­­

PROJET SYNTHÈSE

IFT606 – Sécurité & cryptographie

Présenté à

M. Marc Verreault

M. Mohammed Ouenzar

Par

Kevin Boisvert – 16086334

Denis Ahmed Samson Abel

Jean-François Breton – 15057796

Samuel Perron-Desrochers

UNIVERSITÉ DE SHERBROOKE

02 août 2018

# Description et buts du projet

# OWASP + description de ce qu’on a fait pour prévenir

* **A1-Injection** -> mettre «‘ or ‘’=’» dans login et password pour connexion
* **A2- Broken Authentication** -> Force users to use a strong password policy. Stored username and password values should be salted and hashed, in addition to being encrypted. An authenticated user of the site wants to let their friends know about the sale. The user e-mails the link above without realizing they are also giving away their session ID. When the friends use the link they use the user’s session and credit card. (https://hdivsecurity.com/owasp-broken-authentication-and-session-management)
* **A3-Sensitive data exposure** -> Password fields sont caches + SSL/TLS
* **A4-XML External Entities** ->
* **A5-Broken Access Control** ->
* **A6-Security Misconfiguration** ->
* **A7-Cross-Site Scripting (XSS)** -> Encoder et vérifier les entrées
* **A8-Insecure Deserialization** ->
* **A9-Using Components with Known Vulnerabilities** ->
* **A10-Insufficient Logging&Monitoring** ->

# Distribution des rôles

Pour la distribution des rôles, nous avons divisé le travail en quatre parties où chacun des membres choisissait celle qu’il voulait faire. Au tout début, nous devions commencer par créer la base du projet pour que tout le monde puisse débuter du même endroit. Kevin s’est occupé de faire cette partie en créant un projet web et d’y connecter la base de données, de créer chacun des fichiers nécessaires au développement ainsi que de s’assurer que le projet démarre correctement et sans erreur dans le navigateur. Il a ensuite dupliqué le projet pour en faire une version sécurisée et une non sécurisée. Après ces étapes complétées, le développement de chacun pouvait commencer. Chaque membre pouvait travailler sur sa partie indépendamment du travail des autres, ce qui facilitait beaucoup la gestion des conflits dans le code.

Kevin avait comme rôle de configurer tout le système de connexion ainsi que de la création d’un compte utilisateur. Pour la version non sécurisée du projet, le travail fût beaucoup plus simple que la version sécurisée. Celle-ci ne devait avoir aucune opération de vérification ni de sécurité. Pour la version non sécurisée, lors de la création du compte, l’utilisateur entre son nom, son prénom, se choisit un nom d’utilisateur unique ainsi qu’un mot de passe. Le système ajoute ensuite les informations entrées dans la base de données sans opération de sécurité supplémentaire. Celui-ci peut donc entrer ce que bon lui semble dans chacun des champs du formulaire. Une fois ces étapes effectuées, l’utilisateur est redirigé vers la page d’accueil et peut naviguer sur le site. À l’avenir, l’utilisateur possédant un compte peut se connecter avec son nom d’utilisateur et son mot de passe. Le système fait ensuite une vérification dans la base de données et retourne vrai si les informations entrées sont bonnes. Ensuite, pour la version sécurisée, plusieurs étapes de vérification et de sécurité ont été ajoutées à la connexion ainsi qu’à la création du compte. Kevin a commencé par cacher les champs du mot de passe pour éviter qu’il soit affiché lors de la connexion. Cette étape simple permet de prévenir l’exposition de données sensibles à quiconque pouvant voir l’écran de l’utilisateur. Ensuite, pour éviter les injections SQL dans le projet, Kevin a changé les simples « statements » pour des « prepared statements » pour chaque requête SQL effectuée à la base de données. Ceci évite d’entrer, dans les champs de connexion, des caractères comme : « ‘ or ‘’=’ » qui permettrait à un pirate de se connecter facilement au site. De plus, pour le formulaire de création de compte, plusieurs vérifications ont été ajoutées pour guider l’utilisateur vers un choix sécuritaire de mot de passe. On lui oblige à entrer un mot de passe de minimum 8 caractères avec au moins une majuscule et un chiffre. Celui-ci est par la suite « hashé », avec l’algorithme MD5, avant d’être mis dans la base de données. Donc, au final, chaque utilisateur possède un mot de passe moyen ou robuste ce qui prévient en partie la vulnérabilité « Broken Authentification » et réduit grandement les possibilités de briser un mot de passe. Pour prévenir davantage cette vulnérabilité, Kevin a retiré le ID de la session dans le URL pour la version sécurisée, celui-ci étant visible dans la version non sécurisée du site. Bref, pour la partie de connexion et de création de compte, trois vulnérabilités ont été traitées pour prévenir, en partie, les attaques d’authentification et de session. Kevin s’est aussi chargé de l’interface graphique de la page de connexion et de création de compte des deux versions du projet.

Dans le cas de Jean-François, il avait pour rôle d’intégrer un service de discussion instantané dans la page d’accueil de l’application web. Cette partie du projet avait comme principale cible de permettre l’ajout d’une vulnérabilité au « Cross Site Scripting (XSS) » en permettant à l’utilisateur d’entrer un script malicieux et de l’envoyer aux autres utilisateurs en ligne par l’entremise du service de discussion. Pour remédier à la vulnérabilité dans la version sécurisée de l’application, il a suffi de vérifier et d’encoder les caractères spéciaux dès leurs entrées par l’utilisateur malicieux. Également, une modification des fonctions utilisées pour la réception des données de clavardage par les utilisateurs fut effectuée. Ainsi, on évite l’utilisation d’une fonction telle que « eval () » pour la lecture du JSON, fonction qui permet la lecture de script et ne vérifie pas qu’il s’agit bien de données JSON.

En plus de prévenir l’entrée de code malicieux dans le service de clavardage, Jean-François a également eu comme rôle d’assurer l’intégrité et la confidentialité des transactions entre les clients et le serveur. Pour y arriver, il a fait usage de l’outil « keytool » de Java pour générer un certificat autosigné et une paire de clés RSA. Ensuite, il suffisait de modifier la configuration du serveur pour y ajouter le certificat et permettre les connexions par protocole TLS/SSL. Dans notre cas, nous n’avons évidemment pas fait signer notre certificat par une autorité de certification, ce qui explique pourquoi le certificat n’est pas reconnu lors de la connexion à l’application.

# Outils utilisés

* Git
* Tomcat
* Servlet
* Eclipse
* Java