APP 1 Rapport

Programmation sécurisée

par :

(Nom)	(CIP)
Jean-Christophe Blais-Crépeau	Blaj2133
Ysmaël Fortier	Fory2405

Département de génie informatique
Université de Sherbrooke

13 septembre 2021

Table des matières

1 Diagramme de flux de données

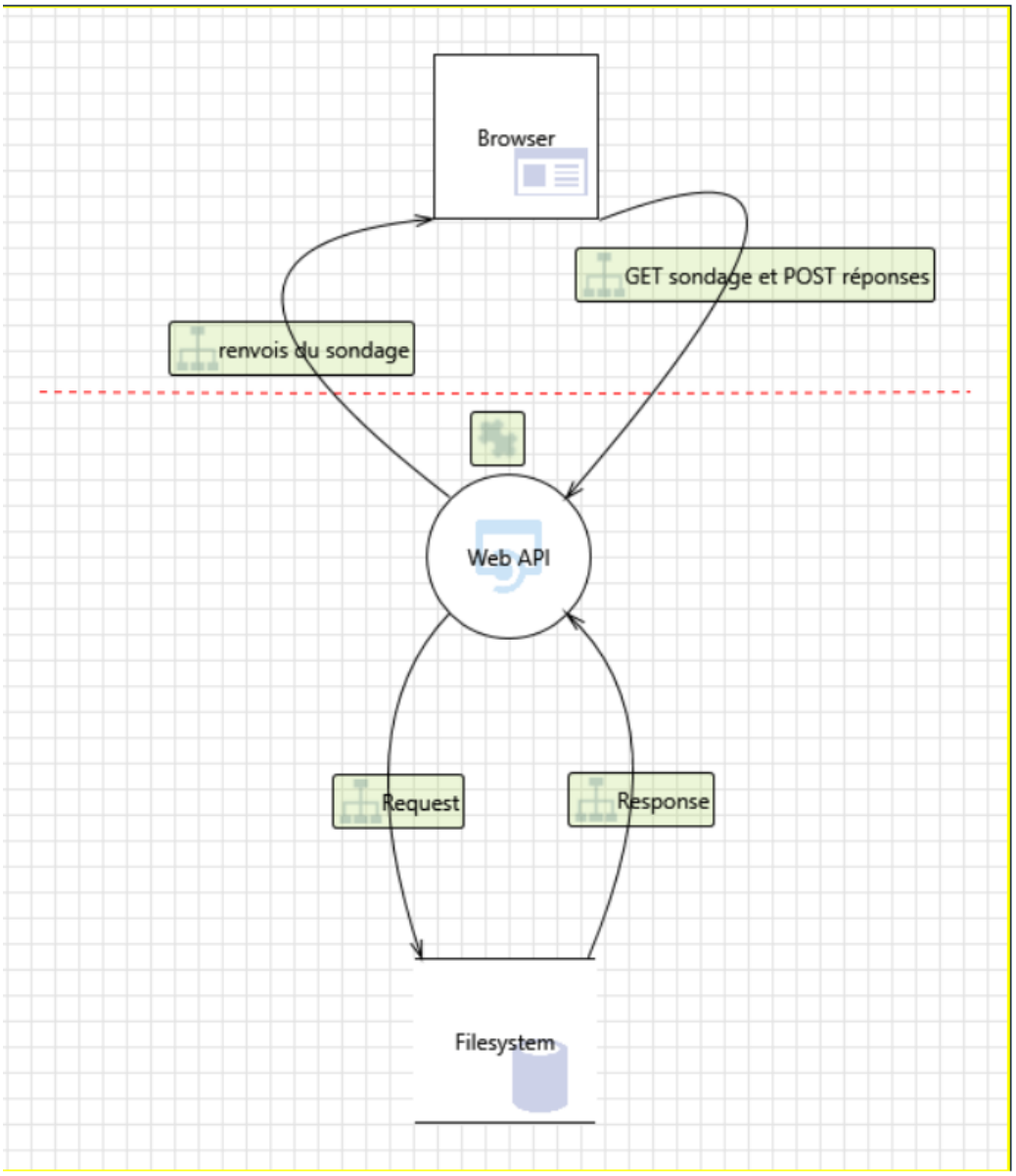

FIGURE 1 – Diagramme de flux de données de l'API

La frontière de confiance se trouve entre l'API et la zone cliente, puisqu'il est assumé que jusqu'au point ou l'API est public (c-à-d. accessible via web), le contrôle de l'appareil et du réseau qui fournis l'API appartiend à celui qui publie l'API. Ceci veut dire qu'il est assumé quue toute vulnérabilité en lien avec comment l'API communique avec les données (qui, dans ce cas, se trouve à être des fichiers) ne sont pas considéré dans le "scope" de l'analyse de la superficie d'attaque.

2 Analyse de la surface d'attaque

2.1 GET - Sondage

Pour cette partie, puisque c'est le client qui intéroge l'API pour avoir de l'information sur le sondage, il est important de s'assurer que l'information qu'il reçoi pendant cette partie ne soit que l'information que l'on veut donner, soit les sondages.
Le vecteur d'attaque est donc par intérogation des chemin d'accès ,P.E. "/../index.php" , ou par vérification de la réponse du système , P.E. créer intentionellement une erreur sur le système pour voir le language de programation, le système d'opération ou deviner comment le service traite les données. Finalement, il y a l'aspect de la répudiatin qu'il faut considérer.

2.1.1 Impact sur l'application

Dans le cas le l'attaque par intérogation de chemin d'accès, ce genre d'attaque consiste à essayer de trouver des pages d'accès qui répondent ou des méconfigurations sur comment le système traite les accès pour gagner accès a des fichiers de configuration ou des accès normalement innaccessible. Il est aussi utilisé pour deviner quel serveur web est utilisé et quel système d'opération est utilisé, ce qui est utile pour trouver des vulnérabilitées sur le serveur qui peuvent être exploité.
Dans le cas de l'attaque par vérification de réponse système, le but est de trouver plus d'information sur le système en regardant comment les erreurs ou trammes dites "illégales" sont traité. Basé sur les codes erreurs reçu par l'attaquant, il est possible de savoir dans quel language de programation et de base de données est utilisé pour fournir l'API. Ceci peut permettre des attaques plus ciblé et de trouvé des vulnérabilitées spécifiques à ceux-ci.
Dans le cas de la répudiation, si les actions d'un attaquant n'est pas enregistré, il est possible que l'attaquant réussi à avoir accès à de l'information sans qu'il y en aye de traces.

2.1.2 Mitigations intégrées

La gestion de code d'erreur de la partie codée de l'API est géré par des systèmes de "Try-Catch" qui redonne des erreurs pré-déterminé à l'utilisateur, ce qui évite qu'un attaquant aye des réponses spécifiques aux erreurs qu'il pourrais poser. Ceci inclus si l'attaquant essaye d'accéder des chemin d'API qui ne sont pas déterminé par notre code. Par contre, du au limitations de la gestion d'erreur de .NET pour les API, il y a certaines erreurs qui ne peuvent pas être intercepté comme les erreurs de mettre du texte ou le système d'API s'attend à avoir un chiffre.
Pour la répudiation, la solution utilisé est de demander un token d'authentificationb valide pour avoir l'accès au sondages. L'utilisation du token d'authentification vas être expliqué en plus de détails dans la section de la réponse au sondage.
Pour plus d'information, voir la partie "Interaction : GET sondage et POST réponses".

2.2 POST - Réponse au sondage

Pour la partie de la réponse au sondage, il est important que l'information reçue soit de l'information qui ne cause pas d'erreur dans le système, que l'information reçue soit bien sécurisé et que les accès aux informations soient restreints le plus possible. De plus, il faut être en mesure d'identifier qui remplis quel sondage, sans que cette information soit identifiable personnellement (IIP). Dans le cas ou il y a de l'IIP, il faut être en mesure d'assurer que cette information soit enregistrée adéquatement.
Le vecteur d'attaque est donc comment l'information se rend sur le système et comment cette information pourrais être extraite.

2.2.1 Impact sur l'application

Si l'information envoyé est mal-formé, est écrite de façon intentionnelle pour causer des erreurs ou est envoyé trop de fois (soit intentionellement, soit par erreur de connection), ceci peut causer des problèmes avec le serveur, l'API ou avec les résultats du sondage.
Dans le case de l'information mal-formé, il est possible que cette malformation cause des problèmes pour la collecte de l'information du sondage ou causer des erreurs dans le système de l'API ou dans le système de fichiers de données (par exemple dans une base de données) ou dans le système utilisé pour sortir et utiliser les données stoquées.
Un problême similaire apparait quand l'erreur dans la tramme de réponse est créé intentionellement. Ceci peut même causer la perte de données (XKCD Jimmy " ;DROP table ;") ou des erreurs que l'attaquant peut exploiter pour avoir plus d'informations sur le système.
Si l'information est envoyé trop de fois, il se peut que les données qui sont collecté dans le sondage soient fausse ou donne de fausse conclusions. Si le système était un API pour des achats, un scénario catastrophique possible serait que la personne achête la même chose plusieurs fois par inadvertence (insert link to UK dinedash incident here).

2.2.2 Mitigations intégrées

Les messages mal-formées ou vides sont géré par un système de "try-catch" similaire à ce qui est utilisé dans la section des sondages. Pour les informations envoyées légitimes mais plusieurs fois, la gestion de ceci est fait avec la combinaison d'un Token d'authentification et d'un index spécifique à la réponse fournis. Pour tout accès à l'API, il faut avoir un token valide pour avoir une réponse valide. Quand l'utilisateur envoie la réponse de son sondage, le token d'authentification est à la fois utilisé dans la connection API pour authentifier l'utilisateur, mais aussi dans la réponse au sondage. Ceci est ce qui est utilisé pour identifier uniquement le répondeur du sondage. Dans la réponse, il y a aussi un index qui est utilisé. Cet index est unique à la réponse envoyé, ce qui permet un suivi des copies envoyé par l'utilisateur. Si une réponse similaire (et donc avec le même index) est envoyé plusieurs fois, seule la première copie est accepté. Si par contre la même réponse est envoyé avec des index différent, il est plus facile d'en faire le suivi puisque toutes ces réponses utilisent le même token d'authentification.
Pour plus d'information, voir la partie "Interaction : GET sondage et POST réponses".

2.3 Accès au fichiers de données

Pour ce qui est de la protection des données stockées, le but est d'empêcher la lecture des données par un acteur malveillant et de se rendre à une lecture potentielle le plus difficilement possible.

Pour empêcher une personne non-autorisée de lire nos données, une encryption à fonction réversible comme RSA-1024 était visé avec une variable d'environnement contenant la clé d'encryption/décryption afin de permettre au code de lire le fichier, le décrypter, en lire son contenu, le modifier, le ré-encrypter et finalement l'enregistrer. La valeur de la variable d'environnement serait injectée à partir d'un secret lors du déploiement de l'application. L'ensemble de ce processus était imbriqué avec la mitigation du problème lié à la diminution de l'aire d'attaque par l'utilisation d'un conteneur "alpine" dotnet officiel de Microsoft. Tout ce processus permet de diminuer le nombre de services exécutés sur la même image et aussi de s'assurer de facilement être capable de garder le système d'opération à jour. Dans le conteneur, seul le port HTTPS serait exposé à l'externe. Maintenant, pourquoi ça n'a pas été implanté dans notre code ; la solution de code construite fonctionnelle sur windows avec obfuscation devenait non-fonctionnelle lorsqu'elle était porté sur linux-musl-x64. Si nous avions plus de temps pour investiguer le problème, nous concentrions notre effort sur cette facette car elle apporte beaucoup de valeur en sécurité et en pérennité de notre code.

Références

[1] Guy Lépine (B.Sc.A) , Guide de l'étudiant - GEI-771 - Programmation sécurisée
`https://www.gel.usherbrooke.ca/s8gsa/a21/doc/app1/file/gei771.pdf`