SR03

Rapport du Devoir 2

GILLET Guillaume

HERKENS Antoine

26/04/2020

Sommaire

Introduction 3
Cahier des charges 3
Contenu du rapport 3
Bonnes pratiques de sécurité 4
1° Violation de contrôle d'accès et de gestion de session 4
2° Injection Sql, XSS et attaque CSRF 4
3° Chiffrement des données sensibles 5
4° Vulnérabilité d'un composant 5
5° Utilisation de l'application 5
Synthèse : sécurité des applications web 6
L'authentification à deux facteurs : sécurité élevée, implémentation simple. 6
Conclusion 7

Introduction

Cahier des charges

Vous allez mettre en place une petite application web à destination des clients et des employésd'une banque, et implémenter des fonctionnalités basiques que l'on peut trouver dans une telleapplication.On part du principe que c'est le même site qui est utilisé par les différents profils d'utilisateurs :clients de la banque et employés de la banque.La nécessité de sécuriser cette application est assez évidente et les attaques potentielles faciles àimaginer. Les actions malveillantes ou non autorisées peuvent dans notre cas venir de différentessources : un pirate complètement extérieur à l'application, un client de la banque, ou même unemployé de la banque !

Contenu du rapport

Dans un premier temps, nous allons aborder la sécurisation de notre application web. Nous verrons quelles failles nous avons sécurisé et comment nous nous y sommes pris. Dans un deuxième temps, nous ferons une synthèse d'une technique de sécurisation actuelle : l'athentification à deux facteurs.

Bonnes pratiques de sécurité

Dans cette partie, nous allons exposer les bonnes pratiques de sécurité implémentées dans notre projet pour combler les failles de sécurité vues en cours et en TD. (Chaque paragraphe recense une faille et la solution implémentée).

1° Violation de contrôle d'accès et de gestion de session

On vérifie que la session n'est pas vide dans chacune des pages qui nécessitent une authentification pour éviter d'afficher les champs de saisie et des informations confidentielles. Dans le cas ou quelqu'un se connecte aux pages directement par leur lien et on redirige automatiquement vers la page de connexion.

On vérifie aussi que dans la session, l'attribut profil_user est correspond à un employé de banque si l'utilisateur essaye d'accéder aux pages réservées à un employé de banque.

Les liens de redirection sont stockés dans des variables (et non affichés dans la barre de navigation) pour éviter que des scripts de redirection permettent de dérouter l'utilisateur de sa destination initiale.

À l'aide d'un fichier include.php, les cookies sont mis en httponly et secure, ce qui permet de bloquer les scripts qui tenteraient de récupérer ces cookies de connexion pour les exploiter à l'insu de l'utilisateur.

Pour empêcher une attaque par la force brute (essayer toutes les combinaisons de mots de passe/login 1 par 1), nous avons implémenté une variable dans la session qui s'incrémente à chaque fois que l'on se trompe dans un login/mot de passe et qui une fois 5 essais atteints, va bloquer la page login et demander de contacter l'hôte du serveur pour qu'il débloque ce token, ou demander d'attendre que la session expire d'elle-même.

2° Injection Sql, XSS et attaque CSRF

Pour contrer les injections sql, on cré une liste de caractères interdits comme les ' , « ,< ,etc… puis on effectue un str_replace sur les chaines de caractères qui vont être entrées dans les requêtes vers la base de données. Cela permet d'éviter qu'un utilisateur malintentionné essaie de récupérer des données sensibles de la base de donnée. Cette correction permet aussi de bloquer le XSS : les scripts ne peuvent pas s'exécuter puisque nous remplaçons les < et > par du vide. Le format numérique des montants et numéro de compte est lui aussi vérifié.

Pour contrer cette même faille, on effectue des requêtes paramétrées. Cela consiste à préparer un pattern de requête puis de lui affecter les paramètres à partir des variables des différents formulaires correspondant.

Pour l'attaque CSRF, à la connexion à vue_compte.php, on crée un token unique et lors d'un virement on vérifie si ce token est bien initialisé et égal à celui généré précédemment dans la session, auquel cas on fait le virement, autrement on bloque le virement et on renvoie l'utilisateur sur la vue de son compte.

3° Chiffrement des données sensibles

Pour plus de sécurité, nous aurions pû ajouter le chiffrement des données dans la base de données, afin que les données sensibles soient stockées en clair côté SQL et pour éviter qu'ils soient lus en clair dans le php, et avec des fonctions php il est facile d'encoder ou décoder des textes avec un pattern généré, mais nous avons manqué de temps.

4° Vulnérabilité d'un composant

Pour éviter qu'un simple accès au fichier config.ini soit réalisé depuis le navigateur, celui-ci à été remplacé par un fichier config.php qui permet de récolter les mêmes identifiants de la base de donnée mais sans risquer qu'ils soient visible car ce composant est très sensible (phpmyadmin), nous avons donc aussi changé le mot de passe de la base de donnée (cf Utilisation de l'application).

De plus, nous avons ajouté un fichier .htaccess qui bloque tout accès aux fichiers config depuis le navigateur, si un utilisateur entre devoir2/config, l'accès lui sera refusé.

Remarques concernant les bonnes pratiques de sécurité :

Les principales failles notamment celles abordées en cours ont été sécurisées. Il est donc plus difficile pour un utilisateur malintentionné de voler des données sensibles. Avec d'avantage de temps, il serait évidemment possible d'augmenter la sécurité de l'application notamment avec l'authentification à deux facteurs.

5° Utilisation de l'application

- Télécharger le dossier sur gitlab cette adresse
- Mettre le dossier dans un logiciel de serveur en local comme Wamp (ce que nous avons utilisé par exemple)
- Executer le script devoir2.sql (dump de la base de données) dans le répertoire local
- Mettre en route ce logiciel et accéder à localhost/devoir2 (la page index.php sera automatiquement chargée)
- Essayer de se connecter avec un des utilisateurs inclus dans le script devoir2.sql
- Si vous utilisez un compte employé il sera indiqué que vous avez accès aux fiches clients, autrement il n'est rien indiqué
- Veiller au mot de passe du serveur php car nous l'avons modifié pour plus de sécurité vis-à-vis de la vulnérabilité du composant phpmyadmin. Le mot de passe utilisé : 4AfYkjmjPHxtF9en généré automatiquement et hashé par phpmyadmin.

Synthèse : sécurité des applications web

L'authentification à deux facteurs : sécurité élevée, implémentation simple.

L'authentification à 2 facteurs est un moyen simple mais très efficace de sécuriser des applications web, qu'elle soit pour une banque ou un réseau social ou encore une plateforme de jeu vidéo. Ce qui fait la puissance de cette authentification à 2 facteurs (ou A2F) est sa versatilité.

Le principe de fonctionnement est simple, lorsque vous souhaitez vous connecter à votre compte sur le site/l'application web il vous est demandé d'entrer un code unique qui vient d'être généré. Ce codeest reçu dans la majorité des cas sur votre smartphone par sms. Dans certains cas vous pouvez aussi le recevoir par mail. Dès lors, il suffit d'entrer le code reçu dans l'application et vous pouvez accéder à vos données en ayant très peu de chance de vous faire pirater votre compte car le pirate n'a pas accès à tous vos appareils comme votre réception de SMS.Il existe un site qui recense tous les sites/applications qui utilisent cette méthode de sécurisation (https://twofactorauth.org).

Certains sites plus aboutis en terme de sécurité ont amélioré cette méthode (notamment Google et Facebook).Google utilise par exemplele smartphone ou tout autre appareil connecté à votre compte sur lequel une notification est envoyée demandant de valider la connexion du nouvel appareil en indiquant sa localisation est son nom/ip pour assurer de ne pas accepter de connexion malintentionnées.

Il existe aussi des applications dites « authenticator » qui permettent de générer en continu de nouveaux codes pour chaque compte de chaque application et le renouvellement de code en continu permet d'éviter les attaques par essais infinis (le code change tous les certains laps de temps). Ces applications sont plus sures que des sms car elles sont gérées uniquement sur un terminal et les codes contrairement aux sms ne peuvent pas être interceptés par des pirates. Mais ils imposent une contrainte lors de la perte du terminal qui contient l'application, dans ce cas la personne le retrouvant pourrait si elle parvient, a y accéder et utiliser les codes de l'authenticator et entrer dans vos différents comptes.

Une nouvelle méthode d'authentification à deux est apparue. Elle utilise une clé USB externe qui une fois connectée à votre terminal va directement vous connecter de manière sécurisée à votre compte, mais pour cela il faut que l'application soit certifiée FIDO et accepte l'utilisation de cette clé U2F.

Finalement on remarque que l'authentification à 2 facteurs prend aujourd'hui différentes formes mais reste une norme qui se démocratise de plus en plus en sécurité, simple d'utilisation et d'implémentation. Elle réduit drastiquement les risques de piratage et donc l'accès aux données personnelles. Mais elle reste encore assez peu utilisée dans certains domaines et demande d'être activée manuellement lors du paramétrage des comptes utilisateurs, elle n'est pas obligatoire.

Concernantune application de banque on comprend que l'A2F est très importante.Elle est notamment utilisée lors des achats sur le net ou pour confirmer un virement. Par exemple lorsque vous utilisez votre carte bleue sur un site internet muni de cette technologie, il va vous être demandé un code, code envoyé par sms pour bien confirmer que c'est vous qui effectuez l'achat et que vous ne vous êtes pas fait voleé vos données de cartes bancaire.

Lors d'une fuite de données liée au piratage du serveur gérant l'application web, si les données personnelles de personnes n'ayant pas activé cette option fuitent, les pirates peuvent récupérer toute information compromettante à traversl'application. Ce n'est donc pas une méthode de sécurité suffisante, elle doit être combinée avec d'autres méthodes notamment des chiffrages côté serveur pour éviter le vol des données brutes.

Conclusion

Pour conclure, nous avons répondu au cahier des charges demandé dans ce devoir. Nous avons effectivement strucuturé l'application comme convenu et sécurisé les failles abordées dans le cours. Avec d'avantage de temps, il serait possible d'ajouter plus de sécurité à notre application, comme nous l'avons vu par exemple avec la possibilité de l'authentification à deux facteurs.