Veille Technologique 2024

Comment créer un site web et le protéger des menaces de l'OWASP top 10 ?

12 FEVRIER 2024

Écrit par : Aron Noa

Table des matières

Introduction

Si vous voulez améliorer votre présence en ligne, stocker joliment vos photos de famille ou bien encore partager ce qui vous fait envie, vous aurez probablement besoin de créer un site web.
Mais seulement, par où commencer ?

Vous avez sûrement entendu parler d'HTML, mais est-ce vraiment tout ? Si vous êtes curieux, vous devez même savoir que c'est lié au CSS. Ces deux langages sont de bonnes bases afin de faire un site d'ores et déjà solide. Cependant, cela ne peut pas se résumer qu'à ces principes "néophytes". Évidemment, rien qu'avec ces deux langages vous aurez déjà bien assez à explorer. Malgré ces efforts, il reste un aspect à ne surtout pas négliger et c'est la sécurité !

Quelle chance pour nous, une organisation regroupe les principales erreurs à éviter avant de commettre l'irréparable : OWASP TOP 10.

Dans ce rapport, nous verrons toutes les choses à savoir dans le but de créer un site web finalisé et sécurisé. J'espère que l'introduction vous a mis en jambe car il nous reste un tas de termes à développer et un tas de chose à coder.
On en vient à notre question :
Comment créer un site et le protéger des menaces de l'OWASP TOP 10 ?

Site web

L'histoire de l'HTML

Au début des années 1990, un groupe de passionnées s'est rassemblé pour résoudre un défi : partager des informations à travers un réseau mondial appelé Internet. Parmi eux se trouvait une personne du nom de Tim Berners-Lee.

Travaillant à L'Organisation européenne pour la recherche nucléaire (CERN), il cherchait un moyen de faciliter la communication entre les chercheurs dispersés dans le monde entier. Un langage qui puisse décrire et lier des documents d'une manière compréhensible par les ordinateurs et les humains. Le nom qui lui fut donné ; l'HyperText Markup Language, l'HTML pour les intimes.

C'était comme une langue universelle pour le World Wide Web naissant. Tim Berners-Lee publia sa première spécification de l'HTML en 1991, permettant aux créateurs de sites web de structurer leurs documents de manière qu'ils puissent être partagés et consultés par d'autres à travers le monde.
Ses versions naissantes étaient simples, décrivant principalement la structure de base des documents, mais il évolua avec le temps pour inclure de nouvelles fonctionnalités et mieux représenter la complexité croissante des pages web.

Les concepteurs de sites web ont rapidement compris le pouvoir de l'HTML pour créer des expériences interactives en combinant le langage avec d'autres technologies émergentes telles que les feuilles de style en cascade (CSS) pour la présentation et JavaScript pour l'interactivité.

À travers les décennies, il est devenu un élément fondamental de la création web. Les versions successives, comme l'HTML4, XHTML, HTML5 ont apporté des améliorations significatives offrant de nouvelles balises, attributs et fonctionnalités.

Sachez que sa lancée ne s'arrête pas ! L'HTML continue d'évoluer pour prendre en charge des technologies émergentes telles que l'Internet des objets (IoT) et l'accessibilité améliorée pour tous les utilisateurs.

Maintenant que vous avez pris conscience de l'histoire, nous sommes fin prêt pour débuter. La première chose à laquelle penser est : quel genre de site produire ? En effet, les besoins ne sont pas les mêmes selon la réalisation voulue. Les nécessités seront différentes si vous comptez héberger des vidéos, ouvrir un blog ou bien d'autres possibilités. Nous avons fait notion plus haut de l'internet des objets, cela se réfère à la connectivité des objets physiques à Internet, permettant la collecte, la transmission et le partage de données entre ces objets. L'HTML peut être utilisé pour concevoir des interfaces utilisateurs (UI) interactives pour les applications IoT. Les technologies web complémentaires, telles que JavaScript et les appels API (Application programming Interface) pour récupérer des données en temps réel peuvent être intégrées dans des pages HTML pour créer des applications IoT interactives et réactives.

Il est donc important de prendre en compte ces paramètres !

Revenons à nos moutons, l'HTML.

IDE

Ce chapitre est très court, c'est tout simplement votre environnement de développement. Pour coder en HTML, c'est simple. Il suffit d'un bloc-notes. Méthode rudimentaire pour débuter. Nonobstant, nous atteignons vite les limites de cette technique. Il y a plein d'outils open-source et plus ergonomiques pour coder. Lors d'un usage plus avancé comme la manipulation de Framework (nous reviendrons sur ce sujet plus tard), le procédé du bloc-notes devient obsolète (bien avant cela dans les faits).

Pour ma part, j'ai utilisé un des plus connus si ce n'est le plus connu : Visual Studio Code. Il possède pléthore de plugin créer par la communauté et rend son utilisation sérieusement agréable. Je recommande l'utilisation du plugin "Live server" qui concède une actualisation en direct du site web. Localement, à chaque modification du code, il actualisera la page.

HTML5

Balises

De nos jours, c'est de cette version que nous nous servirons. HTML de son acronyme HyperText Markup Language est un langage de balisage (en effet markup se traduit par balisage). Concrètement, cela signifie que des balises vont entourer des éléments spécifiques d'une page afin de marquer plusieurs parties de son contenu. Pour citer les plus célèbres :

1. Structure de base :
- <html> : Définit le début et la fin du document
- <head> : Contient des informations sur le document, telles que le titre, les liens vers les feuilles de style, etc
- <title> : Définit le titre du document qui apparaît dans l'onglet du navigateur
- <body> : Contient le contenu principal de la page web.
2. Titres et paragraphes :
- <h1>, <h2>..., <h6> : Définissent les titres de différents niveaux
- <p> : Définit un paragraphe
3. Listes :
- : Définit une liste non ordonnée
- : Définit une liste ordonnée.
- : Définit un élément de la liste
4. Liens et ancres :
- <a> : Définit un lien hypertexte
- <img> : Insère une image
5. Tableaux :
- <table> : Définit un tableau
- <tr> : Définit une ligne dans un tableau
- <td> : Définit une cellule de données
6. Formulaires :
- <Form> : formulaire
- <Input> : champ de saisie
- <Textarea> : zone de texte multi-lignes
- <Button> : bouton
7. Multimédia :
- <Audio> : Intègre des fichiers audios
- <Video> : fichiers vidéo
8. Style et mise en forme :
- <Div> : Divise le document en sections (La balise la plus importante !)
- <Span> : applique du style à une partie du texte
9. Autres :
-
 : Insère un saut de ligne
- <Hr> : insère une ligne horizontale

Sémantique

10. Attention :
- <I> : Définit une partie du texte dans un autre ton ou une autre “humeur”
- <U> : Représente un texte qui doit être “stylistiquement” différent du texte normal, tel que des mots mal orthographiés.
- <Strong> : texte en gras
- <Em> : texte en italique
- <Mark> : permet de mettre en évidence des parties d’un texte.

Pour cette dernière catégorie, il faut savoir en faire cas. Dans les faits, un texte qui sera compris entre deux balises s’affichera bien en gras, mais c’est une mauvaise habitude à prendre en cas de mise en forme graphique. En réalité, elles ne sont pas vouées à cette utilisation. Lors de la consultation des définitions et usages de chacune de ces balises sur la documentation associée, on constate bien que ces balises ont plutôt une fonction sémantique.
En d’autres termes, la sémantique se concentre sur le sens des éléments plutôt que sur leur apparence visuelle.

Pour reprendre l’exemple de la balise , il indique un contenu de haute importance, y compris lorsqu’il s’agit de choses graves ou urgentes comme des messages d’avertissement par exemple. Il se traduit donc par l’affichage du texte en gras.
Il serait donc plus ingénieux d’utiliser une balise afin d’ajouter à votre texte le style que vous souhaitez avec votre CSS associé (voir ci-dessous).
Toutes ces balises permettent d’écrire et d’insérer à peu près ce que vous voulez sur une page web. Laissez-moi vous avertir qu’il ressemblera néanmoins à un site vieux de quelques années (en étant généreux). C’est ici qu’intervient l'épatant langage permettant de moderniser notre HTMTL brut : le CSS.

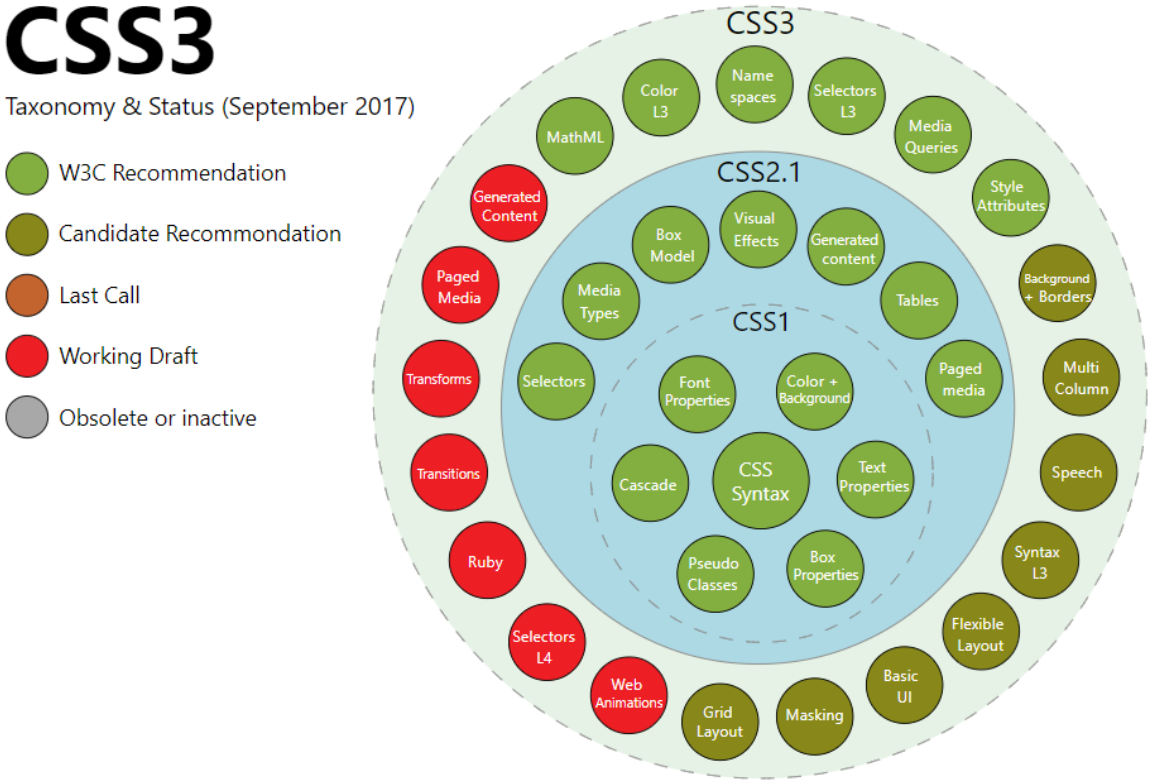
CSS

Les feuilles de style en cascade (Cascading Style Sheets), CSS est un langage utilisé pour décrire la présentation visuelle d’un document écrit en HTML (ou autre langage de balisage, le Markdown par exemple) comme dit plus haut.
Ses principales fonctionnalités sont les suivantes :

- Séparation des préoccupations : il sépare la structure d’une page HTML de sa présentation visuelle. Cela signifie que vous pouvez définir la structure de votre page, puis appliquer le style avec CSS.
- Réutilisation : En utilisant des règles CSS, vous pouvez appliquer un style cohérent à plusieurs pages web. Cela facilite la maintenance et la mise à jour du style global d’un site.
- En cascade : Les styles peuvent être appliqués de manière hiérarchique et être hérités. Si plusieurs règles CSS s’appliquent à un même élément, le navigateur détermine la priorité en fonction de la spécificité des sélecteurs, de l’importance des styles et de l’ordre dans lequel les styles sont déclarés.
-

Au début des années 1990, la mise en forme des pages web était effectuée directement dans le code HTML. Cependant, cela conduisait à un code complexe et difficile à maintenir. En 194, Håkon Wium Lie et Bert Bos proposent une première version du CSS lorsqu’ils travaillent au CERN (encore).

Comme l'HTML, il n'a cessé d'évoluer.



Il y a bien trop de commandes pour que je puisse toutes les expliciter comme je l'ai fait pour l'HTML. Il faut cependant connaitre certains aspects importants.
La syntaxe de base : composées de sélecteurs et de déclarations. Il cible un élément HTML, et les déclarations spécifient le style à appliquer à cet élément.
Exemple :

```
ul {
    list-style: none;
}
```

Classe

Vous pouvez modifier l'apparence d'énormément de choses. Tel que la couleur du texte, la couleur de l'arrière-plan, où sont placés vos paragraphes, des ombres portés, la police d'écriture et tant d'autres.
A noter que l'ajout de classe dans l'HTML permet d'affiner la précision d'un élément. Dans des sites professionnels ou sobrement complexe, il est essentiel d'utiliser ce genre d'astuce. Si vous possédez deux titres <h1> mais que vous négligez la différenciation, ne soyez pas surpris de voir un style similaire sur vos deux titres.

Exemple :

```html
<footer class="footer">
    <div class="container">
        <div class="row">
            <div class="footer-col">
                <h4>à propos</h4>
                <ul>
                    <li><a href="#">sujet</a></li>
                    <li><a href="#">difficultés</a></li>
                    <li><a href="#">informations supplémentaires</a></li>
                </ul>
```

Vous pouvez également ajouter des valeurs à vos paramètres. Spécifier la hauteur d'un texte. A quelle distance voulez-vous qu'il soit du bord droit de l'écran etc.

```css
.footer-col {
    width: calc(33.33% - 30px);
    padding: 0 15px;
    box-sizing: border-box; /* Inclusion du padding dans la largeur */
}

.footer-col h4 {
    font-size: 18px;
    color: #fff2e4;
    text-transform: capitalize;
    margin-bottom: 35px;
    font-weight: 500;
    position: relative;
}
```

Disposition

Pour éviter de rentrer des valeurs relativement aléatoires quant à vos placements d'éléments, la notion de modèle de boite est très importante. Elle définit la manière dont les éléments HTML sont rendus dans le navigateur, en les considérant comme des boîtes rectangulaires avec du contenu, des marges et des bordures.
Flexbox et Grid facilitent la création de mise en pages complexe et réactives.

Exemples (« display : grid » dans le cas inverse) :

```css
.row {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between; /* Espacement équitable */
}
```

Il permet également de créer des animations et des transitions pour ajouter des effets dynamiques aux éléments.
Exemple :

```css
.footer.show {
    transform: translateY(0);
}
```

Ceci représente les principaux concepts pour réaliser un site web fonctionnel en omettant l'hébergement (nous reviendrons sur l'hébergement à la fin de ce rapport).
Mais pourquoi s'arrêter ici ? Après tout on peut aller bien plus loin !

JavaScript

Langage de programmation interprété, principalement utilisé du côté client pour rendre les pages web interactives et dynamiques (je précise bien principalement mais son utilisation est toute aussi effective côté serveur). Le code source est exécuté directement par le navigateur web sans nécessiter de compilation préalable. Il est principalement utilisé du côté client pour interagir avec le Document Object Model (DOM) et manipuler la structure et le style des pages web.
Il permet d'ajouter, de supprimer et de modifier des éléments HTML, ainsi que de réagir aux événements utilisateur.

Il admet la réaction aux événements tels que les clics de souris, les pressions de touches, etc., et d'appliquer des actions en conséquence.
Il prend en charge des opérations asynchrones, comme les requêtes AJAX, permettant de mettre à jour des parties spécifiques d'une page sans recharger toute la page.
D'une simple balise <script>, il peut être inclus dans un code HTML.
Exemple :

```html
<script src="./script.js"></script>
```

Il peut être utilisé pour valider les données de formulaires côté client avant de les envoyer au serveur. Il peut modifier dynamiquement le contenu d'une page en fonction des entrées de l'utilisateur.
Réaction aux événements et manipulation du DOM :

```
darkmodeToggle.addEventListener('click', () => {
    const body = document.body;
    if(body.classList.contains('dark')){
        body.classList.add('light')
        body.classList.remove('dark')
        toggleImage.src = "./img/MoonDarkmode.png";
    }

    else if(body.classList.contains('light')){
        body.classList.add('dark')
        body.classList.remove('light')
        toggleImage.src = "./img/SunDarkmode.png";
    }
})
```

C'est un bout de code que j'ai réalisé afin d'instaurer un bouton changeant le thème de la page. Il est disponible sur le site final résumant ce rapport, incluant des tests et autre. L'url de celui-ci est disponible en bas de page.
Ce n'est qu'une infime partie de ce qu'il est possible de faire avec JavaScript. Si l'envie vous en dit, il est même possible de créer des jeux vidéo directement dans le navigateur, simuler de la 3D et tant d'autres. Et je n'ai même pas parlé des frameworks... Mais si vous y tenez.

Framework

Utilité

C'est une structure logicielle, un ensemble d'outils, de bibliothèques, de conventions et de composants préconçus qui fournit une architecture de base pour le développement d'applications. Les frameworks sont conçus pour simplifier et accélérer le processus de développement en offrant une structure organisée et des fonctionnalités réutilisables.

Introduisons le principe de "convention sur configuration", ce qui signifie que les développeurs peuvent utiliser des conventions prédéfinies pour organiser le code, mais ont la flexibilité de personnaliser ces conventions si nécessaire.

Frameworks les plus connus pour JavaScript

Angular :
Développé par : Google.
Type : Framework MVVM (Modèle-Vue-VueModèle).
Caractéristiques : Utilisé pour la construction d'applications web complexes et dynamiques. Fournit un système de liaison de données bidirectionnelle, une gestion des dépendances, et un ensemble d'outils pour la création d'interfaces utilisateur riches.

React :
Développé par : Facebook.
Type : Bibliothèque de composants.
Caractéristiques : Se concentre sur la construction d'interfaces utilisateur réactives et modulaires. Introduit le concept de composants réutilisables. Souvent utilisé avec d'autres bibliothèques ou frameworks pour créer des applications à page unique (SPA).

Vue.js :
Développé par : Evan You.
Type : Framework MVVM.
Caractéristiques : Léger et flexible, adapté à des applications de toutes tailles. Facile à intégrer dans des projets existants. Fournit un système de composants, une gestion de l'état réactif, et une approche progressive pour l'adoption.

Node.js :
Développé par : Joyent (et la communauté open source).
Type : Environnement d'exécution pour JavaScript côté serveur.
Caractéristiques : Permet l'exécution de JavaScript côté serveur, utilisé pour construire des applications web en temps réel, des API, des serveurs, etc. Extensible avec des modules npm.

Il y en a de toute sorte. Il est important de rappeler que de nouveaux framworks sortent tous les mois et les tendances tendent à varier selon les années. Certains sont plus spécifiques au front-end tandis que d'autres se rapprocheront plus du back-end.

Front-end / Back-end

Petite précision sur ces deux termes ;

- Front-end : on peut également l'appelé le "côté client", ça va être la partie visible d'une application web avec laquelle les utilisateurs interagissent directement. Il est responsable de la présentation et de l'interaction avec l'utilisateur, gérant l'interface utilisateur, le design, la mise en page, les animations et la gestion des événements.
- Back-end : Cette fois-ci, ce sera le "côté serveur" pour lui, ce sera la partie invisible de l'application web qui gère les opérations en coulisses et communique avec la base de données. Il peut utiliser des langages de programmation tels que Python, Ruby, PHP, Java etc. Il exécute sur un serveur et gère la logique de l'application. La gestion de données, l'authentification des utilisateurs, la sécurité et les opérations côté serveurs. Il communique souvent avec une base de données pour stocker et récupérer des informations.

Pour revenir à nos frameworks, concernant le front-end, je citerai seulement Three.js et Bootstrap. L'un utilisant WebGL pour le rendu 3D dans les navigateurs et l'autre simplifiant la création d'interface réactives et esthétiquement agréables.

Quant au back-end, citons Django, un Framework web open source basé sur Python. Suivant le principe du "batteries included", offrant une grande variété de fonctionnalités prêtes à l'emploi, y compris un ORM (Object-Relational Mapping) intégré, un système d'administration, un système de gestion des URL, et des outils de sécurité avancés. Django est souvent utilisé pour le développement rapide d'application web.
Tant d'outils pour tant d'utilisations différentes, c'est à en avoir la tête qui tourne !

Modèle Vue-Contrôleur

C'est un motif d'architecture logicielle qui divise une application en trois composants interconnectés pour organiser le code de manière modulaire et faciliter la maintenance et le développement. Chaque composant a une responsabilité spécifique dans le fonctionnement de l'application.

Brève explication des trois composants du modèle :

- **Model** : représente la couche de données et la logique métier de l'application. Il gère l'accès aux données, effectue les opérations de manipulation des données, et notifie la vue des changements éventuels. Il est indépendant de l'interface utilisateur. Il peut interagir avec la base de données, effectuer des calculs, et gérer les règles métier de l'application.
- **View** : responsable de l'affichage de l'interface utilisateur et de la présentation des données. Elle reçoit des informations du modèle et affiche ces données à l'utilisateur de manière appropriée. Elle est généralement passive et réactive aux changements du modèle. Elle ne contient pas de logique métier significative mais se contente d'afficher les informations fournies par le modèle.
- Controller : agit en tant qu'intermédiaire entre le modèle et la vue. Il reçoit les entrées de l'utilisateur, traite ces entrées (généralement en invoquant des actions sur le modèle), puis met à jour la vue en conséquence. Il contient la logique d'application et interprète les actions de l'utilisateur. Il peut également mettre à jour le modèle en fonction des interactions.

L'architecture MVC favorise la séparation des préoccupations, ce qui signifie que chaque composant est responsable d'une tâche spécifique et ne se préoccupe pas des autres. Cette séparation permet une maintenance plus facile, des tests unitaires plus efficaces, et la réutilisation des composants dans d'autres parties de l'application.

Responsive

L'utilisateur qui se connectera sur votre page web peut se trouver sur un ordinateur, une tablette ou une télévision. Le responsive est le fait que votre site web soit adapté à n'importe quel type d'écran. Lorsque les effets de style s'accumulent, que la page devient généralement complexe, le responsive devient délicat à mettre en place efficacement.
Il s'effectue par de simples requêtes media queries :

```
/*responsive*/
@media (max-width: 767px) {
    .footer-col {
      width: 50%;
      margin-bottom: 30px;
    }
}
```

Quand l'écran devient plus petit que 767 pixels en longueur, tu modifies les paramètres du footer.

On a parcouru l'ensemble des sujets à maitriser afin de faire un site avancé. Tout compte fait, il nous manque quelques notions comme les bases de données, mais nous en parlerons un peu plus loin dans le rapport.
Car maintenant, c'est le moment d'aborder la **sécurité**. Point extrêmement important lorsqu'il y a une interaction avec les données sensibles d'un utilisateur, les clients s'attendent à ce que leurs informations soient traitées de manière confidentielle ou encore que votre site soit dans la capacité d'être toujours disponible en cas d'attaque malveillante.
Nous observerons cela à travers les 10 menaces les plus importantes recensées par expertise il y a 3 ans : OWASP TOP 10.

OWASP TOP 10

Introduction

J'ai précisé qu'une évaluation a élu les places il y a 3 ans, en 2021, mais cette organisation à but non lucratif existe depuis 2003 et se consacre à l'amélioration de la sécurité des logiciels.

La liste est élaborée par des experts de la sécurité de l'information et des professionnels de l'industrie, membres de la communauté OWASP. Ces chevronnés contribuent à l'identification, à l'analyse et à la classification des vulnérabilités les plus critiques et répandues observées dans les applications web.

L'objectif est de sensibiliser les développeurs, les responsables de la sécurité et d'autres parties prenantes aux risques les plus courants afin qu'ils puissent prendre des mesures pour les atténuer.
La liste est ensuite publiée après un processus de révision et de validation par la communauté.

Il est important de noter que ce n'est pas une liste statique, elle évolue avec le temps pour refléter les changements dans le paysage de la sécurité des applications web. Les organisations sont encouragées à utiliser cette liste comme guide pour améliorer la sécurité de leurs applications et à rester informées sur les mises à jour régulières de l'OWASP Top 10.

La liste de 2021 inclut les menaces suivantes :

1. Contrôles d'accès défaillants
2. Défaillances cryptographiques
3. Injection
4. Conception non sécurisée
5. Mauvaise configuration de sécurité
6. Composants vulnérables et obsolètes
7. Identification et authentification de mauvaise qualité
8. Manque d'intégrité des données et du logiciel
9. Carence des systèmes de contrôle et de journalisation
10. Falsification de requête côté serveur

Nous allons voir les spécificités de chaque catégorie et comprendre comment s'en prémunir.

Contrôles d'accès défaillants

Représentant actuellement le défi le plus important pour la sécurité des applications. Le recours excessif à des solutions automatisées pour relever ce défi crée un faux sentiment de sécurité. Les failles dans le contrôle d'accès permettent aux utilisateurs d'accéder à des données sensibles et leur confèrent la possibilité d'effectuer des actions au-delà des autorisations prévues. Les conséquences de ces vulnérabilités peuvent conduire à la divulgation, à la modification, voire à la destruction des données.

Le contrôle d'accès peut être considéré comme une vérification d'autorisation, garantissant que l'accès aux ressources et la capacité d'effectuer des actions sont accordés à certains utilisateurs (les administrateurs par exemple) et pas à d'autres. Ces contrôles sont généralement effectués après le processus d'authentification, au cours duquel l'identité déclarée d'un utilisateur est vérifiée.

En matière de sécurité des applications web, le contrôle d'accès est étroitement lié au contenu, à l'utilisation prévue des fonctionnalités et aux différents rôles des utilisateurs. Il peut s'agir, par exemple, d'empêcher un utilisateur peu privilégié d'exécuter des fonctions administratives, d'empêcher un utilisateur d'accéder aux ressources d'un autre utilisateur, ou de tout contrôle qui accorde ou refuse l'accès à des ressources ou à des fonctions selon des facteurs contextuels.
Lorsqu'il s'agit d'applications vastes et complexes contenant de nombreux rôles et fonctionnalités d'utilisateurs, les contrôles d'accès peuvent rapidement devenir difficiles à mettre en œuvre correctement.

IDOR

Référence directe à l'objet non sécurisée : considérons une application qui permet aux utilisateurs réguliers de consulter et de modifier les informations relatives à leur compte. Un identifiant est attribué à chaque compte. Lorsqu'une demande de modification est envoyée, l'application détermine quel compte doit être mis à jour en fonction de l'identifiant inclus dans la demande. Dans ce scénario, un attaquant pourrait manipuler une requête sortante destinée à mettre à jour son compte, en modifiant l'identifiant de l'utilisateur pour en faire celui d'une victime.

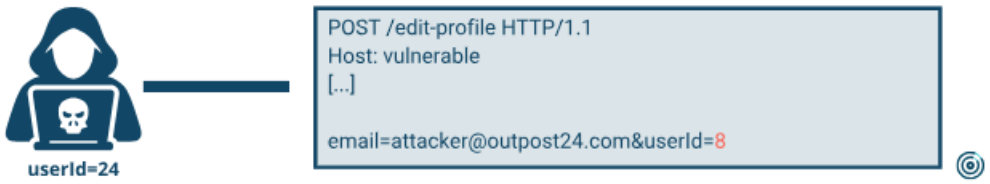


Figure 1 – Attaquant envoyant une requête mettant à jour le compte d'une victime (ID utilisateur 8) au lieu du sien (ID utilisateur 24)

Sans contrôle d'accès en place, le compte de la victime se verrait ainsi modifié, donc une vulnérabilité IDOR avec un impact direct sur l'intégrité. À partir de là, la situation peut rapidement s'aggraver. Si l'attaquant modifie l'adresse électronique de la victime et qu'il demande ensuite une réinitialisation du mot de passe. Cela lui permettrait de définir un nouveau mot de passe et, finalement, de prendre le contrôle du compte de la victime.
Maintenant que nous avons obtenu l'accès au compte d'un autre utilisateur, il convient de clarifier deux termes supplémentaires qui reviennent souvent au sujet du contrôle d'accès défaillant : l'escalade horizontale et l'escalade verticale des privilèges.

L'escalade horizontale des privilèges

Elle consiste à obtenir l'accès aux ressources d'un autre compte avec des autorisations similaires. Dans l'exemple de la section précédente, si le compte de la victime disposait d'autorisations similaires à celles de l'utilisateur attaquant (utilisateur normal), il s'agirait également d'une escalade horizontale des privilèges. En bref, à part l'accès aux ressources d'un autre compte, l'attaquant n'obtient aucune autorisation supplémentaire.

L'escalade verticale des privilèges

Quant à elle, ça désigne l'accès aux ressources d'un autre compte disposant de plus d'autorisations. Dans l'exemple de la section précédente, si le compte de la victime disposait d'autorisations plus élevées (modérateur ou administrateur), il s'agirait d'une escalade verticale des privilèges. L'attaquant peut abuser des autorisations supplémentaires pour lancer d'autres attaques contre l'application.

Traversée de chemin répertoire

Examinons une application qui permettrait aux utilisateurs de visualiser leurs factures à l'aide d'une fonctionnalité de prévisualisation intégrée. Les factures sont stockées sur le même serveur que l'application, au chemin absolu suivant :

La prévisualisation fonctionne en ajoutant la valeur du paramètre « file » au chemin d'accès absolu donné. Il récupère ensuite le contenu de ce fichier et le renvoie à l'utilisateur demandeur.
Cependant, comme dans le cas du scénario IDOR, un attaquant pourrait également capturer la requête sortante et modifier son paramètre « file » en un fichier qui n'est pas destiné à être récupéré.

En l'absence de contrôle d'accès, l'attaquant essaierait alors de récupérer le document :
/var/www/html/vulnerable-application/invoices/../../../../../etc/hosts

La séquence point-point-slash (../) recule d'un répertoire dans le chemin d'accès. Actuellement dans la racine du système de fichiers, une entrée dans le répertoire demande le fichier hosts (fichier par défaut qui établit une correspondance entre les noms d'hôtes et les adresses IP) ne serait probablement pas le premier choix d'un attaquant.
Je le mentionne seulement ici d'inclusion de fichiers locaux. Un pirate essaierait plutôt de récupérer des fichiers sensibles, à la fois à l'intérieur et à l'extérieur du répertoire de l'application.

Octet nul

Petite aparté sur cette attaque permettant de contourner certains filtres au niveau du serveur http, en créant une confusion entre l'adresse demandée et celle compromise par l'application.
En insérant la chaîne de caractères "%00"dans une url, il y a de fortes chances pour que l'application web remplace cette chaine par un caractère de code ASCII 0, qui sert de marqueur de fin déchaîne de caractères dans de nombreux langages de programmation. Un quelconque attaquant pourrait donc extraire une partie du code source de la page.

Solutions

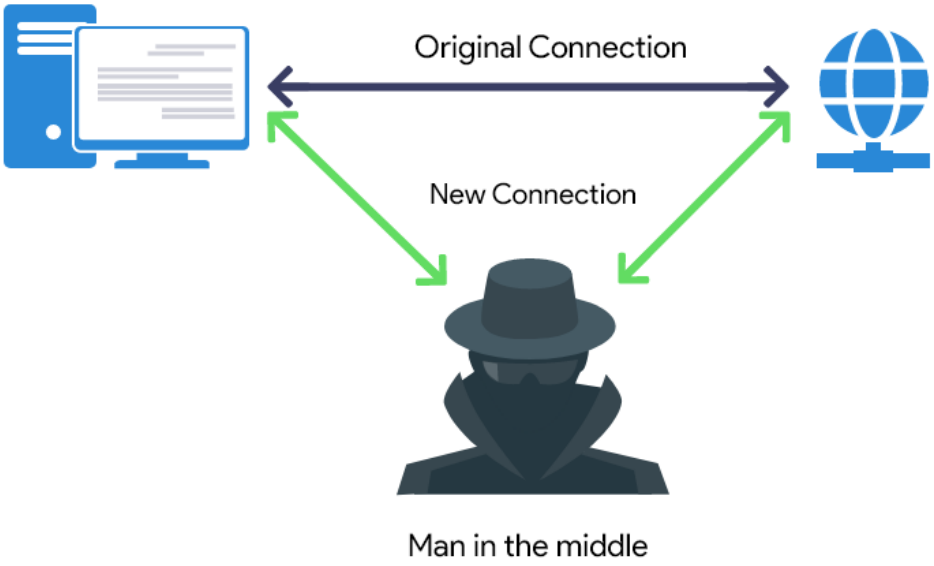
Les scanners de vulnérabilité sont une solution populaire afin d'identifier les failles dans les applications. Cependant, il faut faire attention, bien qu'ils puissent fournir des résultats de manière continue et rapide, ils sont limités dans leur capacité à détecter de nouveaux vecteurs d'attaque ou d'autres vulnérabilités nécessitant de l'intuition et du raisonnement.
Éviter l'utilisation de noms prévisibles ou de références directes à la base de données dans l'URL.
Et pour finir, utiliser des références d'objet indirectes avec des paramètres et des combinaisons clé-valeur.

Défaillances cryptographiques

La cryptographie est aujourd'hui un des moyens les plus sûrs pour protéger des données privées. Que ce soient des données stockées, dans des bases ou sur une partition, ou des données en transit.
Avant ça, il est important de comprendre comment les données transitent la couche de transport.
Pour afficher les pages web, le navigateur utilise le protocole HTTP et TCP/IP pour les transmettre.

Une fois l'URL rentré, le navigateur va lancer une connexion TCP, qui va envoyer des requêtes GET et POST pour vous connecter au serveur web associé au nom de domaine ou à l'adresse IP.
Si le serveur web établit la connexion TCP avec le navigateur, une réponse avec le code statut et le fichier demandé (généralement le fichier index.html pour la page web) sera transmise. Mais dans le cas d'une navigation non sécurisée, les données transitent en HTTP et pas en HTTPS…
Ceci nous amène à la première vulnérabilité. Les données transitant en HTTP peuvent être interceptées, car elles transitent en clair.

Man in the middle



C'est l'une des principales causes de détournement de session. L'attaquant se place au milieu d'une communication et intercepte les connexions réseau et capte ainsi tout ce qu'un client voit.

HTTPS

C'est grâce à ce protocole que nous allons éviter ce problème. C'est la version sécurisée du protocole bien connu http servant à chiffrer les transmissions. Un certificat SSL est nécessaire pour obtenir cette sécurité. Des autorités proposent ces certificats gratuitement.

L'autre manière d'éviter ce genre de problème est le hachage des données mais nous verrons cette spécificité dans un chapitre plus bas dans ce rapport (voir. Identification et authentification de mauvaise qualité).

Injection SQL

Il est enfin temps d'aborder les bases de données...

C'est un outil qui vous permettra de collecter et d'organiser toutes vos informations. Et l'injection SQL est une attaque qui peut aller jusqu'à copier et supprimer intégralement les données de cette base. Cette courte explication sert à comprendre à quel point cette menace est dangereuse.
Pour rechercher dans ces données, nous optons pour l'utilisation de requêtes. Une injection va être le fait d'introduire dans une requête un code malveillant.

Par exemple:
*SELECT `id`, `title` FROM `article` WHERE `title` LIKE '%a'; DELETE * FROM article;%'*

La totalité des articles a été supprimé.

Si l'attaquant ne connait pas le nom de votre table (article pour l'exemple ci-dessus, là où sont stockées les informations), il a tout le temps qu'il veut pour la trouver. Il faut donc faire attention à produire un code PHP efficace afin d'éviter de genre de sérieux problèmes.

Comment s'en prémunir ?

Requêtes préparées

Introduisons rapidement PHP : l'Hypertext Preprocessor est un langage de programmation côté serveur principalement utilisé pour le dev. Web. Repassons sur nos requêtes préparées...

Pour se protéger des injections SQL, nous pouvons donc utiliser la fonction prépare de PDO.
PDO est une extension PHP qui fournit une interface uniforme pour accéder à des BDD. C'est une abstraction de BDD qui permet aux développeurs d'écrire du code PHP indépendant du type de base de données sous-jacente.

Pour exemple :
Une méthode séparant les valeurs envoyées. Pour la première donnée, nous n'envoyons pas la valeur que nous voulons récupérer mais une clef temporaire commençant par deux points :

```php
$sql = "SELECT `id`,`title` FROM `article` WHERE `title` LIKE '%:search%'";
$data = [
  'search' => $_POST['search']
];
$prep = $conn->prepare($sql);
$result = $prep->execute( $data );
```

En résumé, il convient de nettoyer les données d'entrées avant de les utiliser dans des requêtes SQL.
Comme autre choix si l'utilisation de PDO n'est pas possible, il reste des fonctions d'anciennes versions de PHP toujours utilisables mais moins optimales :

PHP 5

```
function sanitize_string($str) {
        if (get_magic_quotes_gpc()) {
                $sanitize = mysqli_real_escape_string(stripslashes($str));
        } else {
                $sanitize = mysqli_real_escape_string($str);
        }
        return $sanitize;
}
```

Plus laborieux à exécuter, il y a une méthode qui se base sur une utilisation de liste blanche. Soit l'autorisation des caractères et des valeurs attendues uniquement.

Conception non sécurisée

Une conception sécurisée repose sur un ensemble de bonnes pratiques de sécurité, ou encore une notion de risque, qui doivent être pensées dès le début de la conception de l'application. Toutefois, une conception parfaitement sécurisée peut présenter des défauts lors de son implémentation, créant alors des vulnérabilités. La correction de l'implémentation reste toutefois plus facilement envisageable.
Avant de créer le projet, vous devez vous questionner sur les réels fondements du dit projet :

- A quelle menace mon application serait exposée ?
- Utilise-t-elle des services tiers externes ?
- Communique-t-elle avec des services internes ?
- Aura-t-elle différents privilèges ou droits, basés sur différents rôles ?

Ce sont des questions à prendre en compte. En cas de penetration testing (pentesting), il faut essayer de repérer les défauts de conception provoquant les problématiques de sécurité.

Prévenir les failles

Stockage des informations d'identification sans protection : Il est impératif de concevoir la gestion des identités et des accès utilisateurs (mot de passe ou système de génération de certificats) de manière extrêmement fiable.

Transgression des limites de confiance : Ces violations surviennent lorsque des données fiables et non fiables sont stockées dans le même espace de stockage, par exemple. Les développeurs peuvent ne pas toujours faire la distinction entre ce qui constitue une donnée fiable ou non fiable. De plus, le backend peut faciliter un échange non sécurisé de données si l'application ne dispose pas d'un mécanisme de validation des entrées fiables.

Révélation d'informations sensibles via les messages d'erreur : La plupart des applications génèrent des messages d'erreur pour informer les utilisateurs d'une défaillance. Ces messages peuvent être trop verbeux et contenir des informations sensibles, comme des identifiants utilisateur, des mots de passe, ou d'autres données permettant à un attaquant de lancer des attaques ciblées (comme l'injection SQL).

Isolement ou compartimentation inappropriés : Ce problème se pose lorsque des composants ou utilisateurs avec des droits, privilèges ou autorisations d'accès différents ne sont pas correctement séparés. Une mauvaise isolation des processus, des ressources et des fonctionnalités de l'application peut augmenter considérablement la surface d'attaque.

Critères d'observabilité : L'observabilité se réfère à la capacité de comprendre l'état d'un système ou d'une application à partir des informations qu'il renvoie, comme des journaux ou des métriques.
Gestion des rôles : L'architecture d'un système, qu'il s'agisse d'une application ou d'une infrastructure, doit être conçue en tenant compte des différents rôles prévus ou configurables. Une matrice d'habilitation basée sur les rôles peut servir de point de départ à cette réflexion.

S'en protéger

Mettre en place un cycle de vie de développement : Utiliser des frameworks comme OpenSAMM pour formuler des stratégies de sécurité des applications adaptées aux risques spécifiques auxquels elles sont confrontées.

Mettre en place des tests unitaires et d'intégration automatisés : Une couverture de code complète avec des tests automatisés, associée à une logique de DevSecOps, permet de réduire les risques.

Gérer les ressources de manière granulaire : Collecter et évaluer rigoureusement les besoins métier, en mettant en place une gestion sécurisée des droits d'accès.

Séparer les couches systèmes et réseau de l'application : En subdivisant le déploiement en sous-systèmes modulaires ou en réseaux virtuels, il devient plus simple de limiter l'accès à des données et ressources spécifiques, rendant ainsi plus difficile l'extension de l'accès en cas d'attaque.

Pour finaliser cette section, il est important de mettre en place des tests unitaires.

Mauvaise configuration de sécurité

Vulnérabilités d'une application

L'application peut être vulnérable si :

- Elle n'a pas fait l'objet d'un durcissement de la sécurité approprié sur l'ensemble des couches protocolaires applicatives, ou si les permissions sont mal configurées sur les services cloud ;
- Des fonctionnalités inutiles sont activées ou installées (ex : des ports, des services, des pages, des comptes ou des privilèges inutiles) ;
- Les comptes par défaut et leurs mots de passe sont toujours activés et inchangés ;
- Le traitement des erreurs révèle aux utilisateurs des traces des piles protocolaires ou d'autres messages d'erreur laissant transpirer trop d'informations ;
- Pour les systèmes à jour ou mis à niveau, les dernières fonctionnalités de sécurité sont désactivées ou ne sont pas configurées de manière sécurisée ;
- Les paramètres de sécurité dans les serveurs d'application, les frameworks applicatifs (ex : Struts, Spring, ASP.NET), les bibliothèques, les bases de données, etc. ne sont pas paramétrés avec des valeurs correctes du point de vue de la sécurité ;
- Le serveur n'envoie pas d'en-têtes ou de directives de sécurité, ou s'ils ne sont pas paramétrés avec des valeurs correctes du point de vue de la sécurité ;

Comment s'en prémunir

Il y a maintes manières de s'en prémunir :

- Un processus de durcissement répétable qui permette de déployer rapidement et facilement un autre environnement correctement sécurisé avec une configuration verrouillée. Les environnements de développement, d'assurance qualité et de production doivent tous être configurés de manière identique, avec des droits différents pour chaque environnement. Ce processus devrait être automatisé afin de réduire au minimum les efforts requis pour mettre en place un nouvel environnement sécurisé.

- Une plate-forme minimale sans fonctionnalité, composant, documentation et échantillon inutile. Supprimer ou ne pas installer des fonctionnalités et frameworks inutilisés.

- Une tâche pour revoir et mettre à jour les configurations appropriées à tous les avis de sécurité, toutes les mises à jour et tous les correctifs dans le cadre du processus de gestion des correctifs. En particulier, examiner les permissions de stockage dans le Cloud.

- Une architecture d'application segmentée qui fournit une séparation efficace et sécurisée entre les composants ou les environnements hébergés, avec de la segmentation, de la mise en conteneurs ou l'utilisation de groupes de sécurité dans le Cloud (ACL).

- L'envoi de directives de sécurité aux clients, par exemple En-têtes de sécurité.

- Un processus automatisé pour vérifier l'efficacité des configurations et des réglages dans tous les environnements.

Composants vulnérables et obsolètes

La société SOCRadar, spécialisée dans l'analyse de menaces, estime la fuite des données à plus de 2To. La fuite de données, baptisée "BlueBleed" par SOCRadar touche les données de 111 pays.
Les informations étaient stockées dans des fichiers datant de 2017 à aout 2022. Le nom de Blue Bleed est par la suite étendu à d'autres fuites de données provenant de Amazon AWS S3 et Google Cloud Storage.

Notifions qu'elles ont plus que triplé entre 2013 et 2023, compromettant 2,6 milliards de dossiers personnels. Nous pouvons envisager un chiffre encore pire pour ces futures années.
Ces fuites de données ont un prix. Qui peut monter très cher qui plus est.
Plus d'explications s'imposent.

XML External Entities

Le langage XML (eXtensible Markup Language) créé pour stocker, partager et transporter des données entre système, indépendant de la plateforme et du langage. C'est tout simplement un moyen de formaliser du texte tout comme l'HTML. Il est assez similaire à celui-ci.
Exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<breakfast_menu>
  <food>
    <name>Belgian Waffles</name>
    <price>$ 5.95</price>
    <description>Two of our famous Belgian Waffles with plenty of real maple syrup</description>
    <calories>650</calories>
  </food>
```

Quelque chose nous intéresse ici : **les entités**.

Elle est, globalement, l'équivalent XML d'une variable et permet de définir des éléments que l'on pourra placer ensuite ailleurs. Une de leur particularité est qu'il est possible de faire référence à une entité externe au document ou alors un fichier local / sur serveur distant.
C'est précisément là que se situe le cœur d'une XXE.

En titre d'exemple, il est commun que certains sites web se servent de données utilisateurs pour créer des fichiers XML (ou reposant sur du XML, tel que des PDF), avant de renvoyer ceux-ci à l'utilisateur.
Un inconnu malveillant est donc dans la capacité d'injecter dans le code XML des références à des entités externes, notamment des fichiers sensibles se situant sur un serveur. Puisque c'est le serveur qui va lire le SML, il va essayer de résoudre les entités externes avant de les incorporer au document final, qui sera ainsi renvoyé au client en contenant des données sensibles.

Avant de continuer, le parsing se définit par l'analyse d'un document XML par un programme informatique. Sachez qu'un service est ébranlable s'il comporte ces 3 comportements :

- Un utilisateur doit être dans la possibilité de manipuler du contenu XML qui sera ensuite parsé par le serveur.

- Le parseur XML autorise la définition et l'utilisation d'entité.
- Le parseur XML doit parser et interpréter les références externes dans les entités.

Prenons également connaissance de la XEE (XML Entity Expansion) qui elle est une attaque par déni de service qui consiste à créer des milliards de références à des entités, ce qui consomme beaucoup de ressources et peut ralentir, voire bloquer un système.

Exemple de XEE :

```
<!DOCTYPE data [
<!ENTITY a0 "dos" >
<!ENTITY a1 "&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;">
<!ENTITY a2 "&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;">
<!ENTITY a3 "&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;">
<!ENTITY a4 "&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;">
<!ENTITY a5 "&a4;&a4;&a4;&a4;&a4;&a4;&a4;&a4;&a4;&a4;">
<!ENTITY a6 "&a5;&a5;&a5;&a5;&a5;&a5;&a5;&a5;&a5;&a5;">
<!ENTITY a7 "&a6;&a6;&a6;&a6;&a6;&a6;&a6;&a6;&a6;&a6;">
]>
<data>&a7;</data>
```

La différence avec une XXE est qu'une XEE n'utilise pas de référence externe.

Se protéger

Pour se protéger de ces attaques, les options suivantes s'imposent :

- Ne pas utiliser de données utilisateurs dans les documents XML. Cette option peut être viable si les données dynamiques des documents sont indépendantes des utilisateurs. Le code SML en lui-même ne dois jamais transiter par le client
- Interdire la définition d'entités ou d'entités externes. La plupart des interpréteurs XML incluent ce genre d'options, que ce soit lors de la configuration, de l'initialisation ou de

l'utilisation. Lisez attentivement la documentation de votre interpréteur pour le configurer et le sécuriser.

Les XXE sont des vulnérabilités assez méconnues et dont l'impact peut être très grave. La versatilité du XML peut rendre difficile leur détection. Tout de même, pas de panique, une fois le risque identifié, sa prévention est normalement et généralement clair et efficiente.

Identification et authentification de mauvaise qualité

Ce chapitre sera plus court que les autres. Lorsque l'on parle d'identification de mauvaise qualité, c'est tout simplement que l'application en question laisse place à une porte ouverte aux données de ses utilisateurs.

Données de l'utilisateur

Prenons comme premier point de vue le côté utilisateur. Nous le savons déjà, depuis quelques couples d'années, une forte sensibilisation est effectuée concernant les mots de passe.
Qu'il fasse un certain nombre de caractère, qu'il contienne un caractère spécial, un nombre ou encore une lettre en majuscule. Pour que ces utilisateurs n'aient pas à se soucier de cela, c'est à nous, développeurs de leur rappeler. On prémunit également quelquefois de rentrer des noms familiers, des dates de naissance etc. Pour ce genre de cas, nous pouvons utiliser des listes de mots prédéfinis que l'utilisateur ne pourra alors plus rentrer.

Et une fois qu'il est rentré, quoi en faire ?

Stockage des mots de passe

Les stocker de manière brute dans une base de données. Vu sous cet angle, lorsque l'utilisateur s'authentifie, il suffit de vérifier si les informations concordent dans la base de données. Mais en cas de cyberattaques, les données sont récupérées telles qu'elles sont rentrées et ce n'est pas envisageable.

Testons de les chiffrer cette fois-ci. Nous devons donc les déchiffrer pour les comparer dans la base. Cependant, cela inclut que la clé ne doit pas être loin et ceci est une mauvaise pratique. Si la clé est trouvée, nous revenons à la case départ. Le chiffrement ne résout pas le problème de confidentialité des données mais le transforme en problème pour stocker les clés.
Nous en venons donc à la dernière et meilleure solution, le hachage.

Hachage

Les données seront décomposées et transformées en une nouvelle forme abrégée appelée valeur de hachage.

Il se fait en 3 éléments :

- Fonction de hachage : c'est l'algorithme de chiffrement qui va décomposer, résoudre et transformer complètement les données, qu'elles fassent la même taille ou non en chaines de longueurs égales.
- Valeur de hachage : Résultat de cette fonction.
- Table de hachage : Les données que nous avons érigées sont placées dans la base de données sous forme de tables de hachage. Nécessitant moins d'espace que les données d'origine qui plus est.

Grâce à cela, il est déjà possible de stocker davantage de données.
Comme méthode d'authentification, il est utilisé pour générer des signatures numériques, appelées empreintes digitales numériques. Cela confirme l'intégrité de la communication entre l'expéditeur et le destinataire.
En termes de sûreté, il n'y a pas de méthode plus sécuritaire.

Attaques

Le risque 0 n'existe pas. Des personnes malveillantes peuvent faire correspondre des listes de hachages volées avec des tables arc-en-ciel. Ce sont des listes avec des valeurs de hachage volées et des données d'accès qui leur sont attribuées.
Nous pourrions alors constater une attaque par force brute, consistant à tester chaque combinaison possible. Par conséquent, il est important d'implémenter un système qui incorpore un nombre d'essai à chaque connexion.
Il faut donc rester à jour sur les algorithmes de hachage nouveaux, ou actuels du moins.

Manque d'intégrité des données du logiciel

Nous faisons ici référence à l'exactitude, l'exhaustivité et la cohérence globale des données. Mais nous rejoignons notre sujet sur la sûreté des données. Il faut qu'elles soient conformes à la réglementation (par exemple la conformité au RGPD).
Nous l'avons vu plus haut mais lorsque l'intégrité des données est vérifiée, les informations stockées dans la base de données restent complètes et fiables. Isolément de leur durée de stockage et du nombre de fois que l'on y accède.

Type d'intégrité des données

Il y a deux types : physique et logique.

- **Physique** : implique la préservation de leur intégrité et de leur exactitude tout au long de leur stockage et de leur récupération. En cas de catastrophe naturelle, de panne de courant ou de violation des systèmes de base de données, cette intégrité peut être compromise. De plus, des erreurs humaines, une détérioration du stockage et d'autres problèmes similaires peuvent également entraver la capacité des responsables du traitement des données, des administrateurs système, des développeurs d'applications et des auditeurs internes à garantir la précision des données.
- **Logique** : préserve la constance des données tout au long de leurs diverses manipulations au sein d'une base de données relationnelle. Elle assure également une protection contre les erreurs humaines et les intrusions informatiques, bien que son fonctionnement diffère de celui de l'intégrité physique. Les quatre types d'intégrité logique sont les suivants.

En réalité ce sont les “principales” mais il y existe d’autres.
L’intégrité de l’entité s’appuyant sur la création de clés primaires pour garantir le fait que ces données ne soient pas stockées plus d’une fois. L’intégrité référentielle concernant l’utilisation de clés étrangères garantissant que seuls les changements, appropriés sont effectués. Ne négligeons pas celle définie par l’utilisateur. Il appliquera ses propres règles et contraintes pour satisfaire ses besoins particuliers.

Sécurité des données

L’intégrité n’est pas la sécurité. La sécurité est l’ensemble des mesures prises pour empêcher la corruption des données. C’est à ne pas confondre.

Conformité au RGPD

Le Règlement Général sur la protection des Données implique le respect des principes et des obligations énoncés dans ce règlement européen. Il se base sur plusieurs éléments :

- Consentement et transparence : les organisations doivent fournir des informations transparentes sur la manière dont des données seront utilisées.
- Droits des individus : accorde aux individus le droit d’accès, de rectification, d’effacement, de limitation du traitement. C’est notamment grâce à cela qu’il est possible de retirer un résultat sur google qui nous concerne et qu’on ne souhaite pas garder en ligne (c’est un exemple).
- Responsabilité et sécurité des données : Les organisations sont tenues de mettre en place des mesures de sécurité appropriées pour protéger les données personnelles.
- Notification des violations de données : En cas de violation de données personnelles, les organisations sont tenues de notifier les autorités de contrôle compétentes et, dans certains cas, les individus concernés, dans les délais prescrits.

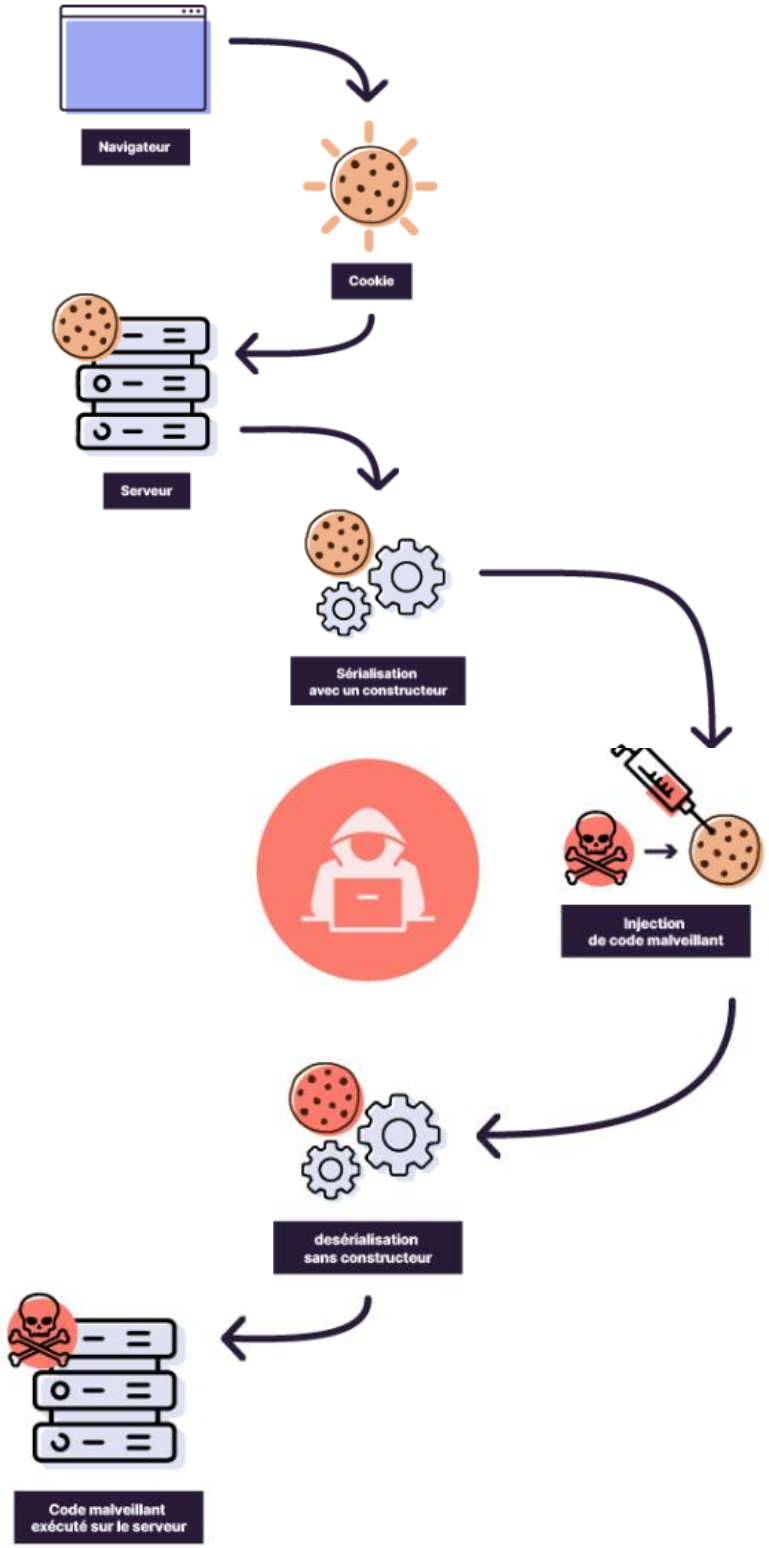
Risque

Autre exemple, le plug-in pour le CMS WordPress “X-WP-SPAM-SHIELD-PRO”. Derrière un plug-in de sécurité pour les sites web sous WordPress se cachait une porte dérobée. Cette backdoor permettait un accès aux données du serveur, avec un risque de potentiellement permettre l’accès à des données confidentielles d’entreprise, ou encore l’augmentation de privilège d’un utilisateur, lui permettant un accès administrateur. Il y a beaucoup d’exemple comme celui-ci.
De régulières vérifications permettraient d’éviter ce problème :

- Mise à jour : Vérifier ces dates. Il se peut qu’un outil laisser à l’abandon laisse des vulnérabilités
- Vérifier ses sources
- Importance de l’outil : Vérifier que l’outil est réellement nécessaire.
- A quelles ressources le composant prend accès : Les permissions demandées sont-elles cohérentes avec la fonction ?

Désérialisation non sécurisée

C'est une vulnérabilité qui se produit lorsque des données non fiables sont utilisées pour abuser de la logique d'une application, causer un déni de service (DoS, Deny of Service), ou même exécuter du code arbitraire.
Prenons un objet, comme un nom d'utilisateur et un mot de passe, et le mettre dans la base de données. Pour pouvoir être stocké, l'objet devra être converti en flux d'octets pour être transporté à travers le réseau afin d'accéder à la base de données. C'est ce qu'on appelle la sérialisation. Lorsqu'il y a un appel pour ce même objet dans la base de données, il doit être désérialisé (conversion du flux d'octets en objet) avant son utilisation.
Ce processus de sérialisation/désérialisation des objets ne se produit pas seulement avec les bases de données. Un objet peut changer son état en un flux d'octets lorsqu'il est stocké dans un fichier, d'ordinateur à ordinateur ou en se déplaçant sur le réseau.
Ces objets peuvent être des cookies, des flux vocaux, des jetons ou des fichiers cache, par exemple.
La désérialisation non sécurisée est donc la capacité d’un attaquant à changer l’état du code pendant sa conversion en binaire. Elle peut être évitée en créant des contrôles sur l’état du code.
Cependant, attention ! Le risque peut provenir d’une bibliothèque externe.



(Source du schéma openclassroom : https://openclassrooms.com/fr/courses/6179306-securisez-vos-applications-web-avec-lowasp/8169666-garantissez-l-integrite-de-vos-donnees-et-logiciels)

Carence des systèmes de contrôle et de journalisation

Sans journalisation et surveillance, les brèches ne peuvent être détectées. La journalisation fait référence à un enregistrement systématique et chronologique des événements. Sans cela :

- Les traces d'audit comme les accès réussies ou échoués et les transactions sensibles ne sont pas enregistrées
- Les alertes et les erreurs générées ne sont pas enregistrées.
- Les journaux ne sont stockés que localement

Dans le cadre d'une utilisation du serveur web Apache HTTPD, des outils comme apache Scapl Webforencik vont permettre de parser le fichier de log pour en extraire les informations importantes, ou des tentatives, réussies ou non, d'intrusion.

SOC

Security Operations Center est chargé de surveiller et protéger les systèmes informatiques, les réseaux et les données d'une organisation.

Avantages d'un SOC

- Surveillance et analyse ininterrompue d'activités suspectes
- Amélioration des temps de réponse
- Plus de transparence et de contrôle sur les opérations de sécurité.

Inconvénients

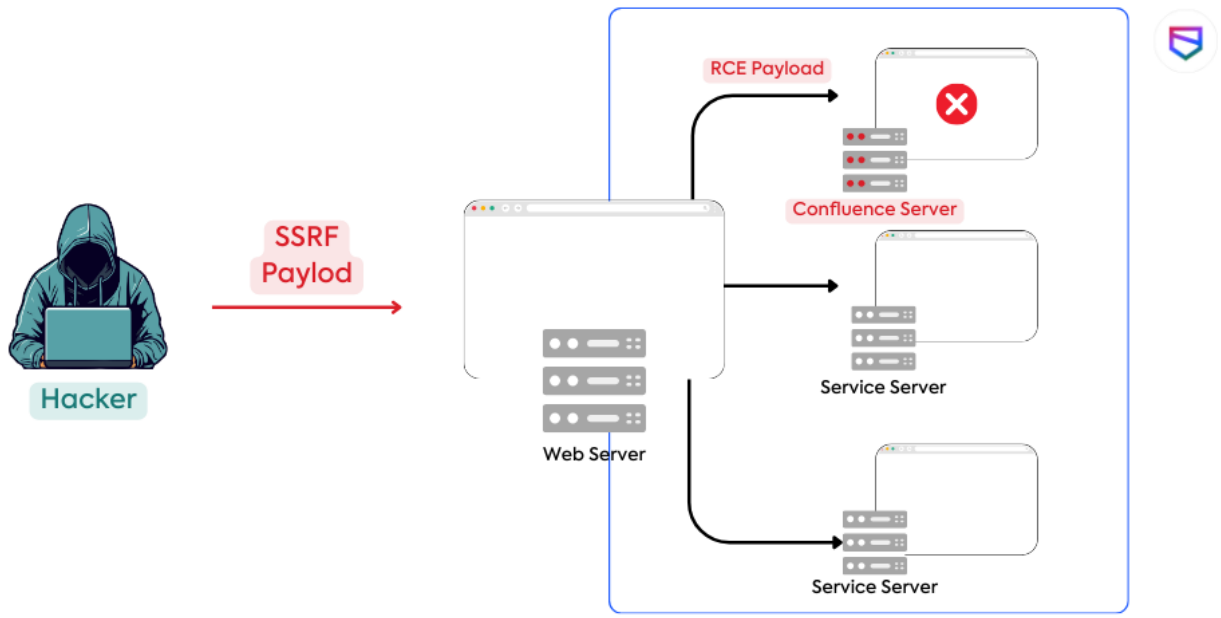
Dans le cadre d'une entreprise conséquente, elle peut se retrouver en face de beaucoup de difficultés :

- Volume : Le nombre d'alertes de sécurité peut surprendre. Des ressources humaines se doivent de catégoriser prioriser et répondre pour anticiper les menaces.
- Complexité : Plus l'entreprise est grande, plus il y aura d'ouvertures
- Coût

En résumé, le SOC est une manière de résoudre le problème de carence des systèmes de contrôle et de journalisation.

Falsification de requête côté serveur

Passons à cette ultime et dernière vulnérabilité de l'OWASP Top 10, de son acronyme SSRF est une vulnérabilité dans les applications web sans vous étonné par laquelle un attaquant peut effectuer des requêtes HTTP via le serveur. Il peut s'en suivre une communication avec tout service interne du réseau du serveur généralement protégé par des pare-feu.



Qu'est ce qui rend un code vulnérable à une SSRF ?

Communément, c'est la confiance aveugle envers les actions d'un utilisateur. Prenons comme exemple un code PHP qui prend des informations envoyées dans une URL sans vérification derrière. C'est un problème car un attaquant pourrait envoyer des requêtes GET à n'importe quel site Web via le serveur.

Ports

Notion importante qui n'avait pas encore été abordée. C'est un point virtuel où les connexions réseau commencent et se terminent. Ils sont basés sur des logiciels et gérés par le système d'exploitation.

Il y en a différents types, les ports matériels et logiciels. Je n'aborderai ici que les ports logiciels car les ports matériels ne nous intéressant pas dans notre situation.
Voyons un port comme une porte. Si une application a besoin de communiquer en réseau, elle ouvrira sa porte au numéro unique, et enverra ses informations à une autre porte numérotée X d'un ordinateur distant.

La plage s'étend de 0 à plus de 60000 ports mais pour ne citer que les principaux :

Numéro	Nom du port
21	ftp
23	telnet
80	HTTP
443	HTTPS
3306	MySQL DB

Exemple

Prenons une page qui nous donne la possibilité de saisir une URL. Un attaquant a la possibilité de rentrer l'adresse IP d'un hôte local avec le numéro de n'importe quel port et vérifier si celui-ci exécute un service. Par exemple rentrer http://127.0.0.1:3306 lui donnera la confirmation ou non qu'une base de données soit en cours d'exécution.
Même en appliquant un filtrage qui stoppe la requête si l'utilisateur rentre "localhost" ou "0.0.0.0" (équivalent d'adresses ip pour les hôtes locaux), cela ne demeure pas suffisant. Car il est également possible d'essayer avec l'adresse IPv6 de l'hôte local.

Types de SSRF

Content-Based : la ressource chargée est retournée par le serveur.
Boolean-Based : la réponse du serveur est différente que la ressource existe ou non.
Time-Based : Lorsque la réponse du serveur reste la même que la ressource existe ou non, c'est le temps de cette réponse qui peut varier avec plus d'importance.
Error-Based : erreur permettant de déduire les ressources existantes ou non.

L'exemple que nous avons vu plus haut était de ce dernier type.

Comment s'en prémunir ?

Il faut toujours être précautionneux lorsque l'on interagit avec les données d'un utilisateur. Évitons la "black list" car nous ne pourrons jamais bannir l'intégralité des contenus pouvant contenir une faille. Optons plutôt pour une "white list" qui n'accepte que les choses dont nous sommes sûrs.
Comme vu dans l'exemple ci-dessus, toute entrée contenant des adresses ip d'hôte local est prohibé. Il faut également définir les protocoles autorisés (Notifions que même avec cela, il est possible d'utiliser des services externes comme nip.io pour scanner le réseau interne).

Nous pouvons imposer un schéma d'URL précis, le port et la destination avec une liste positive d'autorisation, ne pas envoyer de réponses brutes aux clients et enfin désactiver mes redirections HTTP.

Hébergement

Maintenant que votre site est finalisé, vous avez la possibilité de l'héberger localement, que ce soit sous Windows ou sous linux (nous nous concentrerons uniquement sur Windows dans notre cas). Cela signifie que votre site sera votre propre serveur et devra être configuré.

Hébergement sur Windows

L'utilisation de WampServer est envisageable. De nombreuses options sont disponibles comme la configuration de MySQL et d'autres.

A part une personne venant du même réseau que vous, elle ne pourra pas accéder au site.
Pour corriger ceci, il est nécessaire de configurer le fichier httpd.conf (fichier de configuration Apache).
Il faut ajouter ceci au fichier :

- Order Allow, Deny
- Allow from all

Après cette étape, un nom de domaine est inéluctable.
Une fois celui-ci acquis, créer un fichier texte le dossier « extra » d'apache et ajouter ceci :

- ServerAdmin exempledemail@gmail.com
- DocumentRoot « Chemin du dossier »
- SerName mysite.local
- ErrorLog « logs/example.com.log"
- CustomLog "logs/example.com-access.log" common

Notre site est désormais disponible sur Internet.

Plan de diffusion

Cette veille technologique pourrait servir à nombre de développeurs expérimentés (dans un cadre tout de même restreint) ou non. Comme la réalisation le présage, un répertoire GitHub permet de faire connaitre le projet à plus grand nombre de personnes.

Malgré ses termes techniques, il reste un côté vulgarisateur. Je pourrais donc le partager sur des réseaux sociaux comme X.

Création du site WEB

Je ne possède pas de serveur. Je ne peux donc pas proposer un site libre à tous lier à une base de données et sécuriser. Cependant, la réalisation du site web dédié (regroupant toutes les informations sur cette veille technologique) est disponible au lien suivant : Nayxooo.github.io (hébergé par les serveurs de GitHub par lesquels il est impossible d'effectuer quelconque configuration).

En parcourant le site, vous aurez la possibilité de vérifier dans le slider à gauche, l'onglet "Partie locale", les étapes réalisées pour appliquer les manœuvres de sécurité localement.
Les failles présentes dans le top 10 de 2017 comme les failles XSS par exemple n'ont pas été abordées mais cela aurait pu être intéressant à aborder.

Quelles stratégies de gestion des incidents de sécurité peuvent être mises en place pour réagir rapidement aux attaques et aux violations de la sécurité ?

Quels outils et techniques peuvent être utilisés pour effectuer des tests de sécurité approfondis et des audits périodiques ? Cette question renvoie au pentesting mais pourrait être abordée dans une autre étude.

Conclusion

Nous avons vu ensemble les 10 catégories constituant l'OWASP TOP10 2021 et à quel point il est complexe de rester à jour sur les attaques de plus en plus sophistiquées.

Il est important de rester à jour sur les futures potentielles attaques non découvertes. Car peu importe le niveau de sécurité d'une application, il y aura toujours une personne malveillante pour trouver une faille. La réaction/maintenance est donc encore plus importante que la sécurisation elle-même.
Malgré ce fait, nous nous devons de ne pas divulguer les données de nos utilisateurs. Prendre conscience et s'éloigner de ces risques est impératif.

C'est difficile de se rendre compte de l'étendue de ce domaine. Nous n'avons cité que 10 menaces de la liste de 2021, rien que la liste de 2017 en présentait d'autres et qu'en est-t-il de celles non spécifiées. La sécurité des applications web est un cadre vraiment large. J'en tire personnellement une nouvelle détermination à rester à jour sur ces sujets et sur ces nouvelles failles. La quantité d'information est angoissante, mais est richement instructive. De plus, la veille sur ce genre de sujet permet d'être transposer au-delà du développement web (Réseau et cybersécurité).

De nos jours, toutes les entreprises ont besoin d'un site web pour promouvoir leur produit ou des serveurs pour stocker les fichiers. Il est donc important de les sécuriser.

Sources

HTML

https://developer.mozilla.org/fr/docs/Learn/Getting_started_with_the_web/HTML_basics

https://openclassrooms.com/fr/courses/1603881-creez-votre-site-web-avec-html5-et-css3/8061261-creez-votre-premiere-page-web-en-html

https://habefast.ch/glossaire/html/

https://grafikart.fr/formations/html

Non utilisation des balises HTML pour la mise en forme :
https://fr.semrush.com/blog/balises-structurelles-html-semantique/

CSS

https://developer.mozilla.org/fr/docs/Web/CSS

https://fr.wikipedia.org/wiki/CSS

https://blog.hubspot.fr/website/langage-css

https://www.24joursdeweb.fr/2017/choisir-sa-spec-de-positionnement-css/

JavaScript

https://developer.mozilla.org/fr/docs/Web/JavaScript

Framework

https://www.vigicorp.fr/blog/qu-est-ce-qu-un-framework/

https://php.developpez.com/actu/80156/Framework-or-not-framework-Decouvrez-la-vision-personnelle-d-un-developpeur-PHP/

https://www.pure-illusion.com/lexique/definition-de-framework

OWASP TOP 10

https://owasp.org/Top10/fr/

Contrôle d'accès défaillants

https://openclassrooms.com/fr/courses/6179306-securisez-vos-applications-web-avec-lowasp/6520701-empechez-l-exploitation-des-controles-d-acces-defaillants

https://outpost24.com/fr/blog/les-vulnerabilites-des-controles-d-acces/

https://www.digicomp.ch/blognews/2022/11/29/owasp-top-10-failles-securitaires-applications-web

Défaillances cryptographiques

https://www.vaadata.com/blog/fr/chiffrement-des-donnees-et-defaillances-cryptographiques-top-10-owasp-2/

https://www.acceis.fr/failles-crypto/

https://securite.developpez.com/tutoriels/dix-principales-erreurs-en-cryptographie/

Injection

https://www.oracle.com/fr/security/injection-sql-attaque/

https://openclassrooms.com/fr/courses/7727176-realisez-un-test-dintrusion-web/7917166-attaquez-la-base-de-donnees-avec-les-injections-sql

https://www.codeur.com/tuto/php/se-proteger-injections-sql/

Injection SQL : guide complet par Rana Khalil

https://www.youtube.com/watch?v=1nJgupaUPEQ&list=PLuyTk2_mYISLaZC4fVqDuW_hOk0dd5rlf

Conception non sécurisée

https://openclassrooms.com/fr/courses/6179306-securisez-vos-applications-web-avec-lowasp/8169361-securisez-une-application-des-sa-conception

https://fr.parasoft.com/blog/sensitive-data-exposure-owasp-top-10/

Mauvaise configuration de sécurité

https://www.vaadata.com/blog/fr/mauvaise-configuration-de-securite-owasp-top-10-5/

https://www.crowdstrike.fr/cybersecurity-101/security-misconfiguration/

Composants vulnérables et obsolètes

https://www.vaadata.com/blog/fr/composants-vulnerables-et-obsoletes-owasp-top-10-6/

https://www.checkpoint.com/fr/cyber-hub/cloud-security/what-is-application-security-appsec/owasp-top-10-vulnerabilities/

Identification de mauvaise qualité

Tendance sur les algorithmes de hachage :

https://forum.huawei.com/enterprise/fr/Algorithmes-De-Hachage/thread/706319051447484416-667480999820406784

https://www.cnil.fr/fr/definition/force-brute-attaque-informatique

https://www.cnil.fr/fr/les-conseils-de-la-cnil-pour-un-bon-mot-de-passe

https://www.ionos.fr/digitalguide/sites-internet/developpement-web/hachage/

Manque d'intégrité des données du logiciel

https://openclassrooms.com/fr/courses/6179306-securisez-vos-applications-web-avec-lowasp/8169666-garantissez-l-integrite-de-vos-donnees-et-logiciels

https://www.varonis.com/fr/blog/integrite-des-donnees-quest-ce-que-cest-comment-la-preserver

https://www.talend.com/fr/resources/what-is-data-integrity/

https://www.cnil.fr/fr/reglement-europeen-protection-donnees

Carences des systèmes de contrôlé et de journalisation

https://www.logpoint.com/fr/blog/security-operations-center-soc-2/

https://openclassrooms.com/fr/courses/6179306-securisez-vos-applications-web-avec-lowasp/8169731-rendez-possible-la-detection-dattaques-et-les-investigations-grace-aux-journaux-et-alertes

https://fr.scribd.com/document/539582565/audit-controles-v1-4

https://www.vumetric.com/fr/blogue/owasp-top-10-a09-security-logging-and-monitoring-failures/

Falsification de requête côté serveur
https://www.pedagogie.ac-nice.fr/sti-voie-pro/attachments/article/47/PORT%20INFORMATIQUE%20v2.pdf

https://securityboulevard.com/2023/11/penetration-testing-for-server-side-request-forgery-ssrf-in-e-commerce-platforms/

https://www.geeksforgeeks.org/server-side-request-forgery-ssrf-in-depth/

https://www.vaadata.com/blog/fr/comprendre-la-vulnerabilite-web-server-side-request-forgery-1/

https://tryhackme.com/

Hébergement
https://www.hebergementweb.info/comment-heberger-votre-propre-site-web-localement-plus-5-avantages-et-inconvenients/

https://www.ionos.fr/digitalguide/serveur/configuration/installer-un-serveur-apache-nos-conseils/