesures

- Se protéger des injections SQL
- Utiliser la session pour identifier l'utilisateur
- Chiffrer la communication avec du HTTPS (éviter attaque MITM)

Un rôle

Cas 3 : Modifier son rôle pour être considéré comme admin

Elément du système attaqué :

- Page changement de rôle de l'administrateur
- Cookies

Motivations

- Gain de privilège
- Sur l'application permet de supprimer un utilisateur ou faire perdre des privilèges aux autres.

Contre-mesures :

- Ne pas mettre les informations d'autorisation dans les cookies ou les signer(ex : jwt)
- S'assurer qu'aucune page du site ne puisse donner accès à une modification qui permet des gains de privilège.
- Les pages permettant des gains de privilèges doivent être accessibles seulement à ceux qui ont le droit de les changer. (Ex. Administrateur peux élever au rang d'administrateur un utilisateur simple. Mais l'utilisateur n'a pas ce droit).

-

4.1.2 Tampering

Falsification sur le réseau

Cas 1 : Rediriger le flow sur la machine de l'attaquant

Elément attaqué :

- Page web coté client en y injectant une attaque XSS
    - Input et output utilisateur

Motivation :

- Lors du chargement de la page, un script malicieux enverra des données de l'utilisateur ou du serveur à l'attaquant.

Contre-mesures

- Sanitizer les output utilisateur qui vont se retrouver dans du code javascript et html.

Cas 2 : Modifier les données transmises sur le réseau

Elément attaqué :

- Les requêtes html, modifiables grâce à des proxy, genre Burp Suit

Motivation :

- Modifier le contenu d'un message
- Modifier les éléments (ex. Destinataires, source d'un message ou id d'un message à supprimer)

Contre-mesure :

- Chiffré la communication (HTTPS)

Falsification sur la mémoire

Cas modifier le code

Elément du système attaqué :

- Injection sql

Motivation :

- Modifier ou supprimer des données dans la base de données

Contre-mesure:

- Utiliser des requêtes préparées, esaper les entrées utilisateurs

4.1.3 Répudiation

Répudier une action

- Nier l'envoi ou la réception un message, par exemple en changeant le compte émetteur ou récepteur
- Example : Modifier l'émetteur ou le récepteur
- Contre-mesure : demander l'adresse de confirmation et autres informations d'authentification pour confirmer

Attaquer les logs

- L'attaque arrive à déterminer qu'il n'y a pas de logs
- Si présence de logs, les corrompre pour qu'il ne puisse pas être utilisé comme évidence, par exemple en générant beaucoup de logs.

Notre système aurait pu implémenter un système de logs pour vérifier les menaces de "Repudiation".

4.1.4 Information disclosure

Par exemple, contre les données enregistrées

- Trouver des fichiers non protégés
- Récupérer des données depuis les logs ou fichiers temporaires

Contre-mesure: fichier .htaccess

Lire des données sur le réseau

Elément du système attaqué :

- Communication client serveur

Motivation :

- Lire des données secrètes ou des données confidentielles
- Savoir qui parle à qui
- Changer l'émetteur pour que l'attaquent reçoive des réponses

Contre-mesure:

- Chiffrer la transmission
- Eviter au maximum l'envoi de secrets

4.1.5 Déni de service

Motivation pour les points suivants : Restreindre la disponibilité

Déni de service contre une base de données

- Remplir la base de données, par exemple
    - Via les inputs utilisateur
    - Insérer des données de tailles nettement plus importants que prévues par les systèmes
        - Contre-mesure : valider la taille avant allocation (validation des entrées utilisateurs)
- Faire suffisamment de requêtes pour ralentir le système

Déni de service contre un un flux de donnée

- Consommer les ressources réseaux

Déni de service contre un processus

- Absorber la mémoire
- Absorber le CPU

4.1.6 Elévation de privilège

Motivation : avoir plus de privilèges

En corrompant le processus

- Envoyer des inputs qui ne sont pas supportés proprement
- Obtenir des accès pour lire ou écrire en mémoire de manière inappropriée

A travers des vérifications oubliés d'autorisation

- Example : Copier/Coller une url admin avec un compte utilisateur normal
- Contre-mesure : mécanisme d'autorisation

A travers des bugs dans les vérifications d'autorisations

Contre-mesure : centraliser les vérifications pour rendre la rechercher et la gestion des erreurs plus simples

5 Identification des vulnérabilités sur le site

Dans le dossier **rapport** du projet git, des readme expliquent des scénarios concrets d'attaques sur l'application cible : XSS, CSRF et timing attack et injection SQL

De plus, il était possible de supprimer ou d'accéder à des message dont on n'avait pas les droits, ceci a été corrigé dans la partie implémentation avec une meilleur vérification des droits.

5.1 Théorique

Sur stride, des contres mesures en été données en fonction des scénarios d'attaques

5.2 Implémentation

Les sources suivantes ont été utilisées :

Travail de fin de CFC de Ryan Sauge : https://github.com/rya-sge/travail-cfc/tree/main/code/library

Projet de semestre réalisé à l'HEIG-VD : https://github.com/rya-sge/PRO-Angular-Php/blob/master/dev/Backend/zone_protected/security.php

5.2.1 CSRF

Source principale : https://gist.github.com/subhendugiri/69db7e8e276bfd348385b24aa4f2d7a5

5.2.1.1 Token

1) Création d'un token propre à l'utilisateur et stocké dans une variable de session php

```
$_SESSION['token'] = bin2hex(mcrypt_create_iv(32, MCRYPT_DEV_URANDOM));
```

2) Ajouter du token dans un champ caché pour chaque formulaire

```
<input name="token" value="<?php echo $_SESSION["token"] ?>" type="hidden" />
```

3)Vérifier le token en appelant la fonction verifCSRF() et lever une exception si le token du formulaire ne correspond pas au token de l'utilisateur

```
function verifCSRF(){
   $isValid = false;
   if (!empty($_POST['token'])) {
      if (hash_equals($_SESSION['token'], $_POST['token'])) {
         $isValid = true;
      }
   }
   if(!$isValid){
      throw new Exception("Le formulaire ne peut pas être validé");
   }
}
```

Hypothèse de sécurité : Le canal de transmission doit être sûr (HTTPS) car sinon il est possible pour un attaquant qui intercepte le traffic (par ex : wireshark) d'obtenir le token CSRF et de l'utiliser pour valider les formulaires.

5.2.1.2 Requête POST

Le remplacement des requêtes GET par des requêtes POST permet ensuite d'ajouter ce token CSRF en plus de diminuer le risque de fuite dans les logs.

5.3 XSS

On peut utiliser **htmlspecialchars** pour échappés les caractères entrés par l'utilisateur

```
function erreurText($champ1)
{
   $champ1 = htmlspecialchars($champ1, ENT_QUOTES);
   return $champ1;
}
```

5.3.1 Injection SQL

Les requêtes sont préparées et on réalise un bind des paramètres avant qu'elles ne soient exécutées.

```
   $db = getBD();
   // Création de la string pour la requête
   $requete = $db->prepare("UPDATE user
                            SET idRole = :idRole
                            WHERE name = :name
                            AND id !='" . $_SESSION['idUser'] ."'
                            AND id != 1;");

  //On bind les paramètres
   $requete->bindValue(':idRole', $idRole, PDO::PARAM_INT);
   $requete->bindValue(':nom', $name);

 // Exécution de la requête
   $requete->execute();
```

5.4 Rôle

Des fonctions permettent de vérifier le rôle de l'utilisateur et de réaliser une redirection le cas échéant. Celle-ci était déjà présente dans la 1ère version du projet mais pas tout le temps employée.

- Administrateur

```
/*
* @brief vérifie si l'utilisateur a un grade équivalent à 1 (administrateur).
* @return true si le grade de l'utilisateur est 1, false sinon
*/
function testR1()
{
 if(isset($_SESSION['idRole']) && $_SESSION['idRole'] == 1)
 {
return true;
 }
 return false;
}

/*
* @brief Appel de test1(), effectue une redirection si testR1() == False
*/
function testR1Out()
{
```

```
   if(!testR1()){
      header("location: index.php?action = vue_accueil");
      exit;
   }
}
```

- Utilisateur connecté

```
function isConnected()
{
    //L'utilisateur est connecté
   if (empty($_SESSION['login']))
   {
        header("location: index.php?action = vue_accueil");
       exit;
   }
}
```

- Correction du workflow

Lors du premier projet, suite à une mauvaise compréhension de la consigne, la page d'inscription/création de compte était accessible à tous. Nous avons fait en sorte grâce à la gestion des rôles qu'elle ne soit accessible que pour les administrateurs

- Mot de passe

Nous avons défini un nombre minimum de caractères à 12. Comme amélioration possible se serait d'ajouter un captcha contre les attaques par brute-force ainsi que d'établir une politique de sécurité de mot de passes.

- Message

Que ce soit pour l'affichage ou la suppression, on vérifie que l'utilisateur a bien le droit de lire le message spécifié en paramètre

```
$isPresent = selectMessageUserFromId($idMessage);
if (empty($isPresent['id'])) {
    throw new Exception("Erreur : vous ne pouvez pas supprimer ce
message");
}
```

6 Conclusion

Nous avons apprécié ce deuxième projet pour les raisons suivantes

- Le fait d'analyser les sources de menaces nous a fait comprendre que ceci été important pour mieux focaliser nos efforts sur les points critiques à protéger.
- Nous avons apprécié pouvoir nous exercer à la défense d'un système et non uniquement l'attaque d'un de ce dernier.