

BLOC 3 – CYBERSÉCURITÉ : SÉCURISATION DE L'APPLICATION VÉTOPLAN – PRISE EN COMPTE DE LA FAILLE CSRF

CONTEXTE VETOPLAN

L'éditeur logiciel **Medinfo** spécialisé dans les solutions dédiées au milieu médical propose des solutions logicielles en abonnement SAAS. Dans ce cadre ils hébergent eux-mêmes les applications métiers auxquelles ont accès les clients grâce à des formules d'abonnements.

Un pack de solutions est vendu aux cabinets vétérinaires pour

- la gestion des clients,
- la gestion des interventions,
- la gestion des consultations,
- la prise de rendez-vous et
- les visites chez les clients/patients.

Ce pack est un ensemble de solutions logicielles interconnectées qui ont été développées et améliorées depuis leur conception initiale.

Dans cet ensemble de logiciels, l'application **VetoPlan** qui permet de gérer les rendez-vous dans un navigateur sur PC ou sur téléphone mobile n'a pas été mis à jour depuis de nombreuses années car la sécurisation d'autres applications était prioritaires.

Pour autant l'application VetoPlan fait l'objet de nombreux actes malveillants :

- Les données de la base sont souvent corrompues,
- certains vétérinaires accèdent à des rendez-vous qu'ils n'ont pas créé,
- ils soupçonnent aussi que leur base client a pu être dérobée.

Actuellement une équipe complète est dédiée à la correction des valeurs directement dans la base de données pour pallier au plus vite aux actes malveillants mais le cœur du problème n'est pas réglé.

Suite à un incident où un hacker a réussi à modifier le mot de passe d'un utilisateur en lui envoyant un lien sur son mail professionnel, l'entreprise Medinfo a fait appel à une société experte en sécurité pour auditer l'application VetoPlan.

Cet audit de sécurité a mis en évidence des défauts de conception et d'implémentation.

Vous intégrez l'équipe qui a pour rôle de corriger les problèmes de sécurité dans l'application VetoPlan.

L'organisation ne souhaite pas changer d'architecture applicative pour le moment. De nouvelles technologies sont à l'étude mais ne seront utilisées que pour la prochaine version majeure. Vous devez donc corriger les problèmes identifiés directement dans l'application existante même si les technologies employées ne sont plus d'actualité.

Le Product Owner a défini avec le client et la société d'audit les axes de sécurisation prioritaires à mettre en œuvre.

Le premier sprint concerne la correction d'une faille de sécurité présente dans plusieurs fonctionnalités de l'application et illustrée par le scénario de risque 6.

SPRINT 1 – CORRECTION D'UNE FAIBLE DANS LE CONTRÔLEUR DE CRÉATION D'UN RENDEZ-VOUS.

Exercice 1 : Analyse de la faille CSRF dans l'application VetoPlan

A l'aide de l'application VetoPlan installée sur le serveur web de la section (etu.btssiobayonne.fr) et accessible dans Netbeans.

Question 1.1 diagramme de classe

À l'aide du script SQL de création de tables, schématiser la structure de la base de données de l'application sous forme d'un MEA ou d'un diagramme de classe.

Question 1.2 Impact du scénario 6

Consulter le scénario de risque n°6 et indiquer quel est son impact sur le contenu de la base de données de l'application VetoPlan.

La société d'audit vous a donné un fichier permettant de tester la vulnérabilité présentée dans le scénario de risque n°6.

Mise en œuvre du test :

Etape a

Accéder à l'application et se connecter en tant qu'Amal Hecker.

- login : amal.hecker@vetoplan.fr
- mot de passe : sio

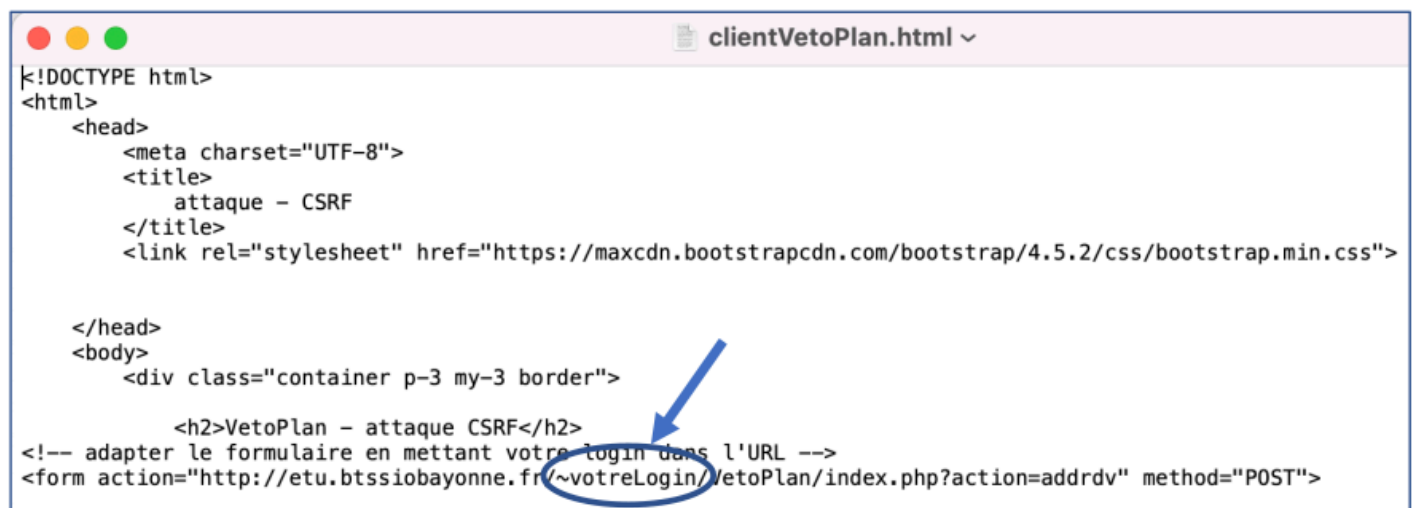
Etape b

Consulter les rendez-vous qui vous sont affectés à l'aide de l'application et confirmer à l'aide de phpmyadmin. Combien de rendez-vous avez-vous ?

En tant qu'Amal Hecker, vous recevez un e-mail malveillant reprenant le design de l'application et contenant un bouton vous permettant d'accéder à vos rendez-vous dans l'application. Cet e-mail est accessible dans le dossier "**client vetoplan**" fourni en ressources. Le contenu du fichier **clientVetoPlan.html** correspond à l'e-mail qu'Amal Hecker a reçu.

Etape c

Adapter le fichier en l'ouvrant à l'aide d'un éditeur de texte (bloc note), et en mettant votre login à l'endroit mentionné dans l'URL.



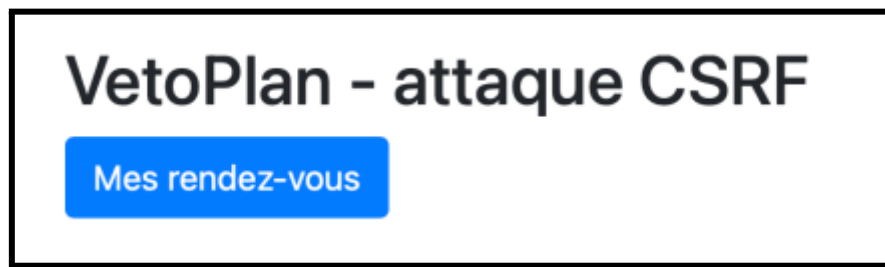
```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>
      attaque - CSRF
    </title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  </head>
  <body>
    <div class="container p-3 my-3 border">
      <h2>VetoPlan - attaque CSRF</h2>
      <!-- adapter le formulaire en mettant votre login dans l'URL -->
      <form action="http://etu.btssiobayonne.fr/~votreLogin/VetoPlan/index.php?action=addrdv" method="POST">

```

Etape d

Ouvrir le fichier clientVetoPlan.html dans le même navigateur que celui utilisé pour se connecter à VetoPlan puis cliquer sur le bouton "Mes rendez-vous". Vous devriez accéder à la liste de vos rendez-vous

*Etape e*

Consultez à nouveau vos rendez-vous dans l'application et dans phpmyadmin. Combien avez-vous de rendez-vous ? Quelle action a produit le bouton du fichier clientVetoPlan.html ?

L'application **VetoPlan** possède un contrôleur permettant d'ajouter un nouveau rendez-vous (**addRdvPost.php**). Ce contrôleur appelle la vue **vueAjoutRdvPost.php**.

Question 1.3

Quels sont les champs du formulaire et la méthode utilisée pour ajouter un nouveau rendez-vous ?

Question 1.4

En observant le code source du fichier client et le code source du contrôleur **addRdvPost.php**, expliquer pourquoi un nouveau rendez-vous a été créé en cliquant sur le bouton "Mes rendez-vous" du fichier client.

Question 1.5

Compléter le tableau "Impacts des événements redoutés" pour l'événement n°6

Question 1.6

En consultant l'annexe 1, indiquer quelle user story est concernée par cet exercice. Les besoins de sécurité sont-ils respectés ? Justifier.

BILAN – FONCTIONNEMENT D'UNE ATTAQUE CSRF

Dans une attaque **CSRF** (Cross Site Request Forgery), l'objectif est de profiter des privilèges d'un véritable utilisateur authentifié dans l'application pour effectuer l'action malveillante.

Le faux lien ou le script transmis dans un mail cache en fait un lien vers une fonctionnalité d'une application :

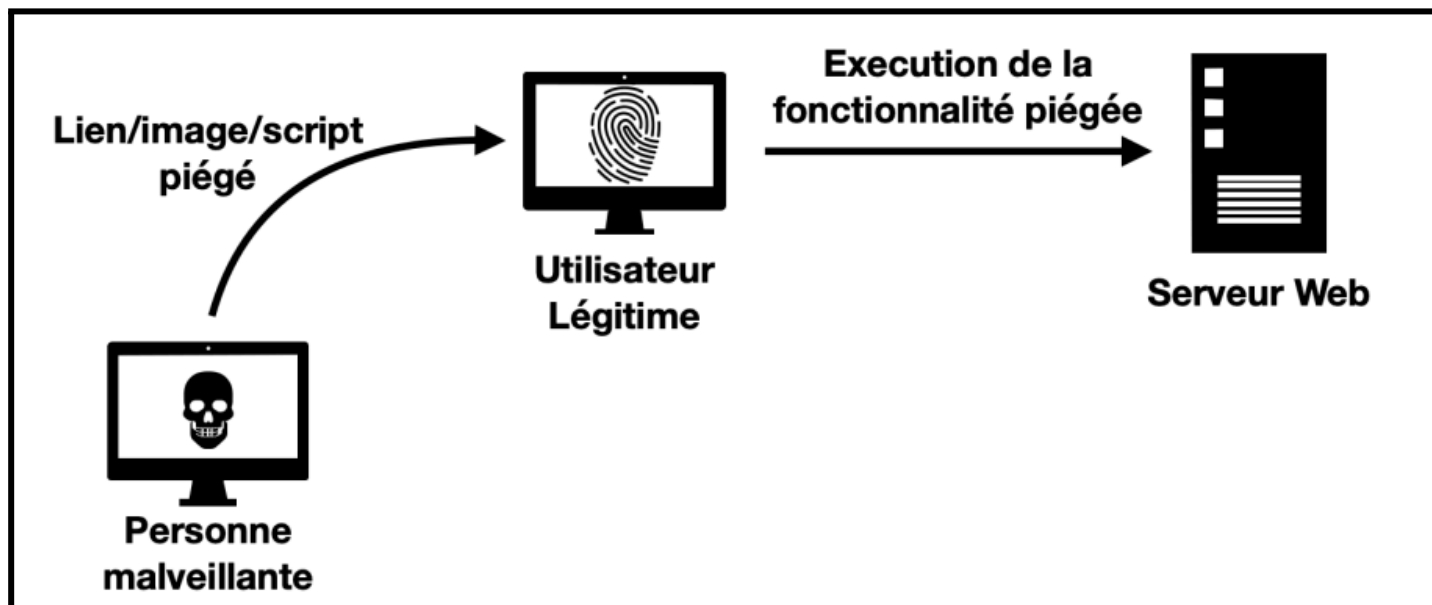


Schéma de fonctionnement d'une attaque CSRF

L'impact sera restreint aux fonctionnalités auxquelles l'utilisateur a accès :

- lorsque c'est un utilisateur avec peu de droits, l'impact est limité mais tout de même grave pour l'utilisateur.
- lorsque c'est un utilisateur avec des droits étendus, un administrateur par exemple, l'impact peut-être très grave.
- Lorsque la gestion des utilisateurs et des autorisations n'est pas bien configurée dans l'application, l'impact peut aussi être très grave.

Une action malveillante peut aller de la publication d'un commentaire diffamatoire, au changement de mot de passe utilisateur, à la perte d'informations.

Exercice 2 : mécanisme de protection contre les failles CSRF

A l'aide du mini projet Symfony (**projet_secu**) installé sur votre serveur virtuel personnel et accessible dans Netbeans. Les frameworks Web intègrent souvent des mécanismes de protection contre les failles CSRF. Le framework Symfony permet d'activer et de désactiver ce mécanisme et donc d'observer la manière dont il protège l'application. Cette configuration est illustrée en [annexe 3](#).

A - Manipulation/préparation :

Etape a

Désactiver la protection csrf dans le projet symfony **projet_secu** : passer le paramètre csrf_protection à false.

Etape b

À l'aide de votre navigateur, consultez le contrôleur **testCSRF** hébergé sur votre serveur.

Question 2.1

Toujours dans le navigateur, afficher le code source de la page affichant le formulaire du contrôleur puis rapporter la structure du formulaire généré

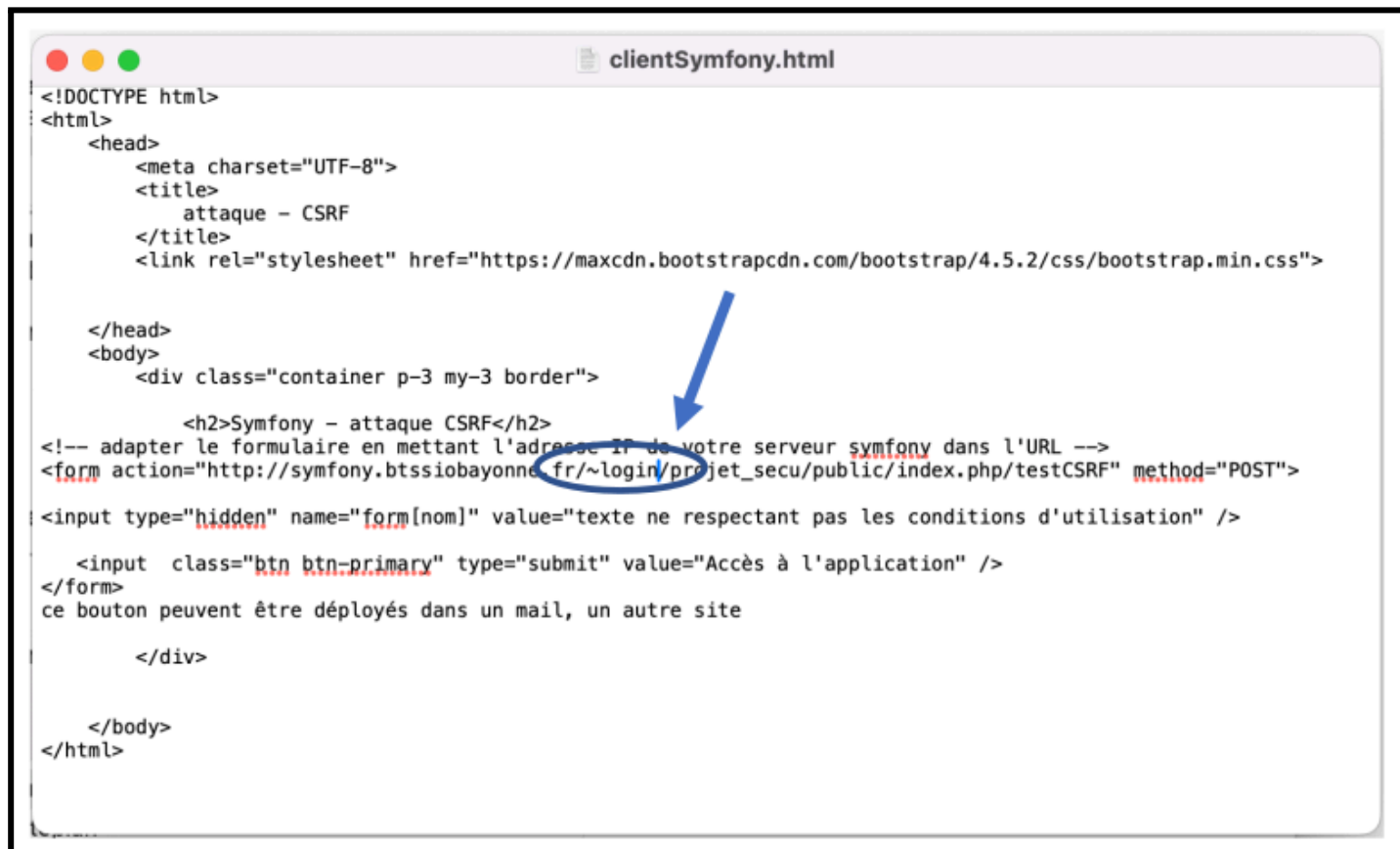
B - Manipulation/préparation :

Etape c

Vérifier le bon fonctionnement du formulaire en ajoutant un "espace" (Espace de vente du TP Upplagg). Vérifier qu'il est bien créé dans la base de données.

Etape d

Éditer le fichier client.html fourni en ressources par la société d'audit dans le dossier "client symfony" et adapter l'URL en remplaçant le mot clé login par le votre.



```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>
      attaque - CSRF
    </title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  </head>
  <body>
    <div class="container p-3 my-3 border">
      <h2>Symfony - attaque CSRF</h2>
      <!-- adapter le formulaire en mettant l'adresse IP de votre serveur symfony dans l'URL -->
      <form action="http://symfony.btssiobayonne.fr/~login/projet_secu/public/index.php/testCSRF" method="POST">
        <input type="hidden" name="form[nom]" value="texte ne respectant pas les conditions d'utilisation" />
        <input class="btn btn-primary" type="submit" value="Accès à l'application" />
      </form>
      ce bouton peuvent être déployés dans un mail, un autre site
    </div>
  </body>
</html>

```

Question 2.2

Relever à l'aide de phpmyadmin les données contenues dans la table espace de la base de données **bdsecu**.

Question 2.3

Ouvrir le fichier **clientSymfony.html** à l'aide du navigateur web et cliquer sur le bouton comme pourrait le faire un utilisateur.

Question 2.4

Observer le contenu de la table espace de la base de données bdsecu. Est-ce qu'une action a été faite ?

C - Manipulation/préparation :

Etape e

Activer la protection csrf dans le projet symfony, puis cliquer à nouveau sur le bouton dans la page clientSymfony.html.

Question 2.5

Observe-t-on un impact sur la base de données ?

Question 2.6

Retourner sur le contrôleur testCSRF à l'aide du navigateur web et actualiser la page. Toujours dans le navigateur, afficher le code source de la page et rapporter la structure du formulaire généré.

Question 2.7

Quel est le nouveau champ du formulaire ? Relever sa valeur (les premiers et derniers caractères).

Question 2.8

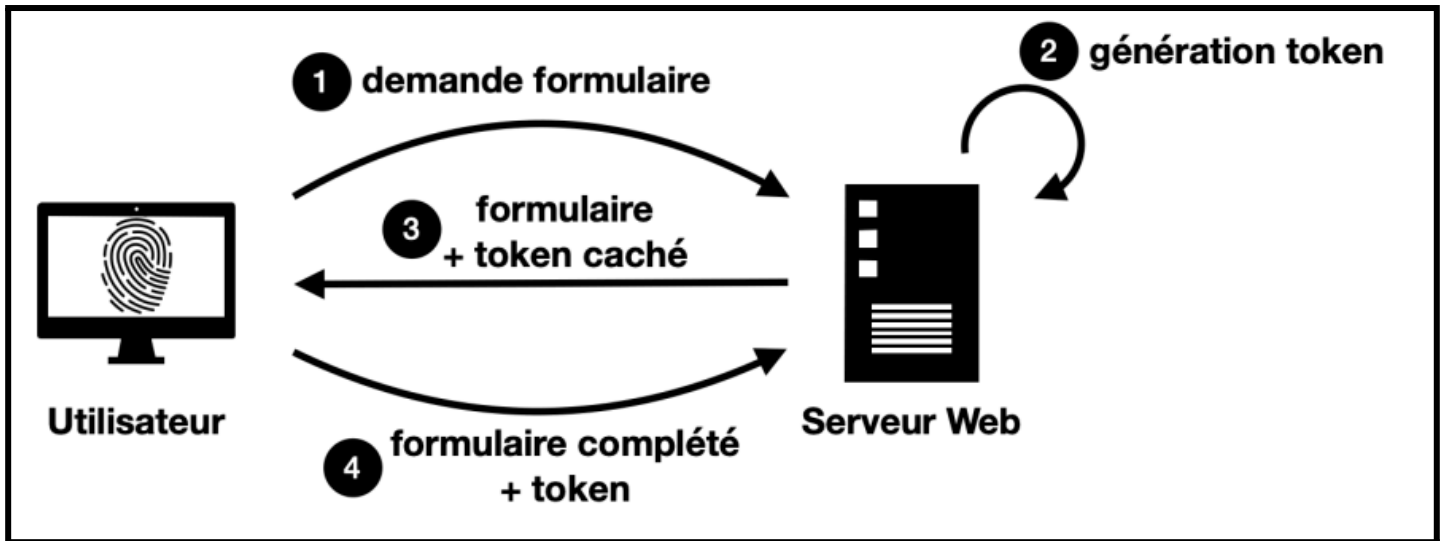
Actualiser le formulaire et relever à nouveau la valeur du nouveau champ du formulaire. Cette valeur est-elle la même ?

Question 2.9

Que manque-t-il dans le formulaire caché du fichier **clientSymfony.html** pour que le nouvel espace soit inséré lorsque l'utilisateur clique sur le bouton ?

BILAN – PROTECTION CONTRE LES ATTAQUES CSRF

Le rôle de cette valeur ajoutée au formulaire (appelée jeton ou token) est de vérifier que le formulaire a bien été complété par la personne qui le valide.



Protection contre les failles CSRF

Lorsque l'utilisateur consulte le formulaire, le serveur génère un token, le stocke et l'ajoute en champ caché dans le formulaire.

Au moment de la validation du formulaire par l'utilisateur, le serveur vérifie que le token généré et caché dans le formulaire est bien celui associé à la session de l'utilisateur.

Dans le cas où le token n'est pas le bon ou est inexistant, les données soumises sont refusées par le serveur.

Conclusion : importance de la revue de code.

La sécurisation d'une application passe par une phase de recherche d'erreurs, d'analyse de qualité de code et de respect de bonnes pratiques.

Le fait de laisser d'autres personnes relire votre code permet d'avoir un œil nouveau et ainsi mieux détecter d'éventuelles erreurs.

En plus d'ouvrir votre code à la validation par vos pairs, il existe des solutions de contrôle de qualité. Ces outils permettent de rechercher les erreurs et de vérifier le respect de bonnes pratiques du code d'une application.

Dans ce support, le fait de protéger les formulaires par des jetons dans une application est une bonne pratique de programmation.

Sonarqube (sonarqube.org) est un exemple d'outil de vérification de qualité.

Recherches :

Répondre aux questions suivantes en utilisant vos connaissances et en recherchant des informations sur internet.

- Un même token peut-il être utilisé par différents utilisateurs à un instant donné ?
- un token peut-il être valable plusieurs fois, pour plusieurs actions effectuées par un même utilisateur par exemple ?
- Comment est stocké le token sur le serveur ?

Exercice 3 : Protection contre les failles CSRF dans l'application VetoPlan

Dans le premier exercice, vous avez constaté que l'application VetoPlan était vulnérable aux attaques CSRF. L'objectif ici est d'implémenter une protection par token dans la fonctionnalité d'ajout de rendez-vous.

A – Outils permettant de générer le token

Les fonctions PHP suivantes vous aideront à générer un jeton :

- **`random_bytes()`**
- **`base64_encode()`**

À l'aide de la documentation PHP et d'exemples trouvés sur internet, rechercher leur rôle, leur fonctionnement, les tester à l'aide d'un simple script PHP puis répondre aux questions suivantes.

Question 3.1

La fonction **`random_bytes()`** produit-elle toujours des chaînes de caractères affichables ? Tester ou rechercher de la documentation pour répondre à la question.

Question 3.2

Quel est le rôle de la fonction **`base64_encode()`** ? Cette fonction retourne-t-elle toujours des chaînes affichables ? Tester ou rechercher de la documentation pour répondre à la question.

Question 3.3

Comment utiliser ces deux fonctions pour générer une chaîne de caractère aléatoire ?

Question 3.4

Écrire la fonction **`generateToken()`** dont le rôle est de générer et de retourner une chaîne de caractère aléatoire affichable d'au moins 20 caractères.

B – Implémentation de la gestion du token dans la fonctionnalité d'ajout de rendez-vous de l'application VetoPlan

Modification du contrôleur et de la vue d'ajout de rendez-vous dans l'application VetoPlan.

Lors de la génération du formulaire, le script doit générer un jeton et l'utiliser pour :

- Coté serveur : le stocker en variable de session,
- Coté client : l'intégrer au formulaire sous forme d'un champ caché

Question 3.5

Adapter le code de validation du formulaire afin de faire en sorte que le formulaire ne soit validé que si le jeton en variable de session et la valeur cachée transmise par le formulaire sont les mêmes. Dans le cas contraire, un message indiquant que le jeton est invalide doit être affiché à l'utilisateur et les données saisies ne sont pas enregistrées en base.

Bonus - pour aller plus loin :

Prendre en compte la durée de vie du jeton : un jeton ne doit pas être valide indéfiniment : au bout de 60 secondes, le jeton utilisé pour générer le formulaire doit être déclaré invalide. Plusieurs approches sont possibles pour répondre à ce besoin. En proposer une et l'implémenter.

Exercices supplémentaires :

Correction d'autres problèmes de sécurité et analyse de risques de l'application vetoPlan.fr

SPRINT 2 – CORRECTION D'UNE FAIBLE DANS LA FONCTIONNALITÉ DE MISE À JOUR D'UN RENDEZ-VOUS

Exercice 4

Mise en situation : Se connecter à l'application avec un compte utilisateur. Aller dans la fiche de détail d'un des rendez-vous à l'aide de l'application. Saisir un nom d'animal, une note, et ajouter en fin de note le texte suivant : ' ;#

Enregistrer la modification puis consulter le contenu de la base de données.

Question 4.1

Quel est l'impact de la valeur saisie sur la base de données ?

Question 4.2

Quel contrôleur et quelle fonction du modèle sont concernés par cette faille de sécurité ?

Question 4.3

Quelle requête SQL devrait être exécutée lors d'une mise à jour normale d'un rendez-vous ?

Question 4.4

Quelle requête SQL a été effectivement exécutée lorsque vous avez saisi la chaîne ' ;# ?

Question 4.5

Cette faille concerne-t-elle les critères suivants ? Justifier.

Disponibilité, Intégrité, Confidentialité, Preuve

Question 4.6

Compléter les tableaux "Impacts des événements redoutés" et "Scénarios de risques et mesures" à prévoir de [l'annexe 2](#).

Question 4.7

Corriger l'erreur présente dans la fonction et vérifier que la faille de sécurité est bien comblée. Quel type de faille vient d'être corrigé ?

SPRINT 3 – CORRECTION D'UNE FAILLE DANS LA FONCTIONNALITÉ DE CONSULTATION D'UN RENDEZ-VOUS.

Exercice 5

Contexte :

Le vétérinaire Lionel Romain envisage de s'installer dans la même région qu'Amal Hecker mais n'a pas encore de clientèle. Après s'être connecté à l'application il a profité d'un problème de sécurité non comblé pour récupérer les coordonnées des clients d'Amal.

Mise en situation :

Se connecter en tant qu'utilisateur Lionel Romain (mot de passe : sio) puis aller dans la fiche de détail d'un des rendez-vous à l'aide de l'application. Modifier l'URL en changeant la valeur de l'idR transmit et tenter d'accéder à un rendez-vous d'Amal.

Question 5.1

Est-il possible d'accéder à un rendez-vous d'un autre vétérinaire ? Peut-on le modifier ? Pourquoi est-il possible d'effectuer ces accès ?

Question 5.2

Cette faille concerne-t-elle les critères suivants ? Justifier.

Disponibilité, Intégrité, Confidentialité, Preuve

Question 5.3

Identifier le scénario de risque (annexe 2) associé au problème étudié et compléter la section "mesures à prévoir".

Question 5.4

Implémenter les modifications nécessaires.

SPRINT 4 – MISE EN CONFORMITÉ AVEC LA RGPD.

Exercice 6 : protection des données personnelles

En consultant la structure de la base de données, expliquer quels champs devraient être chiffrés ou hachés pour respecter la RGPD. Justifier vos choix.

ANNEXES

Annexe 1 : Besoins de sécurité pour les user stories de l'application VetoPlan

| | Intitulé de la <i>user story</i> | Disponibilité | Intégrité | Confidentialité | Preuve |
|---|---|---------------|-----------|-----------------|--------|
| 1 | En tant que vétérinaire je peux consulter mes rendez-vous. | ** | ** | ** | - |
| 2 | En tant que vétérinaire je peux consulter les informations détaillées d'un de mes rendez-vous. | * | ** | ** | - |
| 3 | En tant que vétérinaire je peux rédiger une annotation et changer le type d'animal examiné d'un de mes rendez-vous. | * | ** | * | * |
| 4 | En tant que vétérinaire je peux créer un nouveau rendez-vous. | * | ** | * | - |
| 5 | En tant que vétérinaire je peux changer mon mot de passe. | * | ** | ** | * |

- : pas de besoin

* : besoin important

** : besoin très important

Annexe 2 : Extrait du rapport d'audit de l'analyse des risques et menaces de l'environnement

Acteurs à l'origine de la malveillance

| Acteurs malveillants | Modes opératoires | Probabilité |
|-------------------------------------|---|-------------|
| Attaquant externe (<i>hacker</i>) | L'attaquant externe modifie des données dans la base. | ** |
| | L'attaquant externe ajoute des données corrompues dans la base. | ** |
| | L'attaquant externe surcharge le système. | *** |
| Vétérinaire | L'utilisateur consulte des données qu'il ne détient pas | ** |
| | L'utilisateur modifie des données qu'il ne détient pas | * |
| | L'acheteur surcharge le système. | * |
| Utilisateur non connecté | L'utilisateur consulte des données qu'il ne détient pas | * |
| | L'utilisateur modifie des données qu'il ne détient pas | * |
| | Le visiteur surcharge le système. | * |

* : faible probabilité

** : forte

*** : très forte probabilité

Impacts des évènements redoutés

| Numéro de l'évènement | Évènement | Impact pour l'entreprise | Gravité |
|-----------------------|--|---|---------|
| 1 | Le système ne répond pas. | Perte de clients | * |
| 2 | Un utilisateur non connecté parvient à consulter un rendez-vous | Problème de confiance | * |
| 3 | Un utilisateur non connecté parvient à modifier l'animal et les notes d'un rendez-vous | Perte de clients Problème de confiance | ** |
| 4 | Un vétérinaire parvient à consulter un rendez-vous ne lui appartenant pas | | |
| 5 | Un vétérinaire parvient à modifier l'animal et les notes d'un rendez-vous ne lui appartenant pas | Perte de clients Problème de confiance | ** |
| 6 | Un attaquant externe parvient à faire créer un faux rendez-vous à un vétérinaire | Perte de clients | ** |

| | | | |
|----|--|---|----|
| | | Problème de confiance Désorganisation des rendez-vous | |
| 7 | Un attaquant externe parvient à faire en sorte qu'un vétérinaire modifie son mot de passe sans qu'il ne le sache | Perte de clients Application inaccessible Problème de confiance | ** |
| 8 | Un attaquant externe parvient à consulter un rendez-vous | Problème de confiance | * |
| 9 | Un attaquant externe parvient à faire modifier l'animal et les notes d'un rendez-vous par un vétérinaire | | |
| 10 | Un attaquant externe parvient à modifier l'animal et les notes d'un rendez-vous | Perte de clients Problème de confiance | ** |

* : modérée

** : très élevée

Scénarios de risques et mesures à prévoir

| Numéro de l'évènement | Scénario de risque | Mesures à prévoir |
|-----------------------|--|--|
| 4 | En tant que vétérinaire malveillant, je souhaite pouvoir récupérer les informations des clients d'autres vétérinaires. | |
| 5 | | |
| 6 | En tant qu'attaquant externe, je souhaite faire créer un faux rendez-vous par un vétérinaire. | 6.1 Implémenter un système de protection contre les failles CSRF |

Remarques :

Les numéros d'évènements des tableaux "Impacts des évènements redoutés" et " Scénarios de risques et mesures à prévoir" sont liés.

Annexe 3 : configuration Symfony – protection csrf

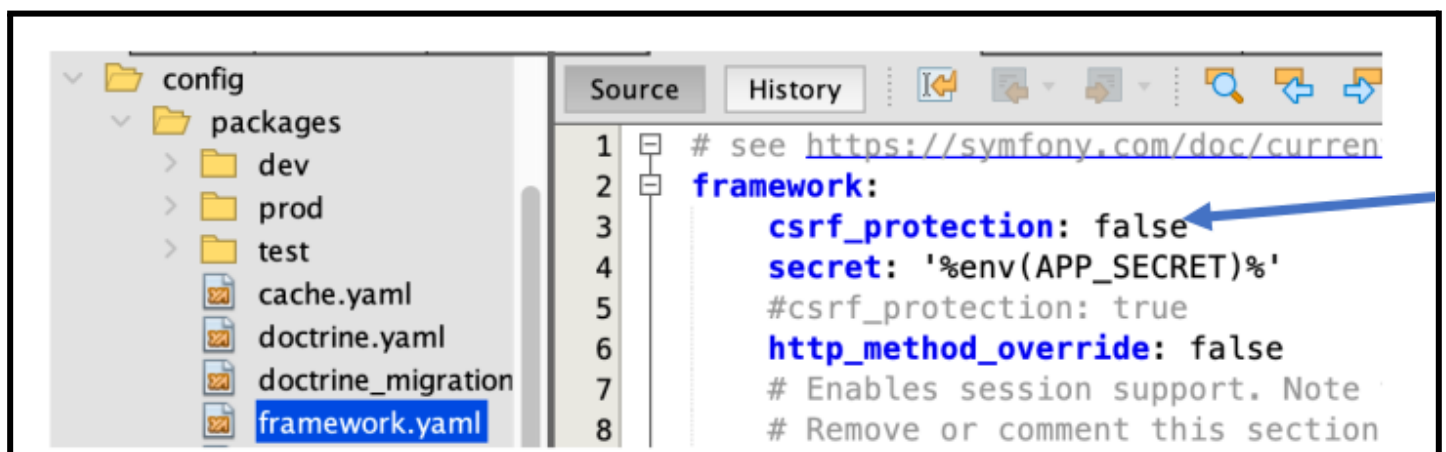
Pour gérer l'activation et la désactivation de CSRF dans symfony :

Si il n'est pas déjà présent, ajouter le paramètre "**csrf_protection**" dans la section framework du fichier **config/packages/framework.yaml**

```
framework:
  csrf_protection: true
```

(attention à bien respecter l'indentation existante et celle de l'option ajoutée. La valeur true permet d'activer la protection contre les failles csrf. La valeur false permet de la désactiver.

Exemple :



Protection CSRF désactivée