Stratégie de sécurisation du site pire2pire

Vasseur Bastien

Introduction

Contexte

The goal of this documentation is to present the security measures we will take to ensure the security of pire2pire.com. Today, the dangers of cyber threats are omnipresent, and we must ensure that security measures are taken at every step of the project, starting with this document, through the conception of the site and its functionalities, to the development and release.

The goal of this document is to be understandable for everyone, from beginners to experts. This means it will be presented in a pedagogical way. We will focus on the different dangers, present and explain potential solutions to those dangers.

Pire2pire.com is an e-learning website where people will be able to learn different courses. There will be visitors, students, and coaches.

Impact

A poorly secured website can cause a lot of harm. Here is a list of dangers that may include:

- Sensitive information theft: Malicious actors may gain access to sensitive data, such as financial information, a corporation's confidential information, or an individual's personal information. The danger is that the attacker may use this stolen data for identity theft, blackmail, or fraud.

- Damage to brand : A website can be defaced, which means inappropriate content can be inserted, political messages can be inserted,

advertisements for products can be inserted, or a warning that the website has been hacked. All of these will damage the trust of current and potential customers. In case of a successful attack, it is strongly advised to inform the customer as quickly as possible so that customers can protect themselves and be more careful against potential identity theft or fraud.

- Criminal activity : Gaining customer information or taking control of the website can lead to criminal activities like spreading malware, posting illegal content, conducting phishing, or using the platform to attack other systems or people who have access to it.

- Banning or suspension: Content inserted on the website may be suspicious, either spam, malware, or links to distrustful websites. These can be flagged as dangerous or suspicious by search engines. These flags can lead the hacked website to be flagged as suspicious, which will impact search engine results (up to the removal of the website from results) and affect traffic to the website. In the worst-case scenario, the host of the site may shut down the website.

- Harm to the user experience: One of the side effects of being hacked may be activities or code that slow the response time to requests, which will ruin the user experience. Adding to that problem, there is the possibility of spam and malware also affecting the user experience negatively, which may lead users to leave the website.

- Financial losses: Access to financial data may result in financial loss, especially if the hacker gains enough access to perform transactions. The loss of customers may also impact revenue. There is also the potential for compensation to customers and the cost of dealing with and repairing the security breach.

Organization of the documentation

This document will present the different types of dangers that may occur during the lifetime of the site. We will then explain how these problems will be stopped. Finally, we will detail all the protections in each section of the documentation, from the database to the front.

Information Général

RGPD

Le Règlement Général sur la Protection des Données (RGPD) est une réglementation de l'Union européenne que le site devra suivre et appliquer. Il se concentre sur le traitement des données personnelles, la collecte, le stockage, le partage et donne plus de contrôle aux utilisateurs.

Nous devrons donc :

- Consentement : Demander le consentement clair et explicite de l'utilisateur avant toute collecte de données.

- Droits : L'utilisateur a le droit d'accéder aux informations récoltées, de les rectifier, le droit à la portabilité, le droit d'opposition au traitement des données ainsi que le droit à l'oubli.

- Responsabilité : Nous avons l'obligation de protéger ces données, ce dossier étant une explication des mesures techniques que nous mettrons en place.

- Notification : Toute violation de données doit être signalée aux autorités dans les 72 heures suivant la détection.

En cas de non-conformité avec la réglementation, l'entreprise peut recevoir une amende allant jusqu'à 20 millions d'euros ou 4 % du chiffre d'affaires mondial.

Type d'attaques rencontrées

Ici, nous allons présenter les différents types d’attaques que ce guide a pour but de prévenir :

- XSS : L'attaque (Cross-Site Scripting) consiste à injecter du code (JavaScript ou HTML) dans une page web pour obtenir un comportement visé lorsque ce code sera exécuté. Le but est de récupérer des informations sur les utilisateurs d'un site.

- CSRF : L'attaque (Cross-Site Request Forgery) consiste à tromper un utilisateur pour le forcer à effectuer une action voulue sur un site donné depuis un site tiers. Par exemple, envoyer une requête cachée à `pire2pire.com` en interagissant sur un site créé par l'attaquant.

- SSRF : L'attaque *Server-Side Request Forgery* consiste à demander au serveur d'effectuer des requêtes vers une destination choisie par l'attaquant, en profitant éventuellement des privilèges du serveur.

- SQLi : L'attaque *SQL injection* consiste à envoyer une requête non prévue par les fonctions du site ou les droits d'utilisateur pour affecter la base de données. Cela donne la possibilité à l'attaquant de faire ce qu'il veut.

- LFI/RFI : L'attaque *Local/Remote File Inclusion* consiste à trouver un moyen d'accéder à un fichier local ou distant sur le site normalement pas accessible pour une personne n'ayant pas les droits.

- DDoS : L'attaque *Distributed Denial of Service* consiste à envoyer un grand nombre de demandes automatisées au serveur pour le surcharger et

perturber l'utilisation des autres utilisateurs, en ralentissant voire en rendant le serveur incapable de répondre aux demandes.

- XXE : L'attaque *XML External Entity* consiste à injecter du XML malveillant pour exécuter ce code et obtenir des accès normalement impossibles.

Pratique de développement

Pour protéger de ces attaques, certaines mesures vont être respectées à travers le développement du projet.

Unités distinctes

Chaque couche de l'application sera séparée (Front, Base de données, API). Cette séparation nous permettra de bien définir les interactions entre chacune d'entre elles. Ces interactions étant bien définies, nous allons pouvoir développer des mécanismes de défense et de vérification pour chacune d'entre elles.

Moindre privilège

Le moindre privilège consiste à donner le moins de possibilités d'action possible à chaque rôle des utilisateurs. Les visiteurs du site ne doivent pas avoir accès à des fonctions dédiées aux étudiants, coachs ou administrateurs.

Réduction de la surface d'attaque

Plus nous ajoutons de composants au site et plus nous agrandissons l'exposition aux réseaux de notre site. Plus il y a d'opportunités de se faire attaquer. Nous n'allons pas rajouter des composants inutilisés ou non

sécurisés, ainsi que retirer ceux qui sont seulement utiles au cours de la phase de développement lorsque nous passerons à la phase de déploiement. De même pour l'exposition réseau : plus on en a, plus il y a de vulnérabilités. Nous devons limiter les accès réseau.

Sécurisation des échanges de données

Tout échange de données doit être sécurisé pour ne pas être intercepté pendant son transfert. Nous allons utiliser des protocoles comme le protocole HTTPS pour la protection.

Conformité du contenu

Nous allons assurer la conformité des données pour faire en sorte que le site apparaisse dans le navigateur des utilisateurs tel qu'il a été conçu, afin d'assurer l'expérience promise.

Journalisation

La journalisation consiste à enregistrer les interactions sur chaque couche du site dans des journaux d'événements pour garder une trace de tout ce qui s'y passe ainsi que du moment où ces événements se sont produits. Le but étant que, en cas de problème, d'attaque ou de tentative d'attaque, on puisse retracer l'attaque pour corriger de potentielles failles ainsi qu'avoir une idée de l'ampleur des dégâts.

Test unitaire

Chaque composant de l'application devra être testé avec une série de tests simulant, en plus de la détection de bugs, les tentatives d'attaque, dans le but de détecter d'éventuelles faiblesses de sécurité.

Une série de tests d'automatisation visant la partie mise à jour sera mise en place pour tester chaque changement apporté au projet. Ainsi,

nous pourrons découvrir et corriger les éventuels problèmes de sécurité dès qu'ils sont identifiés.

La Base de données

La base de données est au cœur de la sécurité du site, c'est l'endroit où toutes les informations importantes se trouvent, et les protections de la base de données sont extrêmement importantes au cas où les protections des couches rencontrées avant ont été contournées ou si l'attaquant arrive à attaquer directement la base de données.

Authentification

Utiliser un système d'authentification est important pour protéger une base de données. Ce système permet de créer des rôles qui gèrent les droits sur la base de données, cela permet de gérer ce qui peut être vu, écrit, supprimé ou mis à jour.

Forcer une identification permet de bloquer les requêtes où la personne qui l'envoie n'a pas moyen d'accéder aux identifiants. Une gestion des rôles est à faire, avec tout rôle inutile devant être supprimé. Le principe du moindre privilège doit être utilisé, par exemple ne pas donner un droit d'écriture sur des cours lorsqu'il a été décidé que le rôle étudiant n'a que le droit de lire ces informations.

Les rôles demandant un identifiant et un mot de passe pour se connecter, les bonnes pratiques d'utilisation de mot de passe doivent être appliquées, de la complexité du mot de passe au hachage.

UUID

UUID est l'Universel Unique Identifier. Contrairement au UID qui utilise l'incrémentation pour se différencier, le UUID est utile pour éviter qu'un hacker puisse, au hasard, trouver un ID lors de son attaque.

L'UUID se présente sous la forme de 36 chiffres hexadécimaux, ce qui rend le nombre énorme avec un nombre de variantes paraissant presque infini.

Si l'attaquant se retrouve en position où il doit trouver un ID pour son attaque, que ce soit pour voler des données ou tenter une attaque LFI par l'URL, l'UUID permet de rendre cette tâche presque impossible.

Chiffrement

Le chiffrement est le fait de transformer les données pour qu'elles soient illisibles à toute personne qui arrive à voler les données.

Elle fonctionne avec une clé qui permet de décrypter les données quand les vérifications montrent que la demande est légitime.

Bien sûr, cette clé demande alors de ne pas être facilement accessible. On utilise un système de gestion de clés pour qu'elle soit dans un endroit sécurisé.

Les clés de chiffrement

Dans un système de gestion de clés, des bonnes pratiques pour l'accès doivent être utilisées. Le moindre privilège, la clé doit être utilisée pendant un certain temps, un délai de 90 jours sera appliqué. Nous allons donc utiliser ce délai pour pire2pire.com. Au bout de ces 90 jours, une rotation de la clé de chiffrement du site devrait être faite pour le déchiffrage des données et les chiffrer de nouveau avec la nouvelle clé.

Le cycle de vie d'une clé :

- Génération : la clé est générée avec l'algorithme de chiffrement.

- Distribution : la distribution de la clé doit être faite de manière sécurisée via une connexion TLS.

- Utilisation : elle est utilisée pour encrypter toutes les données précédemment décryptées par l'ancienne clé.

- Stockage : on la stocke ensuite dans notre gestionnaire de clés.

- Rotation : la clé étant à la fin de sa vie, la rotation commence, on déchiffre toutes les données pour la clé suivante.

- Révocation : tous les droits d'utilisation de la clé sont révoqués et elle devient inutilisable.

- Destruction : la clé est officiellement détruite.

ORM

ORM (*Object-Relational Mapper*) fournit une couche orientée objet entre la base de données et le code. Il permet de faire en sorte que l'on n'envoie pas la requête reçue directement dans la base de données.

Il permet de configurer des points d'entrée vers la base de données plutôt que d'envoyer directement une requête SQL. On utilise les fonctionnalités de l'ORM pour la créer pour nous.

Ainsi, seules des requêtes prédéterminées peuvent aller jusqu'à la base de données par les autres couches de l'application, empêchant ainsi des attaques SQLi.

Le tout crée une couche protectrice autour de la base de données, limitant les types d'accès à la base de données.

Site de sauvegarde

Il faut avoir un site de sauvegarde sécurisé pour la base de données. Un site non sécurisé peut se faire attaquer par une personne directement.

L'attaquant aura alors accès à la base de données directement et seules les mesures d'authentification et d'encryptage peuvent protéger la base de données.

Copie de la base de données

En cas de destruction du lieu physique de la base de données ou d'une attaque détruisant des données, des copies de la base de données sont nécessaires.

Les copies devront se trouver dans un lieu différent de la base de données principale. Pire2pire.com étant un site de e-learning, la base de données et la mise à jour fréquente de la base de données demandent que celle-ci soit copiée plusieurs fois par jour.

Audit

L'audit de la base de données est un journal de toutes les activités qui se passent sur la base de données.

Il permet de se rendre compte de toutes les actions qui se passent dans la base. Il garde toutes les actions qui se passent, que ces actions soient réussies ou non, ainsi que l'heure de ces actions.

Cela permet de voir les tentatives qui se passent sur la base de données et surtout si ces attaques ont réussi, afin que l'on puisse examiner l'ampleur des dégâts et ainsi pouvoir prévenir et au mieux y remédier.

Le Back

Le back-end d'un site web est l'endroit où toutes les informations transitent, du front qui est vu par tous les utilisateurs à la base de données qui contient toutes les informations importantes que nous voulons protéger. Cela veut dire que c'est l'endroit où l'on va pouvoir faire le maximum de vérification.

Vérification des entrées

Toutes les demandes envoyées par un utilisateur doivent être vérifiées. Par exemple, dans la barre de recherche de pire2pire, pour trouver une formation, on doit vérifier s'il n'y a pas de code HTML, JS ou des requêtes SQL dans la demande envoyée.

Dans le cas de l'accès à une autre page, on doit vérifier si l'utilisateur a le droit d'accès à cette page, avec l'authentification et les jetons JWT. Le but étant d'éviter qu'une personne, comme un visiteur, puisse faire des dégâts sur le site en accédant à des endroits où il n'a pas le droit.

Si l'utilisateur n'a pas le droit et arrive à trouver la page juste en cherchant le chemin, ceci est l'attaque LFI dont on a parlé plus tôt, et la vérification des droits est le moyen de la bloquer.

On peut noter que cela inclut aussi le fait d'essayer de trouver des fichiers qui ne sont pas des pages à afficher, mais juste des fichiers utilisés pour le bon déroulement de l'application. Dans ce cas-là, nous allons bloquer tout caractère qui montre qu'on essaie de naviguer dans l'arborescence des fichiers.

La vérification se fait aussi par l'origine des requêtes. Pire2pire n'étant pas une API ouverte, les requêtes devront être vérifiées dans l'en-tête HTTP pour s'assurer qu'elles viennent bien de pire2pire.com ou d'une autre source autorisée, par exemple les informations de transaction bancaire.

Les entrées sont aussi celles venant d'autres sources que le front, que ce soit la base de données, la récupération de la clé d'encryption ou encore une autre API.

Dans le cas de la base de données, il faut vérifier si les données reçues correspondent au format attendu de la requête envoyée.

Le but de toutes ces vérifications est de limiter le nombre d'accès à l'API pour pouvoir surveiller et contrôler ce qui se passe.

Traitement des données et sortie des données

Toutes les données qui passent dans l'API doivent être traitées après vérification pour leur sortie.

Dans le cas des requêtes vers la base de données, par exemple la création d'un nouveau compte, les données doivent être vérifiées, préparées et encryptées avant d'être envoyées à la base de données. Dans le cas des mots de passe, il faudra les hacher en plus.

Ensuite, tout est envoyé à l'ORM pour qu'il fasse lui-même la requête vers la base de données.

Le cas inverse se passe aussi pour ce que l'on a reçu de l'ORM : on traite les données reçues, on les déchiffre et on les formate pour les rendre conformes aux besoins du front, le but étant d'éviter le traitement de données côté client.

Mot de passe

Le mot de passe est important pour l'authentification et la sécurité du site, car c'est une étape importante pour déterminer le rôle de l'utilisateur et à quelles données il a droit d'accès. Il est donc primordial de le protéger.

Menaces

Il y a plusieurs types de menaces sur les mots de passe :

- Attaques en ligne : Le hacker essaie de passer outre les défenses du système pour avoir accès aux identifiants.

- Attaque par force brute : Le hacker essaie des mots de passe au hasard, potentiellement venant de dictionnaires de mots de passe avec des tests automatisés.

- Attaque par protocole d'authentification : Le hacker intercepte des données d'identification par un transfert non sécurisé ou vole un token pour se faire passer pour l'utilisateur.

- Vol du moyen d’authentification : Le mot de passe est volé, soit par un bout de papier, par hameçonnage, etc.

Sécurité du mot de passe

Dans le cas de pire2pire.com, les facteurs d'authentification seront des facteurs de connaissance.

Nous devons protéger le mot de passe de la meilleure façon possible. Il y a plusieurs mesures que l'on peut mettre en place :

- La complexité du mot de passe : Le mot de passe ne doit pas être trop simple. Il doit être assez long et avoir des caractères assez diversifiés pour se protéger contre les attaques par force brute. Il est donc conseillé de mettre une taille minimum de 10 caractères avec au moins une majuscule, une minuscule, un chiffre et un caractère spécial.

- Le hachage : Avant d'être encrypté, le mot de passe doit être masqué en le hachant avec un algorithme de hachage comme Bcrypt. On y rajoute un sel pour assurer la sécurité et protéger contre les attaques de rainbow table.

- Limitation du nombre de tentatives : Quand un nombre de 10 tentatives a été essayé, l'utilisateur sera bloqué pour une dizaine de minutes, avec comme seul recours le changement de mot de passe.

- Détection des motifs simples : 12345, date d'anniversaire, etc.

Note : On ne dira jamais lors d'un échec si l'email ou le mot de passe est faux.

Changement de mot de passe

Le changement de mot de passe est une étape nécessaire dans toute application pour éviter de perdre des utilisateurs qui ont perdu l'accès à leur compte.

Le changement de mot de passe se fait en demandant l'email de l'utilisateur. On ne confirme pas si un compte correspond à l'email ou non pour éviter qu'un hacker puisse faire une liste de comptes existants sur le site.

Un token sera généré avec une durée de 15 minutes. Il sera envoyé dans un lien par email à l'utilisateur.

L'utilisateur pourra cliquer sur le lien pour choisir un nouveau mot de passe.

Restriction des derniers mots de passe

On garde en mémoire le hachage des 5 derniers mots de passe pour dire à l'utilisateur qu'ils ne peuvent pas être réutilisés. Cela sert en cas d'attaque pour éviter qu'un utilisateur réutilise un mot de passe qu'il a l'habitude de changer, et qui pourrait être compromis.

Pour les rôles plus importants

Dans le cas des employés de pire2pire qui ont accès à des informations sensibles, on préconise une authentification plus forte, avec un facteur de possession. Après l'entrée du mot de passe confirmé, un

SMS contenant un code sera envoyé. L'employé devra entrer ce code pour se connecter.

Une autre sécurité que l'on peut mettre en place est une durée de vie limitée pour un mot de passe. La personne devra changer son mot de passe tous les mois.

JWT

Le JWT est un token généré lors de l'identification. Il est envoyé côté client de manière sécurisée pour l'identification. Il est conservé de manière sécurisée côté client et est envoyé à chaque fois qu'un appel à l'API est effectué. Il permet de prouver que le client est bien lui-même et d'éviter des attaques CSRF.

On mettra une durée d'expiration de 10 heures sur ce token, ce qui forcera les utilisateurs à se reconnecter le lendemain.

WAF

Le WAF (Web Application Firewall) se situe entre le client et l'API et protège l'API des attaques. Il permet de protéger contre les attaques SQLi, XSS et les attaques de type DDoS, et d'analyser les requêtes HTTPS.

On peut le configurer pour qu'il laisse passer uniquement les requêtes voulues et qu'il bloque les requêtes malveillantes.

Le Front

Le front est le côté client, c'est ce qui se passe sur la machine du client. Le client peut donc avoir accès à toute information qui est envoyée. Nous devons donc protéger toutes les informations qui y arrivent pour empêcher qu'une autre personne que le hacker puisse y avoir accès.

TLS

Le Transport Layer Security (TLS) est un protocole de sécurité utilisé pour chiffrer les communications entre les applications sur Internet.

On l'utilisera avec le protocole https pour sécuriser le transfert des données, le but étant d'empêcher un potentiel hacker de capter les informations par des attaques de type Man-in-the-Middle.

On utilisera la dernière version de TLS (1.3) pour la sécurité.

On utilisera aussi les mesures de HSTS pour forcer tout accès en HTTPS et aucun en HTTP.

On utilisera aussi les certificats TLS pour indiquer que le site est sûr. Ils seront surveillés par le Certificate Transparency, le but des certificats étant de prouver que le site est reconnu comme sûr et ne pas avoir de problèmes avec les navigateurs et le référencement.

SOP

Le SOP (Same-Origin Policy) est une règle utilisée par défaut pour protéger où arrivent les données.

Le but étant que tout transfert vers un autre site non autorisé ne soit pas possible et que les informations sensibles, comme les cookies et les identifiants, ne soient pas lisibles par une autre origine. Cette mesure peut protéger contre les attaques XSS et CSRF.

CORS

Le Cross-Origin Resource Sharing (CORS), on l'utilisera pour contourner le SOP quand on aura besoin d'accéder à d'autres origines que la nôtre, comme par exemple pour des vidéos YouTube créées par **pire2pire.com**.

Le CORS permet aussi d'utiliser les prévols, ou une vérification que la demande est conforme et autorisée, avant son exécution.

CSP

Le CSP (Content Security Policy) est une politique de sécurité que nous allons utiliser en complément de CORS pour contrôler l'activité venant d'autres origines. Si CORS consiste à dire avec qui on peut communiquer, CSP est là pour dire avec qui on ne peut pas communiquer.

Le but du CSP est de dire, dans une en-tête, ce que le navigateur doit laisser passer et bloquer comme ressource venant d'une autre origine. Le

but étant d'empêcher le site d'exécuter un script venant d'une source que l'on n'approuve pas, cela arrête les attaques **XSS**.

CSP peut aussi bloquer les connexions vers d'autres domaines non autorisés, et donc éviter que l'utilisateur n'active une requête silencieuse à son insu.

CSP peut aussi empêcher le clickjacking, qui est le fait qu'un attaquant place un filtre invisible devant une page du site.

La dernière fonctionnalité de CSP est qu'il peut garder un journal des tentatives de contournement, ce qui nous permet de voir les potentielles attaques et comment les hackers attaquent le site.

Le CSP peut également bloquer les origines d'exécution du JavaScript, en autorisant seulement dans un fichier dissocié, ce qui permet d'empêcher des attaques XSS.

Toutes ces fonctionnalités seront utilisées pour le site pire2pire.com.

Referrer Policy

Le Referrer Policy est une politique de sécurité qui nous permet de contrôler l'URL qui passe dans l'en-tête quand un utilisateur clique sur un lien.

Cela nous permet de, par exemple, voir de quelle page vient l'utilisateur. Dans le cadre de pire2pire, cela va être utilisé comme mesure de protection. pire2pire étant une plateforme de e-learning, il est possible que

des liens frauduleux soient postés ou envoyés à quelqu'un. Pour éviter que toutes les informations qui se trouvent dans l'URL soient envoyées, on peut utiliser la Referrer Policy pour indiquer quoi mettre dans l'en-tête HTTP envoyé : de l'URL entière à seulement le nom de domaine, voire rien.

Dans le cas de pire2pire, on enverra seulement le domaine vers d'autres origines.

Stockage de données côté client

Il y a plusieurs moyens de stocker des données côté utilisateur :

- WebStorage, IndexedDB, et Cookies sont ces différents moyens.

Le WebStorage est accessible au JavaScript, et les données sont stockées en clair, ce qui le rend susceptible aux attaques **XSS**.

L'IndexedDB stocke en clair directement les données de l'utilisateur. Cela peut être utilisé par des **Web Workers**, ce qui le rend aussi susceptible aux attaques **XSS**.

Les cookies, par contre, ont des options de protection. On peut limiter leur accès, les empêcher d'être transmis vers d'autres domaines et leur donner une date d'expiration.

Les cookies sont donc la solution que nous allons utiliser pour l'authentification, en gardant l'identité de la personne ainsi que son JWT token, en utilisant toutes les options de sécurité et une date d'expiration.

Sécurité des informations envoyées depuis le client

L'utilisateur va envoyer des informations à travers des formulaires. Ces informations devront être vérifiées pour voir si elles sont conformes à ce qu'on s'attend.

En plus de ces vérifications, on doit vérifier que les données ne soient pas dangereuses. Les données envoyées peuvent contenir du code qui va s'exécuter. Nous allons donc échapper les entrées utilisateur pour éviter toute exécution de code ainsi que ne pas utiliser de fonction qui pourrait potentiellement exécuter ce code, comme eval. Nous n'allons jamais insérer du contenu dans le DOM sans traitement et vérification avant.

L'envoi d'une requête peut être une requête détournée par un site piégé envoyé à l'utilisateur. Pour éviter cela, nous allons utiliser des tokens CSRF. Le token sera normalement généré quand l'utilisateur arrive sur la page, il sera conservé dans la session et mis en valeur cachée dans le formulaire. On pourra les comparer lors de l'envoi du formulaire pour voir si c'est vraiment une requête venant de l'utilisateur utilisant la méthode normale ou une attaque **CSRF** pour manipuler les informations de l'utilisateur.

Appel à l'API

Tous les appels vers une API seront faits avec Fetch, XMLHttpRequest étant plus sensible aux attaques XSS.

Fetch a un meilleur support avec le CSP et le CORS, mais aussi la possibilité d'interagir avec les cookies de crédentiels.

Nous utiliserons donc la méthode Fetch pour le site.

Iframe

Dans le site, l'incrustation de vidéos YouTube peut être utilisée pour un autre support aux étudiants.

Nous pouvons isoler ces vidéos YouTube pour que le domaine YouTube n'ait aucun accès à ce qui se passe, cela permettra d'empêcher le deuxième domaine que l'on fait apparaître sur la page de soumettre des formulaires, d'exécuter du code, d'accéder aux informations protégées par le SOP ou d'accéder au DOM.

Ainsi, on peut utiliser les avantages de fonctionnalités d'autres sites sans mettre en danger la sécurité de **pire2pire.com**.

Protection contre les injections de codes

Strict Mode

Toute utilisation de JavaScript sera faite avec l'activation du strict mode. Le strict mode force les meilleures pratiques de codage. Avec ce mode, des simples erreurs lors de l'exécution du code bloqueront son avancement.

L'ajout de ce mode peut bloquer le code injecté lors d'une attaque XSS.

Web Worker

On peut utiliser les Web Workers pour l'exécution du code JavaScript dans un environnement séparé, sans qu'il ne puisse affecter directement le DOM. Cela nous permet de faire de l'exécution de code JS avec plus de sécurité.

Nous devons quand même faire des vérifications sur ce qui entre dans le Web Worker, ainsi que sur ce qui en sort avant de l'utiliser dans le DOM.

Le CSP doit être utilisé avec le Web Worker pour limiter les sources externes.

Template String

Lorsque l'on reçoit des informations, on peut utiliser les fonctions d'échappement des template strings pour remplacer les caractères qui indiquent des fonctionnalités de code. Le navigateur comprendra qu'il devra afficher les caractères initiaux lors de l'affichage, mais ils ne seront pas reconnus comme du code lors de l'exécution.

Conclusion

Nous allons mettre en place toutes ces mesures dans le but de respecter la réglementation, mais aussi éviter toute forme de dégât sur pire2pire et donner la meilleure expérience utilisateur à toutes les personnes qui interagiront avec le site, que ce soit des visiteurs, des étudiants ou bien encore des employés de pire2pire.

Source

anssi guide recommandations mise en oeuvre site web maitriser standards securite cote navigateur

https://cyber.gouv.fr/sites/default/files/2013/05/anssi-guide-recommandations_mise_en_oeuvre_site_web_maitriser_standards_securite_cote_navigateur-v2.0.pdf

anssi guide authentification multifacteur et mots de passe

https://cyber.gouv.fr/sites/default/files/2021/10/anssi-guide-authentification_multifacteur_et_mots_de_passe.pdf

OWASP Cheat sheet Series

https://cheatsheetseries.owasp.org/index.html

Recommandations pour la sécurisation des sites web

https://cyber.gouv.fr/sites/default/files/IMG/pdf/NP_Securite_Web_NoteTech.pdf

guide pratique RGPD

https://www.cnil.fr/sites/cnil/files/2024-03/cnil_guide_securite_personnelle_2024.pdf

Authetification SQL Server

https://www.datasunrise.com/fr/centre-de-connaissances/authentification-sql-server/?utm_source=chatgpt.com

Consequeces de la sécurité

https://www.teamcolab.com/insights/the-consequences-of-compromised-website-security-and-how-to-protect-your-company/

UUID

https://laconsole.dev/blog/comprendre-uuid

Clé

https://www-encryptionconsulting-com.translate.goog/education-center/what-is-key-management/?_x_tr_sl=en&_x_tr_tl=fr&_x_tr_hl=fr&_x_tr_pto=rq

Podcast

https://youtu.be/zVu-KEO8hps?si=neskin2KKLl-ZP1G

MDM web docs

https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides