الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière
Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Sciences et Technologie de l'Information et de la Communication (STIC)

Thème :

Détection des attaques XSS pour la sécurité des sites Web : Une approche basée sur l'apprentissage automatique

Encadré par :

Dr. HANNOUSSE ABDELHAKIM

Présenté par :

GHERAIBIA SABRINA

Juin 2022

(بسم الله الرحمن الرحيم)

Remerciement

Avant tout, je remercie Dieu qui m'a donné la capacité et la patience d'accomplir ce travail malgré tous les obstacles.

J'exprime mes sincères remerciements au superviseur ***Mr HANNOUSSE Abdelhakim*** *pour son soutien continu, sa disponibilité, ses conseils, sa contribution au projet et l'aide constante qu'il m'a apportée*

Mes remerciements vont également aux jurys qui ont accepté de lire et d'évaluer ce travail.

Je tiens à remercier tout particulièrement tout le personnel enseignant et administratif du Département d'informatique, pour leur sympathie et aussi pour leurs précieux conseils et instructions durant mes années de premier cycle.

Enfin, nous remercions toutes les personnes qui ont contribué de loin ou de près à la concrétisation de ce travail.

** إهـــــداء **

الى من وضع المولى سبحانه وتعالى الجنة تحت قدميها ووقرها في كتابه العزيز

أمي الحبيبة

إلى خالد الذكر الذي وافته المنية وكان خير مثال للطيبة لم يتهاون يوم في اسعادي..

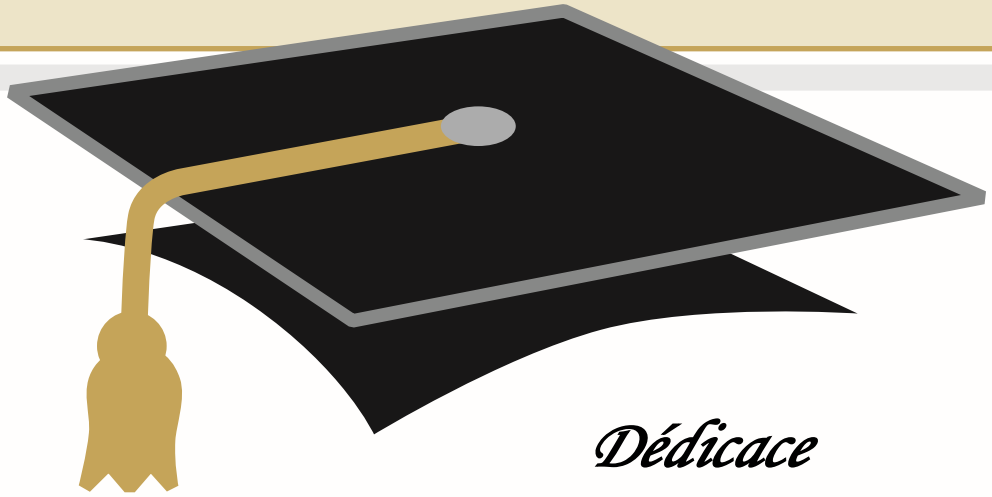
أبي الموقر

إلى اخوتي أصدقائي الذين أحبهم وأحترمهم...

إلى أساتذتي بجامعة قالمة وقسم الاعلام الالي خصوصا

أهدي لكم بحثي

صبرينة غرايبية



Dédicace

A mon cher père qui m'a enfournée de mon amour et affection qu'il trouve dans ce modeste travail, le fruit de ses efforts et ses sacrifices (j'espère que dieu lui fasse miséricorde).

À l'être le plus cher de ma vie, ma mère

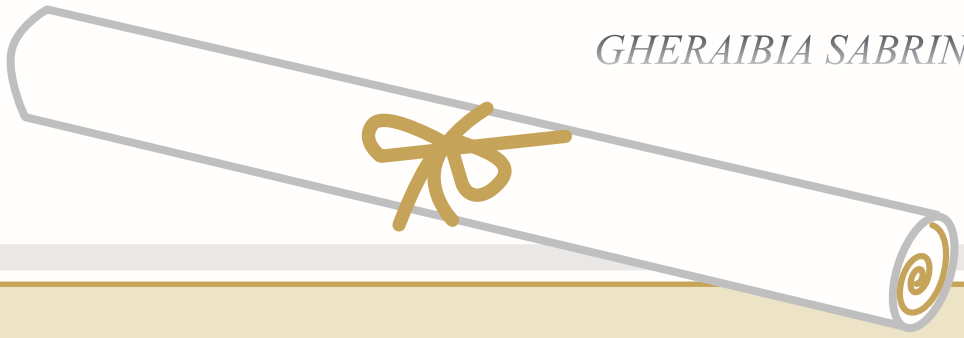
À Mes chers Frères.

À tous mes amis de promotion de 2éme année Master en Informatique,

Toute personne qui occupe une place dans mon cœur.

Je dédie ce travail à tous ceux qui ont participé à ma réussite en particulier ***à HANNOUSSE Abdelhakim****, et le futur Docteur, mon frère* ***BENSALAH Hazem****.*

GHERAIBIA SABRINA



Résumé

Le cross-site scripting (ou XSS en abrégé) est l'une des attaques les plus anciennes et les plus dangereuses menaçant la confidentialité des données et la navigation en ligne des applications Web de confiance. Depuis sa révélation, vers la fin de l'année 1999 par des ingénieurs en sécurité de Microsoft, plusieurs approches, techniques et outils ont été développés dans le but de sécuriser la navigation web et de protéger les applications web contre ce type d'attaques. Le problème s'est aggravé avec l'émergence rapide des technologies Web avancées et des nouveaux styles de programmation. Dans ce projet, nous contribuons en explorant la capacité des modèles d'apprentissage automatique pour la détection des attaques XSS. Nous recueillons un ensemble de données de taille raisonnable, composé de scripts bénins et de vecteurs d'attaques XSS. Nous accordons une grande importance à l'ingénierie des caractéristiques et aux modèles supervisés traditionnels pour leur simplicité et leur interprétabilité. L'étude menée a montré que KNN surpasse tous les modèles existants et atteint plus de 99% en termes de mesure F1.

Mots-clés : Sécurité Web, Cross-site scripting; Détection des attaques XSS ; Apprentissage automatique.

Abstract

Cross-site scripting (or XSS for short) is one of the most ancient and dangerous attacks menacing the privacy of data and the online navigation of trusted web applications. Since it has been revealed in late 1999 by Microsoft security engineers, several approaches, techniques and tools have been developed in the aim to secure web navigation and protect web applications against such kind of attacks. The problem became worse with the rapid emergence of advanced web technologies and new programming styles. In this project, we contribute by exploring the ability of machine learning models for the detection of XSS attacks. We collect a reasonable in size dataset of benign scripts and XSS attack vectors. We put much focus on feature engineering and traditional supervised models for their simplicities and interpretabilities. The conduced study showed that KNN outperforms all existing models and reach more than 99% in term of F1-score.

Keywords : Web security, Cross-site scripting; XSS attack detection; Machine learning.

ملخص

تعد البرمجة النصية عبر المواقع (أو اختصارًا XSS) واحدة من أقدم وأخطر الهجمات التي تهدد خصوصية البيانات والتنقل عبر الإنترنت لتطبيقات الويب الموثوقة. منذ أن تم الكشف عنها في أواخر عام 1999 من قبل مهندسي الأمن في Microsoft ، تم تطوير العديد من الأساليب والتقنيات والأدوات بهدف تأمين التنقل على الويب وحماية تطبيقات الويب من مثل هذا النوع من الهجمات. تفاقمت المشكلة مع الظهور السريع لتقنيات الويب المتقدمة وأنماط البرمجة الجديدة. في هذا المشروع ، نساهم من خلال استكشاف قدرة نماذج التعلم الآلي على اكتشاف هجمات XSS. نقوم بتجميع مجموعة بيانات ذات حجم معقول من البرامج النصية الحميدة ونواقل هجوم XSS. نركز كثيرًا على هندسة الميزات والنماذج التقليدية الخاضعة للإشراف لبساطتها وتفسيراتها. أظهرت الدراسة التي تم إجراؤها أن KNN يتفوق على جميع النماذج الحالية ويصل إلى أكثر من 99 ٪ من حيث درجة F1.

الكلمات الدالة :

أمن الويب ، البرمجة النصية عبر المواقع ؛ اكتشاف هجوم XSS ؛ التعلم الالي .

TABLE DES MATIERES

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION GENERALE

INTRODUCTION GENERALE

Depuis l'apparition d'Internet, le nombre de ses utilisateurs a connu une augmentation phénoménale. Avec l'apparition du Web 2.0 [1], et notamment les réseaux sociaux, le nombre des utilisateurs et l'échange de données personnelles sont devenu démesurés et ne cessent d'augmenter. Aujourd'hui, l'utilisation des applications Web via Internet est devenue un moyen indispensable pour différentes organisations commerciales et gouvernementales pour réduire leurs frais de gestion et d'extension, accélérer leurs activités, améliorer la qualité de leurs services et atteindre le plus grand nombre de personnes ciblées. Les utilisateurs bénéficient également des grands avantages des services fournis en ligne. Cependant, ces gains ne sont pas sans risques ; les applications Web nécessitant l'enregistrement des utilisateurs via des formulaires de saisie, sont des cibles privilégiées pour différentes cyberattaques, mettant en danger leurs propres données confidentielles et celles de leurs utilisateurs.

Le cross-site scripting (XSS) est reconnu comme l'une des cyberattaques les plus dangereuses menaçant la sécurité de plusieurs types d'applications Web depuis 1999. Les attaques XSS sont des attaques basées sur l'injection de code ; les attaquants peuvent tirer parti des vulnérabilités, également appelées XSS, découvertes dans des applications Web de confiance, leurs plugins ou serveurs d'hébergement, pour injecter des scripts malveillants dans le but de les compromettre. De cette façon, les pages explorées des applications Web compromises sur les navigateurs des utilisateurs permettent aux attaquants d'élever leurs privilèges d'accès, révéler des informations utilisateur sensibles conservées par les navigateurs, telles que les noms d'utilisateurs victimes et leurs mots de passe [2]. Les attaques XSS sont principalement causées par le manque de mécanismes de contrôle appropriés des entrées utilisateurs dans les formulaires de saisie. Cela est dû à [2] :

(1) La méconnaissance des développeurs Web des problèmes et des pratiques de sécurité,
(2) Les exigences de changements rapides des services Web pour répondre aux besoins des nouveaux clients,

(3) La nature simple de l'attaque elle-même où toutes les entrées des utilisateurs doivent être considérées comme des vecteurs d'attaque potentiels.

Les attaques XSS peuvent être lancées depuis le client et/ou le serveur, ce qui les rend difficiles à traquer. La vulnérabilité XSS apparaît dans la liste des 10 principales vulnérabilités signalées par l'Open Web Application Security Project (OWASP) depuis 2003 jusqu'au dernier rapport publié en 2021 [W1].

Pour lutter contre ce type de cyberattaques, plusieurs efforts ont été réalisés pour le développement des outils qui permettront de détecter les attaques XSS. Notre projet de fin d'étude s'inscrit dans ce cadre, il s'intéresse à examiner un ensemble de modèles se basant sur les technologies d'apprentissage automatique. Spécifiquement, nous proposons un modèle d'apprentissage automatique en se basant sur des caractéristiques observées sur les différents vecteurs attaques reconnus. Pour concrétiser les fruits de nos recherches, nous développons une interface graphique simple qui met en pratique notre modèle et permet la détection de la méfiance des scripts donné par les utilisateurs. Afin d'atteindre ce but, le présent mémoire est structuré en trois chapitres :

Chapitre 1 : dans ce premier chapitre, nous définissons l'attaque et la vulnérabilité XSS et nous expliquons ses différents types et le principe de base de chaque type.

Chapitre 2 : dans ce chapitre, nous présentons les différentes solutions récentes proposées pour la détection des attaques XSS. Cette présentation nous permettra d'avoir un aperçu sur les derniers développements réalisés pour la détection des attaques XSS par l'utilisation de l'apprentissage automatique.

Chapitre 3 : dans ce chapitre, nous détaillons notre étude et solution proposée pour la détection des attaques XSS. L'étude menée dans ce chapitre, inclut les différentes phases élaborées pour la construction de notre modèle d'apprentissage automatique. Cela comprend : la collection des données, l'extraction des caractéristiques, le choix du meilleur classificateur et le déploiement du modèle construit par le biais d'une interface utilisateur simple.

CHAPITRE I.

INTRODUCTION AUX ATTAQUES XSS

CHAPITRE I.

INTRODUCTION AUX ATTAQUES XSS

Avec le développement rapide de la technologie Web, les individus, les organisations et même les objets sont devenus étroitement interconnectés. Cela a des bénéfices socioéconomiques énormes mais aussi des risques non-négligeables. Les cyberattaques ne cessent d'augmenter ; ils sont de plus en plus fréquents sur le Web. Pour mieux se protéger, il est indispensable de savoir comment ces attaques fonctionnent et les cibles préférées de chacune d'elles. Cela permet de concevoir des mécanismes efficaces pour se prémunir contre ces attaques et éviter leurs conséquences délétères.

En réalité, un large nombre de cyberattaques menacent la confidentialité des données et la navigation sur le Web. Dans ce chapitre, nous nous focalisons sur l'une des plus anciennes et dangereuses cyberattaques, appelée : *cross-site scripting* (XSS). Nous décrivons le principe de cette attaque avec les scenarios possibles que les attaquants exploitent pour affecter un large nombre d'individus. Nous expliquons, notamment, la cause derrière la popularité et la persistance de cette attaque depuis sa révélation vers la fin de l'année 1999.

1. Les risques de navigation sur le Web

Le monde d'Internet a été créé vers la fin des années 1980, depuis, il ne cesse d'évoluer. L'utilisation d'Internet n'est devenue popularisé qu'après la naissance du Web en 1989. Le Web est une application d'Internet qui permet, depuis un navigateur, de publier et consulter des pages et des documents stockés, sans se préoccuper de leur emplacement sur les serveurs Internet. Avec l'apparition du Web 2.0 dans les années 2000, un grand nombre d'individus appelés communément *des internautes* sont devenus très actifs et participent de plus en plus à la création de contenus et d'applications partagées sur le Web [1].

Aujourd'hui, l'utilisation des applications Web via Internet est devenue un moyen indispensable pour différentes organisations commerciales et gouvernementales pour réduire les coûts de leurs gestions, accélérer leurs activités, améliorer la qualité de leurs services et atteindre le plus grand nombre de publiques ciblées. Les internautes bénéficient des avantages immenses des services fournis en ligne. Cependant, ces gains ne sont pas sans risques ; les applications Web sont devenues des cibles pour différentes cyberattaques, mettant en danger leurs propres données confidentielles et celles des utilisateurs enregistrés sur les serveurs qu'ils hébergent [2].

Les problèmes que les cyberattaques peuvent causer sont énormes, nous citons à titre d'exemple, le vol d'informations sensibles et le déni des services Web. Les attaquants peuvent profiter des failles, appelées aussi *vulnérabilités*, découvertes sur les applications Web, leurs plugins associés ou les serveurs d'hébergement, pour injecter des scripts malveillants dans le but de les compromettre. De cette façon, les pages explorées des applications Web compromises sur les navigateurs des utilisateurs, permettent aux attaquants d'augmenter leurs privilèges d'accès et de révéler des informations sensibles conservées par les navigateurs, telles que les noms et les mots de passe des utilisateurs victimes [2].

2. Le cross-site Scripting (XSS)

Le cross-site scripting ou XSS est reconnu comme un type de faille de sécurité qui se trouve typiquement dans les applications Web qui utilisent des formulaires de saisie pour accéder aux espaces personnels, chercher, créer, modifier ou pré-visualiser des

données enregistrées. Quand ces applications n'utilisent pas assez de validation sur les données saisies, les attaquants profitent de cette faille pour injecter des codes malicieux dans le formulaire de saisi et ensuite dans le contenu des pages de l'application. Les attaquants utilisent des codes interprétables directement par le navigateur Web, généralement écrits en langage de script comme JavaScript, ActiveX ou VBScript. Cela permet l'exécution automatique des codes injectés par le navigateur lors de la visite des pages infectées. L'injection se fait principalement de deux manières distinctes [5] :

1. Injection dans les paramètres des URLs avec suscitation des victimes à accéder à ces URLs depuis des liens soumis par emails.
2. Injection directe dans les serveurs sous forme de messages ou fichiers infectés, tous les visiteurs des pages infectées deviennent alors des victimes.

Les navigateurs Web des victimes ne peuvent pas détecter la malveillance des scripts injectés ; ces derniers contournent les contrôles d'accès, usurpent l'identité des utilisateurs et le contenu HTML peut en être réécrit.

Les premières attaques XSS ont été découvertes assez tôt vers la fin de l'année 1999 par les ingénieurs de sécurité de Microsoft. Depuis, les attaques XSS ne cessent de cibler de nouvelles applications et plateformes Web bénéficiant du développement rapide de la technologie Web et la nécessité de la disponibilité rapide de ces applications [5].

3. Conséquences des attaques XSS

Les effets XSS varient d'une nuisance mineure comme l'affichage des pop-up indésirables, à des risques de sécurité importants ; tout dépend de la nature de l'application Web ciblée, ainsi que des informations traitées par cette application. Dans ce qui suit, nous en discutons des conséquences les plus populaires.

3.1. Vol des cookies de session

Les cookies sont des informations manipulées par le navigateur et stockés dans des fichiers textes. Le cookie est un moyen de communication entre un serveur Web et un

client. Il sert à sauvegarder des informations sur l'utilisateur concernant ses visites fréquentes à une application Web ; ces informations sont échangées avec le serveur hébergeant l'application Web afin de faciliter l'identification des clients et de mettre à jour automatiquement leurs préférences de navigation pour cette application. Les cookies servent ainsi à l'enregistrement des informations d'identification des utilisateurs sur un service Web, tel que l'identificateur et le mot de passe ; cela permet aux utilisateurs de rester connectés sur le site sans avoir besoins de re-saisir les informations d'identification d'accès [6]. Ces informations sensibles peuvent être volées par une attaque XSS appropriée ; ce qui permet à un attaquant de se faire passer pour une victime et d'accéder à des informations et fonctionnalités sensibles en son nom [5].

3.2. Modification de l'apparence et la convivialité des sites Web

Les attaques XSS peuvent être utilisées pour altérer le contenu original d'une page Web ; des fausses informations, des images et des mots offensives peuvent être facilement intégrées sur des page Web légitimes, ce qui peut ternir la réputation des sites de confiance [5].

3.3. Déni de service

Les attaques XSS permettent aussi l'injection d'un code malveillant, capable de générer un nombre important de requêtes vers un serveur afin de réaliser un déni de service [2].

4. Prévalence des vulnérabilités aux attaques XSS

Les attaques XSS sont principalement dues aux failles des applications Web. La vulnérabilité XSS apparaît dans la liste des 10 principales vulnérabilités signalées par l'Open Web Application Security Project (OWASP) [W1] depuis 2003 jusqu'au dernier rapport publié en 2021 [2]. La figure 1.1 montre la disposition des vulnérabilités XSS vis-à-vis des autres vulnérabilités dans le Web selon les classements annuels établis par OWASP.

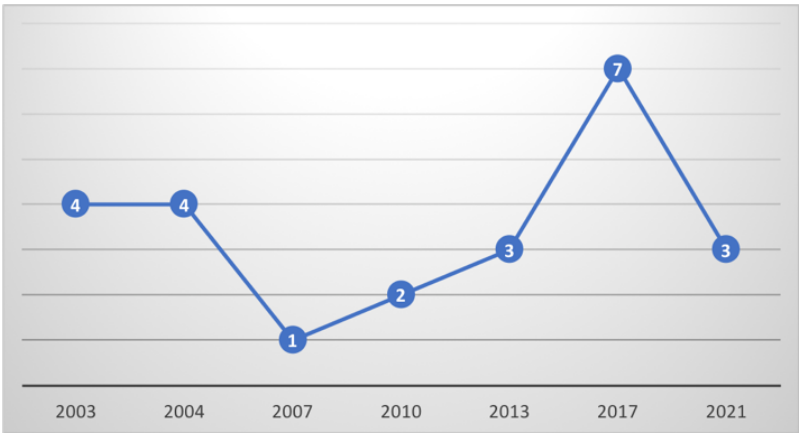


Figure 1.1. Rapport sur les vulnérabilités XSS [2]

La non-stabilité du classement des attaques XSS dans la figure 1.1 est probablement dû à l'évolution des types des attaques et les mesures de sécurités adoptées pour les contrer.

5. Les types des attaques XSS

L'Open Web Application Security Project (OWASP) [W1] a signalé l'émergence de trois types d'attaques XSS, nous discutons chacun de ces types dans ce qui suit :

5.1. Les attaques XSS reflétées (Reflected XSS)

Les attaques XSS reflétées (ou *reflecled XSS* en anglais) se produisent lorsque des données malveillantes fournies dans le cadre de requêtes HTTP ou URLs font partie des réponses du serveur sans être correctement validées. Par conséquent, les données malveillantes intégrées dans les réponses HTTP apparaissent du côté client. Le navigateur de la victime exécute alors le code malveillant généré sur la page résultat [W1]. La figure 1.2 montre le scénario typique d'une attaque XSS reflétée.

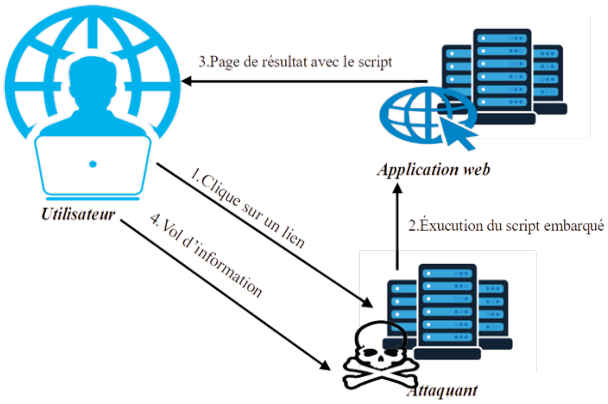


Figure 1.2. Scénario d'une attaque XSS reflétée

Les cibles typiques de ce type d'attaque sont les applications Web avec des capacités de recherche, où les données malveillantes intégrées font partie des résultats de recherche ou des messages d'erreur [2, 5]. Ces attaques sont également appelées *non-persistantes* ou de *Type I* [W1]. Elles sont dites non-persistantes car les données malveillantes ne sont pas stockées en permanence sur le serveur. Par conséquent, tous les champs des formulaires présentent des vulnérabilités de sécurité potentielles qu'un attaquant peut exploiter via XSS. L'attaquant crée un lien masqué vers une application Web qui contient du code malveillant dans ses paramètres. En utilisant ce lien, la victime force le navigateur à exécuter ce code. Le Web 2.0 et ses systèmes de gestion de contenu ont popularisé cette attaque en permettant de publier des liens aisément et de manière visible sur tout le Web [4].

Exemple : On suppose que l'attaquant vise à infecter les utilisateurs d'une application Web légitime (`http://sitelegitime.com/index.php`). Si cette application est vulnérable aux attaques XSS, c.-à-d., accepte des données utilisateurs sans validation dans un de ses champs de saisies (e.g., `user=`), l'attaquant utilise l'URL de cette application pour former un lien avec un code malveillant, appelé aussi *vecteur d'attaque* comme suit :

```
http://sitelegitime.com/index.php?user=<script>alert(document.cookie)</script>
```

Le lien, peut être partagé dans le but d'être cliqué par un de ses récepteurs. Lorsque la victime cliquera sur ce lien, le code malveillant s'exécutera sur son navigateur.

Dans cet exemple, une boîte de dialogue s'affichera contenant les cookies de la victime. Dans le cas d'une véritable attaque, une fois que la victime aura cliqué sur le lien, ses cookies seront automatiquement transférés vers le serveur de l'attaquant sans que la victime ne s'en rende compte [W3].

5.2. Les attaques XSS stockées (Stored XSS)

Les attaques XSS stockées (ou *stored XSS* en anglais) apparaissent lorsque des données malveillantes sont fournies dans des formulaires d'entrée pour être stockées dans des bases de données ou des fichiers d'application serveur sans être nettoyées. De cette façon, les utilisateurs qui demandent des données à ces applications reçoivent des réponses du serveur contenant des données malveillantes qui deviennent nuisibles lorsqu'elles sont rendues par les navigateurs [2, 5]. Ce type d'attaques est plus dangereux que le premier (i.e., attaque XSS reflétée) car le code fait partie intégrante des données de l'application Web et peut atteindre plusieurs victimes [4]. Les cibles typiques de ce type d'attaques sont les réseaux sociaux et les forums où des données malveillantes peuvent être publiées et stockées dans des bases de données et donc infecter chaque utilisateur qui y accède. Les attaques XSS stockées sont également appelées *persistantes* ou de *Type II* [W1]. Elles sont persistantes car les données malveillantes stockées sur le serveur restent nuisibles et efficaces jusqu'à ce qu'elles soient détectées, supprimées ou filtrées [2]. La figure 1.3 montre le schéma d'une attaque XSS stockée.

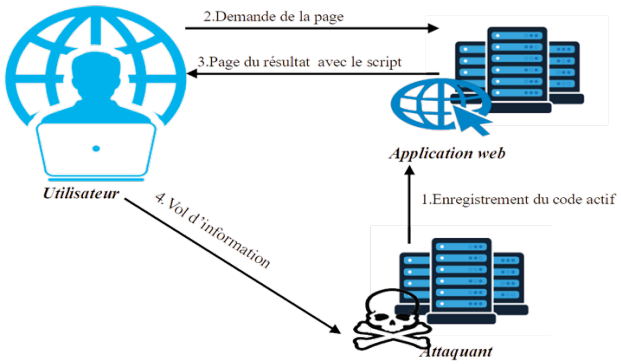


Figure 1.3. Scénario d'une attaque XSS stockée

Exemple : Prenons le cas d'un forum de discussion disposant d'un espace d'édition pour poster des commentaires sur des articles quelconques. Si l'espace réservé aux commentaires est vulnérable aux attaques XSS, l'attaquant va pouvoir saisir un code malveillant (e.g., `<script>alert(document.cookie)</script>`) au lieu d'un code claire pour un article X. Ce code sera sauvegardé sur le serveur et s'exécutera pour chacun des utilisateurs accédant aux commentaires sur l'article X. Dans cet exemple, une boîte de dialogue s'affichera dans le navigateur des victimes, contenant leurs cookies. Dans le cas d'une véritable attaque, ses cookies seront automatiquement transférés vers le serveur de l'attaquant sans que la victime ne s'en aperçoive [W3].

5.3. Les attaques XSS basées sur le DOM (DOM-based XSS)

Les attaques basées sur le DOM (ou *DOM-based XSS* en anglais) sont des attaques qui surviennent au moment de l'interprétation des pages par le navigateur. Contrairement aux autres types d'attaques XSS, les données malveillantes utilisées dans les attaques basées sur le DOM sont utilisées pour modifier dynamiquement les arbres DOM (Document Object Model) générés par les navigateurs lors de la phase d'interprétation des résultats des requêtes [7]. Les attaquants utilisent des attaques DOM pour les applications Web qui acceptent des données provenant de sources contrôlées par l'utilisateur pour être utilisées comme valeurs d'objets DOM spécifiques. La particularité de ce type d'attaques est que des données malveillantes peuvent être intégrées dans des URLs en tant que valeurs d'objets DOM ou d'éléments HTML spécifiques mais n'atteignent jamais le serveur [2,5]. Ces attaques sont également appelées *Type 0* et ont été signalées pour la première fois en 2005 par Amit Klein [W2]. L'attaquant crée un lien avec le code malveillant et l'envoie à la victime. Lorsque la victime clique sur le lien, elle reçoit une réponse non malveillante. Le code malveillant est exécuté côté client et un attaquant pourrait obtenir des informations sensibles de la victime. Le processus détaillé est illustré dans la figure 1.4.

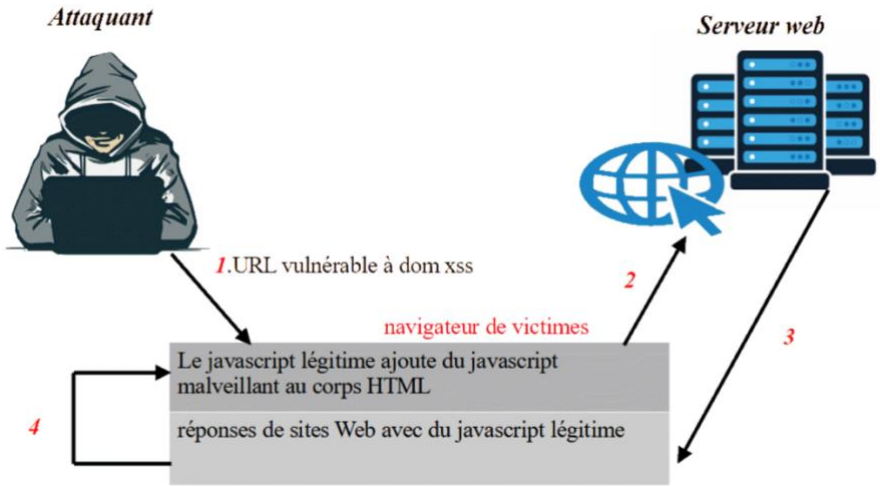


Figure 1.4. Scénario d'une attaque XSS basée sur le DOM

Exemple : Prenons l'exemple d'une page Web en HTML contenant le code suivant :

```
<script>
      document.write('Lien disponible sur :' + document.location.href + '.');
</script>
```

L'attaquant peut ajouter le code malveillant sous la forme suivante :

```
http://sitelegitime.com/index.php#<script>alert(document.cookie)</script>
```

Par un simple clic sur ce lien, le code malveillant devient une partie de la structure DOM de la page replaçant la valeur de l'attribut (`document.location.href`). En conséquence, une boîte de dialogue s'affichera aux victimes contenant leurs cookies. Dans le cas d'une véritable attaque, ces cookies seront automatiquement transférés sur le serveur de l'attaquant à l'insu de la victime [W2, W3].

6. Les principales causes des attaques XSS

Les attaques XSS sont principalement causées par l'application Web elle-même. Les développeurs inexpérimentés face aux problèmes de sécurité, omettent souvent l'intégration des désinfections appropriées à toutes les données saisies par l'utilisateur. Cela est dû à différentes causes :

6.1. Manque d'expérience des développeurs Web

La majorité des développeurs Web méconnaissent les problèmes et les pratiques de sécurité. Ils utilisent pour la plupart, des modèles prêts à l'emploi (ou frameworks), qui peuvent être facilement exploités, personnalisés et adaptés pour la création automatique des applications. Ces frameworks se concentrent généralement sur les aspect fonctionnelles et visuelles des applications tout en ignorant les aspects non-fonctionnelles les plus avancés comme la sécurité. Ces dernières nécessitent des experts pour les intégrer efficacement dans les applications Web [2].

6.2. Exigences socio-économiques

Les entreprises sont exigeantes en matière de changements rapides des services Web pour répondre aux besoins des nouveaux clients. Ces exigences forcent les organisations à se focaliser sur la disponibilité de ces applications et à améliorer leurs services offerts, tout en négligeant l'aspect de sécurité qui nécessite plus du temps pour être étudié, analysé et fixé [2, 5].

6.3. La nature de l'attaque

L'attaque XSS est de nature simple. Pour lancer une telle attaque, il suffit d'injecter un script dans n'importe quel endroit vulnérable aux attaques ; ces scripts sont énormes, et peuvent être écrits par différents langages. Pour rendre la détection de ces scripts difficile, ils peuvent être dissimulés en utilisant une panoplie de techniques, comme l'encodage et le cryptage. Pour sécuriser parfaitement une applications Web contre les attaques XSS, les entrées des utilisateurs ne doivent jamais être dignes de confiance et doivent être considérés comme des vecteurs d'attaque potentiels [2].

7. Mesures de sécurités contre les attaques XSS

Pour réduire la probabilité d'être affecté par les attaques XSS, différentes mesures et pratiques de sécurité doivent être prises, d'abord par les développeurs et les propriétaires ainsi que par les utilisateurs des applications sur le Web. Théoriquement, ces mesures peuvent interdire l'apparition des attaques en premier lieu et permettre aux développeurs d'une part, de concevoir des applications Web

sans risques XSS et d'autre part, protéger les clients contre les attaques XSS. Pour les développeurs et propriétaires, quelques mesures doivent être prises très tôt dès les premières étapes du processus de développement des applications Web, en particulier lors de la phase de construction, avant et après le déploiement.

7.1. Mesures de sécurité côté développeurs ou propriétaires

Afin de réduire les risques d'attaques, plusieurs mesures peuvent être prises par les développeurs et les propriétaires d'applications Web :

1. *Utiliser des APIs sécurisés* : Les navigateurs Web sont souvent livrés avec des API intégrées qui permettent d'effectuer des opérations complexes, telles que la manipulation de haut niveau de la structure DOM des documents Web. Ces APIs ne sont pas sans défauts ; Elles peuvent être utilisés par des intrus pour lancer des attaques XSS. Pour remédier à ce problème, des APIs plus sécurisées peuvent être installées et utilisées par les navigateurs. Les APIs sécurisées imposent le typage des valeurs des éléments des balises HTML pour empêcher toute injection de script dans des endroits indésirables [2].

2. *Utiliser des Framework sécurisés* : Des modèles plus sécurisés doivent être utilisés pour la génération automatique des applications Web. Ces modèles doivent incorporer des routines de filtrages de toute donnée saisie par l'utilisateur. Ceci libère les développeurs des soucis de sécurité et les aide à se focaliser sur l'aspect fonctionnel de leurs applications [2].

3. *Utiliser des routines de désinfection appropriées* : Des bibliothèques de filtrage de données doivent être développées et mises à la disposition des développeurs, avec un guide compréhensif indiquant quand, où et comment ces routines doivent être intégrées dans l'application [2].

4. *Contrôler tous les points d'injections des applications* : Assurer que toutes les pages des applications n'acceptent les entrées utilisateur qu'après décodage et filtrage [W2].

5. *Utiliser des outils de détection automatiques des vulnérabilités* : Utiliser des outils d'aide pour la détection des vulnérabilités XSS et appliquer les correctifs adéquats si nécessaire [W2].

6. *Mettre à Jour des applications* : Mettre à jour les applications régulièrement afin d'éviter tout nouveau type d'exploitation dans le futur [W2].

7.2. Mesures de sécurité côté clients ou utilisateurs

Les utilisateurs sont aussi concernés par les mesures de sécurité, pour ne pas être des cibles faciles pour les attaquants ; nous citons ici quelques mesures qui doivent être prises du côté des clients :

1. Désactiver tous les scripts non nécessaires [W4].

2. Éviter de cliquer sur des liens provenant d'un tiers, surtout ceux qui surviennent par email (cas de d'hameçonnage) [4].

3. Taper directement l'URL des sites désirés dans la barre d'adresse du navigateur, au lieu de passer par la source ou le lien d'un tiers [W4].

4. Mettre à jour régulièrement les navigateurs pour renforcer leurs sécurités [W4].

5. N'installer que les plugins nécessaires et certifiés [W4].

6. Réduire la sauvegarde et l'activation des cookies sur les navigateurs [W4].

En pratique, ces mesures sont difficiles à appliquer et nécessitent des outils automatiques qui aident à leurs adoptions. En plus, les attaques XSS avancées peuvent se propager malgré l'application de ces mesures. Par exemple, quelques attaques XSS avancées n'injectent pas des scripts malveillants, mais plutôt, les

attaquants utilisent des scripts déjà utilisés dans les applications Web pour lancer des attaques XSS, ce qui rend les méthodes de détection classiques de ces attaques inefficaces [2].

8. Conclusion

Dans ce chapitre, nous avons présenté quelques concepts de base concernant l'une des plus dangereuses attaques sur le Web. Nous avons présenté l'attaque XSS, son principe et ces différents types. Nous avons aussi montré la prévalence des vulnérabilités XSS au cours des dernières années. Dans le chapitre suivant, nous présenterons les derniers travaux scientifiques et les techniques utilisées pour la détection des **attaques** XSS.

CHAPITRE II.

TRAVAUX CONNEXES SUR LA DETECTION DES ATTAQUES XSS

CHAPITRE II.

TRAVAUX CONNEXES SUR LA DÉTECTION DES ATTAQUES XSS

La détection en temps réel des attaques XSS constitue la dernière ligne de défense dans le processus de développement des applications sûres. Ces techniques sont nécessaires pour couvrir les risques que l'analyse et la réparation des vulnérabilités ne permettent pas de traiter. Elles peuvent également être considérées comme des solutions de protection alternatives lorsque la réparation des vulnérabilités devient difficile à réaliser dans certaines circonstances, telles que le cas des applications Web anciennes, volumineuses ou critiques. Les techniques de détection peuvent être installées indépendamment des applications Web, dont le but est de signaler la présence des attaques lorsqu'elles se produisent. Deux techniques de bases sont utilisées dans la littérature pour la détection des attaques XSS en temps réel : *surveillance du trafic HTTP* et *apprentissage automatique* [2]. Ce chapitre est consacré à la présentation et discussion de quelques approches de chacune de ces deux catégories en se focalisant sur ceux basées sur l'apprentissage automatique.

1. Techniques basées sur la surveillance du trafic HTTP

Grâce à la surveillance du trafic HTTP, il est possible de capturer, en temps réel, toute requête ou réponse du serveur liée à une application Web cible. Cela permet de confirmer ou infirmer l'injection des scripts malicieux durant l'échange de données entre clients et serveurs. Cette surveillance permet aux administrateurs et aux clients de prendre les mesures appropriées pour se défendre contre les attaques XSS.

Johns et al. [8] ont utilisé la surveillance du trafic HTTP pour la détection de deux types d'attaques XSS: reflétées et stockées (voir Chapitre I). Les attaques XSS reflétées correspondent aux scripts extraits des requêtes et des réponses HTML après avoir supprimé tout encodage et effectué une correspondance de similarité basée sur un seuil. Pour les attaques XSS stockées, un modèle est entrainé sur des scripts générés par l'application Web cible par soumission de données confiantes ; toute différence détectée en temps réel est signalée comme une tentative d'attaque.

Sundareswaran et Squicciarini [9] comparent les graphes de flux de contrôle des pages Web sûres avec ceux générées en temps réel sur un serveur proxy. Les pages Web des requêtes bénins sont envoyées aux serveurs et les pages résultantes sont envoyées au proxy pour une phase d'analyse. Le proxy génère leurs graphes de flux de contrôle et les enregistrent localement. Un plugin côté client intercepte les requêtes HTTP et les envoie au proxy qui génère également les graphes de flux de contrôle des pages résultant de la réponse du serveur ; il extrait et compare les caractéristiques des graphes stockés et générés des pages visitées dans le but de chercher des anomalies. Un seuil est utilisé pour vérifier la présence d'une attaque. Le plugin client est alors appelé pour bloquer ou autoriser le rendu de la page.

Sun et al. [10] interceptent et comparent les scripts incorporés dans les requêtes http et leurs réponses. Tout d'abord, les requêtes HTTP sortantes sont analysées et les scripts utilisés comme valeurs de leurs paramètres sont extraits, décodés et stockés. La structure d'arborescence de la réponse du serveur est générée ; les scripts sont extraits depuis les : (1) gestionnaires d'événements, (2) attributs HTML, (3) attributs ou balises spécifiques au navigateur et (4) URIs ; ils sont ensuite décodés. Enfin, une similarité est mesurée entre les scripts décodés résultant des requêtes et les réponses HTTP sortantes. Une correspondance est signalée comme une attaque XSS.

2. Techniques basées sur l'apprentissage automatique

Différents modèles basés sur l'apprentissage automatique ont été construits pour la détection des attaques XSS. Ces modèles sont développés dans le but d'apprendre les propriétés cachées des vecteurs d'attaque et de faire des prédictions correctes à l'exécution [2]. Différents types d'apprentissage automatique ont été explorés dans la

littérature : *simple*, *ensembliste* et *profond*. Dans la suite de cette section, nous discutons quelques propositions remarquables.

2.1. Approches à base d'apprentissage automatique simple

Nunan et al. [11] ont identifié un ensemble de caractéristiques qui permettent la classification automatique précise des vecteurs XSS dans les pages Web. Ils ont expérimenté deux méthodes d'apprentissage simples, à savoir une classification naïve bayésiennes (NB) et une classification par machine à vecteurs de support (SVM), pour classer des pages Web à l'aide d'un ensemble de données composé de 216 054 sites Web. Trois types de caractéristiques sont extraites des URLs et du contenu des pages Web qui leurs correspond : caractéristiques d'obscurcissement (e.g., longueur d'URL et présence d'encodage), motifs suspects (e.g., nombre des domaines dans les URLs et présence de caractères doubles) et caractéristiques des scripts (e.g., présence de certains tags et gestionnaires d'événements). Après le décodage du contenu des pages Web, les caractéristiques extraites sont transmises aux classificateurs pour validation. Les résultats montrent que les deux classificateurs fonctionnent bien, mais que SVM fonctionne mieux et atteint 99,89% en termes du succès (*Accuracy*).

Goswami et al. [12] ont proposé une approche non-supervisée pour la détection des attaques XSS reflétées. Lorsque le client envoie une requête au serveur, elle est filtrée côté client et soumise à un test d'alpha-divergence en référence au profil attaque/normal. Si la valeur du test dépasse une valeur de seuil prédéfinie, la demande n'est pas traitée davantage ; elle est bloquée directement du côté client. Sinon, la demande est transmise au proxy pour un traitement ultérieur. Le serveur proxy effectue l'extraction de 16 caractéristiques incluant le nombre de caractères, de lignes et de commentaires dans le script ; en plus, il effectue la sélection des caractéristiques à l'aide d'un processus d'agrégation à partir de l'exécution de plusieurs algorithmes de sélection. À la fin, les profils générés sont soumis à une classification par le biais d'un algorithme de regroupement (KMeans). L'approche proposée atteint 98.89% en termes du succès.

Mokbal et al. [13] ont proposé l'utilisation d'un perceptron multicouche (MLP) pour la détection des attaques XSS. Le processus de détection permet l'extraction dynamiques de 41 caractéristiques depuis les réponses HTTP des requêtes des

utilisateurs. Ces caractéristiques incluent la présence de certains tags HTML, la longueur maximale et minimale des scripts incorporés. Un réseau MLP est entrainé sur des vecteurs construits à partir des valeurs des caractéristiques obtenues. Le modèle a montré une efficacité sur la détection des attaques XSS et atteint 99,32% en termes du succès sur l'ensemble des données collectés.

2.2. Approches à base d'apprentissage automatique ensembliste

Mereani et Howe [14] ont proposé un système de classification en deux étapes. À la première étape, un classificateur d'arbre de décision (DT) est utilisé pour prédire la nature des entrées de l'utilisateur (c'est-à-dire des textes clairs ou des scripts). Les scripts sont soumis à une deuxième étape de classification avec un classificateur SVM (machine à vecteurs de support) pour prédire leur malveillance. Au total, 62 caractéristiques alphanumériques et non alphanumériques sont extraites et utilisées pour la classification. Le système atteint 99.93% en termes du succès.

Zhang et al. [15] ont utilisé deux modèles de mélange gaussien (GMM) distincts entraînés respectivement sur des scripts bénins et malins. Les probabilités produites des deux GMMs, sur les scripts testés, sont comparées pour parvenir à une prédiction finale. Ils ont constaté que la combinaison des caractéristiques extraites des requêtes et des réponses HTTP augmente le score total du succès du modèle à 96.49%.

Li et al. [16] ont utilisé un modèle ensembliste de type forêt d'arbres décisionnels (RF) pour une classification semi-supervisée. Afin de rectifier les échantillons mal étiquetés, un algorithme des voisins les plus proches pondéré (KNN pondéré) est adopté pour ré-étiqueter les échantillons en fonction de leurs distances pondérées par rapport à d'autres échantillons similaires. Le modèle atteint 97.30% du succès.

Zhou et Wang [17] ont utilisé un modèle ensembliste formé par un vote de plusieurs réseaux bayésiens. Dans le but d'améliorer la prédiction du modèle proposé, les prédictions obtenues à partir du réseau sont combinées avec un ensemble de règles de renseignement sur les menaces extraites (i.e., cyber threat intelligence), y compris

les adresses IP et les domaines malveillants depuis PhishTank[1] et FireHOL[2]. Le modèle atteint 98.54% du succès.

Outre les caractéristiques traditionnelles extraites des contenus URL et HTML, Wang et al. [18] et Rathore et al. [19] ont expérimenté un nouveau type de caractéristiques nommé OSN pour la détection des vers XSS (XSS *worms* en anglais). Ces vers se distinguent par leurs capacités à s'auto-propager. Ils peuvent donc infecter un maximum d'utilisateurs sur les réseaux sociaux sans intervention de l'utilisateur. Les caractéristiques OSN capturent le comportement observé sur les services des réseaux sociaux ciblés. Ils incluent la vitesse de propagation et la fréquence des données suspectes dans le trafic réseau. Les auteurs ont expérimenté plusieurs classificateurs avec et sans les caractéristiques OSN et ont constaté que les nouvelles caractéristiques sont plus discriminantes par rapport aux autres caractéristiques classiques. Wang et al. [18] ont opté pour un modèle ensembliste de type AdaBoost qui atteint 93.90% de précision alors que Rathore et al. [19] ont opté pour un modèle à base de forêt d'arbres décisionnels (RF) qui atteint 97.40% du succès.

2.3. Approches à base d'apprentissage automatique profond

Kadhim et Gaata [20] ont proposé l'utilisation d'un modèle d'apprentissage profond constitué d'un réseau de neurones convolutifs (CNN) avec une couche de mémoire longue à court terme (LSTM). Les vecteurs d'attaques XSS sont d'abord nettoyés par (1) le décodage pour restaurer le format lisible, (2) généralisation : remplacer les domaines par un domaine unifié synthétique, remplacer les nombres par 0, éliminer les caractères de contrôle vides, remplacer les paramètres de fonction par une chaîne spécifique, (3) tokenisation où l'algorithme de vectorisation Wod2Vec est appliqué. Le modèle proposé atteint 99.30% en termes de mesure F1.

Chaudhary et al. [21] ont axé leur solution sur la conception d'une approche qui détecte les attaques XSS dans les réseaux des objets (IoT) pour protéger la confidentialité des données. Ils proposent un réseau de neurones convolutifs (CNN) pour détecter les vecteurs d'attaque XSS après avoir appliqué certaines méthodes de préparation des données. Cette approche aide à prévenir les atteintes à la vie privée,

[1] PhishTank : https://phishtank.org
[2] FireHOL : https://firehol.org

ce qui aide finalement les entreprises à renforcer le lien avec leurs utilisateurs. Les résultats expérimentaux ont révélé que l'approche atteint une précision de détection de 99.37% après l'exécution réussite des méthodes de préparation des données.

Liu et al. [22] ont proposé un modèle de détection par le biais d'un réseau à convolution graphique (GCN). Ce dernier pourrait distinguer les vecteurs XSS injectés dans le contenu soumis par l'utilisateur. Le modèle proposé atteint 99.70% en termes de mesure F1.

Le tableau suivant résume les détails des approches discutées auparavant ; il indique, pour chaque approche, le type d'apprentissage automatique utilisé, le modèle adopté, la source et le nombre de caractéristiques (*Source* et # respectivement), les types des attaques XSS ciblées et le meilleur score obtenu.

Approche	Type	Modèle	Source	#	Attaques ciblées	Score
Nunan et al. [11]	Simple, supervisé	SVM	URL et Contenu des pages Web	6	reflétées, stockées et DOM	Accuracy = 99.89%
Goswami et al. [12]	Simple, non-supervisé	KMeans	Scripts	16	reflétées	Accuracy = 98.89%
Mokbal et al. [13]	Simple, supervisé	MLP	URL et Contenu des pages Web	41	non-spécifié	Accuracy = 99.32%
Mereani et Howe [14]	Ensembliste, supervisé	DT, Stack(SVM-L,SVM-P,RF,NN)	Scripts inclus dans les requêtes HTTP	62	stockées	Accuracy = 99.97%
Zhang et al. [15]	Ensembliste, supervisé	Stack(2GMM)	Requêtes/Réponses HTTP et réponses vectorisées	200	reflétées, stockées	Accuracy = 96.49%
Li et al. [16]	Ensembliste,	RF + KNN pondérée	URL	12	non-spécifié	Accuracy

	semi-supervisé					= 97.30%
Zhou et Wang [17]	Ensembliste, supervisé	Vote(xBN) et threat intelligence	Scripts inclus dans les requêtes HTTP	30	non-spécifié	Accuracy = 98.54%
Wang et al. [18]	Ensembliste, supervisé	AdaBoost	URL, Contenu des pages Web et OSN	12	vers XSS	F1-score = 93.90%
Rathore et al. [19]	Ensembliste, supervisé	RF	URL, Contenu des pages Web et OSN	25	vers XSS	F1-score = 97.40%
Kadhim et Gaata [20]	Profond, supervisé	CNN avec une couche LSTM	Scripts vectorisés	300	non-spécifié	F1-score = 99.30%
Chaudhary et al. [21]	Profond, supervisé	CNN	Scripts vectorisés	200	reflétées, stockées	F1-score = 99.37%
Liu et al. [22]	Profond, supervisé	GCN	Scripts vectorisés	200	reflétées, stockées	F1-score = 99.70%

Tableau 2.1. Détails des approches basées sur l'apprentissage automatique

3. Conclusion

Dans ce chapitre, nous avons présenté quelques approches qui ont été déjà explorées dans la littérature pour la détection des attaques XSS. Le problème majeur de ces travaux réside dans l'absence des ensembles de données de référence, pour la comparaison effective et équitable des différents modèles proposés. Dans ce projet, nous visons la collection d'un ensemble de données de taille raisonnable et l'expérimentation de différentes caractéristiques et différents classificateurs pour la détection des attaques XSS. C'est ce qui fera l'objet du prochain chapitre.

CHAPITRE III.

DETECTION DES ATTAQUES XSS PAR APPRENTISSAGE AUTOMATIQUE

CHAPITRE III.

DETECTION DES ATTAQUES XSS PAR APPRENTISSAGE AUTOMATIQUE

L'apprentissage automatique est une discipline de l'intelligence artificielle (IA) qui se concentre sur le développement et l'utilisation des algorithmes qui apprennent à partir des expériences passées et des données enregistrées. Ces algorithmes ont la capacité d'améliorer leurs précisions au fil du temps sans être programmés pour le faire. Les algorithmes d'apprentissage automatique sont *entraînés* sur des données disponibles afin de faire des prédictions sur de nouvelles données. Plus l'algorithme est bien entrainé (i.e. ; quantité et qualité des données utilisées en phase d'apprentissage), plus les prédictions deviendront précises. L'objectif principal est de permettre aux machines d'apprendre automatiquement sans l'intervention humaine et d'ajuster leurs actions en conséquence [23]. L'apprentissage automatique est devenu une technique populaire et fréquemment utilisée en cyber sécurité [2].

L'élaboration d'un modèle d'apprentissage automatique pour résoudre un problème donné est un processus à plusieurs étapes. Le processus implique deux phases fondamentales : la *préparation des données* et la *conduite et l'évaluation du modèle d'apprentissage*. Chaque phase comporte un ensemble d'étapes élémentaires qui peuvent être exécutées de manière séquentielle ou en parallèle comme le montre la figure 3.1 [23]. Ces étapes sont expliquées en détails dans ce chapitre en se focalisant sur l'application de la technique d'apprentissage automatique pour la détection des attaques XSS. Nous détaillons également les différentes étapes du processus d'élaboration des modèles et nous discutons les résultats obtenus.

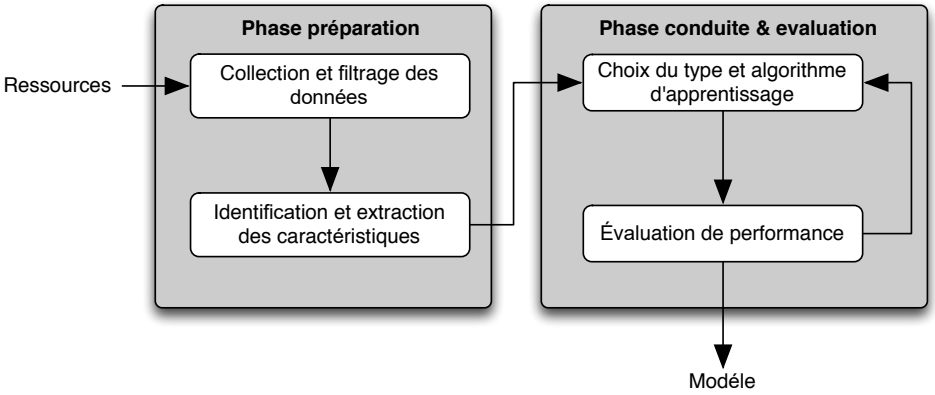


Figure 3.1. Processus général de l'apprentissage automatique

1. Collection des données

La collecte des données est le fondement du processus d'apprentissage automatique. Les données sont nécessaires pour l'entrainement des classificateurs. Dans notre projet, les données collectées ne sont qu'un ensemble de scripts (écrits en JavaScript), communément utilisés pour lancer des attaques XSS. Lorsque l'apprentissage automatique est conçu pour résoudre un problème de classification, au moins deux classes de données doivent être collectées pour permettre aux classificateurs de distinguer les différentes données en test. Dans le contexte de notre projet, deux types de scripts doivent être collectés : *malin* et *bénin*. L'ensemble de scripts malins représente des attaques XSS réelles détectés par les experts et partagés sur Internet, alors que les scripts bénins sont des scripts couramment utilisés dans les applications Web pour réaliser des actions dynamiques authentiques. Récemment, Hannousse et al. [2] ont conduit une revue systématique sur les techniques de détection des attaques et vulnérabilités XSS. Dans cette étude, l'ensemble des données utilisées par des recherches antérieures trouvées dans la littérature, a été collecté. Nous tirons profit de ces données pour collecter l'ensemble des données nécessaires pour l'élaboration d'un modèle d'apprentissage automatique. Les tableaux 3.1 et 3.2 montrent la liste des liens vers les données malignes et bénignes respectivement utilisées par les études antérieures, identifiées par Hannousse et al. [2].

Liste des liens vers des vecteurs d'attaques XSS
https://github.com/duoergun0729/1book/tree/master/data

https://github.com/payloadbox/xss-payload-list
http://www.xssed.com/
https://aw-snap.info/articles/js- examples.php
https://www. linkedin.com/pulse/20140812222156- 79939846-xss-vectors-you-may-need-as-a-pen-tester
http://www.xss-payloads.com/payloads.html
https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/XSS%20Injection
https://gbhackers.com/top-500-important-xss-cheat-sheet/
https://packetstormsecurity.com/files/112152/Cross-Site-Scripting- Payloads.html
http://www.xssed.com/archive/special=1
https://figshare.com/articles/dataset/XSS_dataset1_csv/13046138/4
https://ieee-dataport.org/open-access/detecting-xss-attacks-combining-cnn-lstm
https://github.com/IramTariq/XSS-attack-detection
https://github.com/duoergun0729/1book/tree/master/data
https://github.com/das-lab/deep-xss
https://github.com/ajinabraham/OWASP-Xenotix-XSS-Exploit-Framework
https://github.com/bsmali4/XSSfork
http://portswigger.net/burp
https://github.com/fuzzdb-project/fuzzdb
https://codeload.github.com/foospidy/payloads/zip/master
http://ha.ckers.org/xss.html
https://www.owasp.org/index.php/XSS
http://html5sec.org/
http://xss2.technomancie.net/vectors/
http://www.vulnerability-lab. com/resources/documents/531.txt
https://twitter.com/XSSVector
https://www.owasp.orgi/index.php/XSS_Filter_Evasion_Cheat_Sheet
http://airdump.net/xss-pentest-plugin-cross-site-scripting
https://github.com/is mailtasdelen/xss-payload-list
https://www.kaggle.com/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deep-learning
https://github.com/WhiteRabbitc/Wooyun-Email-XSS-Dataset/tree/master/malious-sample
http://www.ict-romulus.eu/web/wapiti
https://www.impactcybertrust.org/dataset_view?idDataset=940
https://pastebin.com/BdGXfm0D
http://www.vulnerability-lab.com/resources/documents/531.txt

http://www.acunetix.com/blog/news/cross- site- scripting- xss- facebook
http://www.acunetix.com/websitesecurity/xss.htm
https://www.owasp.org/index.php/Testing_for_Stored_Cross_site_scripting
https://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet
http://cdn. buyukkayhan.com/public/xss_sqlite.db
http://www.securityfocus.com/archive/ 1/404300.
http://seclists.org/bugtrap/ 2007/Sep/0324.html.
http://www.securityfocus.com/archive/ 1/435127/30/ 120/threaded.
https://chefsecure.com/blog/the-12-exploits-of-xss-mas-infographic
http://www.securityfocus.com/ archive//435127
https://github.com/cure53/DOMPurify/blob/master/test/expect.json
exploit-db.com
https://twitter.com/XSSVector
http://xss2.technomancie.net/vectors/
http://html5sec.org/
https://github.com/danielmiessler/SecLists/ blob/master/Fuzzing/XSS/XSS-RSNAKE.txt
https://github.com/ danielmiessler/SecLists/blob/master/Fuzzing/XSS/XSS- RSNAKE.txt
MyBB 1.0.2 XSS Attack in search.php Redirection. http://www. securityfocus.com/archive/1/423135
phpBB 2.0.18 XSS and Full Path Disclosure. http://archives.neohapsis.com/ archives/fulldisclosure/2005-12/0829
XSS in WebCal (v1.11-v3.04). http://archives.neohapsis.com/archives/ fulldisclosure/2005-12/0810.html%,
https://portswigger.net/web-security/cross-site-scripting/cheat-sheet
https://gist.github.com/kurobeats/9a613c9ab68914312cbb415134795b45
https://code.google.com/p/fuzzdb
https://www.owasp.org/index.php/OWASP Xenotix XSS Exploit Framework
https://nvd.nist.gov/vuln-metrics/cvss
https://github.com/payloadbox/xss-payload-list
https://code.google.com/p/xcampo/

Tableau 3.1. Liste des liens vers des données malignes [2]

Liste des liens vers des données (scripts) bénins
http://dmoztools.net/
http://www.dmoz.org
http://www.lemur project.org

http://sifaka.cs.uiuc.edu/~wang296/Data/index.html
https://www.kaggle.com/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deep-learning
https://www.cs.cmu.edu/~enron/
http://boston.lti.cs.cmu.edu/Data/
https://blogs.msdn.microsoft.com/testing123/2009/02/06/email-address-test-cases/
https://github.com/danielmiessler/SecLists/
https://www.w3schools.com/
https://www.kaggle.com/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deep-learning

Tableau 3.2. Liste des liens vers des données bénignes [2]

L'entrainement des modèles d'apprentissage sur des données similaires ou des données d'un type particulier, peut rendre le modèle complètement inefficace pour d'autres types de données. C'est pourquoi, il est impératif que des considérations nécessaires soient prises lors de la collecte des données, car les erreurs commises à cette étape ne feront que s'amplifier au fur et à mesure pendant la progression vers les dernières étapes [23]. Pour cela, la liste des données est soigneusement examinée en supprimant les liens répétés. En plus, l'une des règles de l'apprentissage automatique est qu'il est important d'équilibrer l'ensemble des données. La principale raison en est de donner une priorité égale à chaque classe. Pour cette raison, nous avons opté pour l'utilisation de la technique de rééchantillonnage, plus précisément la méthode de *sous-échantillonnage*. Cette méthode fonctionne en diminuant le nombre d'observations de la classe majoritaire afin d'arriver à un ratio *classe minoritaire/classe majoritaire* satisfaisant. Le tableau 3.3 suivant montre les résultats de la collection de données après chaque phase de collecte et de filtrage.

Phase de collection/filtrage	#bénins	#malins (XSS)
Collectés	49873	48588
Après déduplication	38531	3242
Après sous-échantillonnage	3242	3242

Tableau 3.3. Collecte et filtrage des données

2. Extraction des caractéristiques

L'extraction des caractéristiques consiste à transformer des données brutes en un ensemble d'information qui représente au mieux le problème sous-jacent des modèles prédictifs, ce qui améliore la précision du modèle sur de nouvelles données. En d'autres termes, l'extraction des caractéristiques permet de transférer les données collectées en des éléments que l'algorithme d'apprentissage peut comprendre. Les connaissances des domaines sont fréquemment utilisées pour en tirer le meilleur parti. Si cette étape est effectuée correctement, la puissance prédictive des algorithmes d'apprentissage automatique en sera forcément améliorée [23]. Pour la détection des attaques XSS, un ensemble de caractéristiques diverses a été utilisé dans la littérature [2]. Pour notre projet, nous avons adopté un ensemble constitué de 56 caractéristiques divisées sur 6 catégories :

2.1 Caractéristiques d'ordre générale

Ces caractéristiques fournissent des informations générales sur les scripts en entrées. Deux caractéristiques de ce genre sont considérées :

1. **Longueur du script (input size) :** des scripts longs sont généralement utilisées par les attaquants afin de réduire la chance d'être facilement interprétés et identifiés par les victimes. La longueur d'un script est un bon indicateur de sûreté ou méchanceté d'un script. Afin de calculer cette valeur pour chaque script en entré (désginé ici par une variable nommé *input*), un code Python simple est utilisé :

```
len(str(input))
```

2. **Nombre de protocole (http/ftp/file) :** les attaquants utilisent des vecteurs d'attaques afin de rediriger l'utilisateur vers des pages ou des fichiers malveillants afin de lancer des attaques. Cela peut se faire uniquement en utilisant l'un des protocoles d'échange de données tel que : http, ftp ou file. Le nombre de présence de ces mots dans le script en entrée peut indiquer une attaque XSS. Le code python suivant est utilisé pour calculer la valeur de cette caractéristique :

```
str(input).lower().count('http') + str(input).lower().count('ftp') +
str(input).lower().count('file')
```

Il est à noter que le script doit d'abord être transformé en minuscules avant de compter le nombre de ces mots, afin d'éviter la perte d'informations liée aux pratiques fréquemment utilisées par les attaquants pour déguiser leurs vecteurs d'attaque. Cette fonction est appliquée à toutes les caractéristiques qui suivent.

2.2 Caractéristiques HTML

Le principe des attaques XSS est d'injecter des scripts malicieux dans les pages Web afin d'être exécutés par les navigateurs des victimes. Ces scripts malicieux sont généralement injectés sous forme de balises HTML ou de valeurs à des éléments des balises HTML. En effet, nous examinons le nombre de balises HTML fréquemment utilisées pour l'injection des attaques XSS utilisés dans les scripts. Voici la liste des balises HTML considérées dans notre étude:

```
script, javascript, confirm, img, src, href, window, document, cookie,
domain, iframe, windows, location, svg, form, action
```

Chacune des caractéristiques *c* est calculée pour un vecteur *input* par l'instruction python suivante. Ceci se fera de même pour le reste des caractéristiques:

```
str(input).lower().count(c)
```

2.3 Mots-clés malveillants

Certains mots-clés sont aussi utilisés par les attaquants pour repérer leurs attaques. La liste de ces mots-clés a été collectée par des experts [2] d'après l'observation des attaques antérieures. Voici la liste des mots-clés considérés dans ce projet :

```
evil, h@ck, anonymous, hack, h4ck, under the control, control by,
controlled by, in control
```

2.4 Fonctions/commandes malveillantes

Pour lancer une attaque XSS, certaines fonctions et commandes JavaScript peuvent être utilisées. Nous considérons les fonctions et commandes les plus répondues :

```
pwd, pown, alert, eval, prompt, fromCharCode, getElementsByTagName,
write, fetch, getElementById, signup, login, search, query
```

2.5 Événements HTML DOM malveillants

Les événements HTML DOM permettent à JavaScript d'enregistrer différents gestionnaires d'événements sur les éléments d'un document HTML. Les événements sont normalement utilisés en combinaison avec des fonctions, les fonctions ne seront pas exécutées avant que l'événement ne se produise (par exemple, lorsqu'un utilisateur clique sur un bouton). Ces événements peuvent aussi être utilisés pour lancer des attaques XSS suivant l'action des victimes. La liste des événements considérés est la suivante :

```
onerror, onload, onmouseover, onfocus, onclick
```

2.6 Caractères malveillants

Un ensemble de caractères est utilisé pour former des attaques XSS ; Certains de ces caractères rentrent dans la construction du script malveillant, d'autres sont généralement utilisés pour déguiser le script. La liste des caractères considérés est la suivante :

```
espace, <, >, ', '', \\, , , +
```

3. Choix du type et de l'algorithme d'apprentissage

Les algorithmes utilisés dans l'apprentissage automatique se divisent en trois types de base comme le montre la figure 3.2 : *apprentissage traditionnel supervisé*, *traditionnel non-supervisé* et *apprentissage profond*.

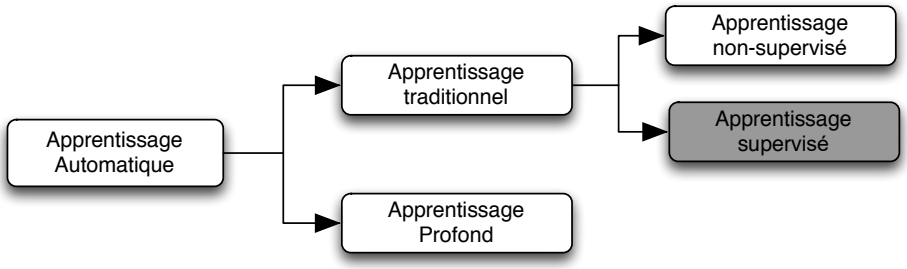


Figure 3.2. Taxonomie des techniques d'apprentissage automatique

Le principe de l'apprentissage automatique traditionnel non-supervisé est de regrouper les individus de l'ensemble de données sans aucune connaissance préalable de leurs étiquettes (leurs classes). L'apprentissage non-supervisé vise alors à découvrir les structures sous-jacentes et intrinsèques dans les données d'entrées pour savoir prédire de nouvelles données [23]. L'apprentissage non-supervisé est relativement simple et rapide à faire.

Au contraire, l'apprentissage supervisé consiste à entrainer les algorithmes exclusivement sur des données étiquetées pour réaliser des prédictions sur de nouveaux individus non-étiquetés. Pendant la phase d'apprentissage, l'algorithme supervisé cherche à apprendre les relations entre les données en entrée et les étiquettes qui leur correspondent, afin de déduire des règles de l'apprentissage. Ces règles sont par la suite utilisées pour prédire les étiquettes pour un nouveau jeu de données. Le principal avantage de l'apprentissage supervisé est qu'il nous permet de recueillir des informations et de produire des connaissances à partir des expériences passées. Les algorithmes de classification supervisés prévoient des valeurs discrètes. Ces algorithmes visent à classer les données en entrée en deux ou plusieurs catégories.

L'apprentissage profond et l'une des nouvelles techniques de l'apprentissage automatique. Il diffère de l'apprentissage automatique traditionnel par l'absence de l'étape d'extraction des caractéristiques. Cette dernière est une étape très difficile et très couteuse en termes de temps. Les algorithmes de l'apprentissage profond sont basés sur l'utilisation des réseaux de neurones artificiels (ou Artificiel Neural Networks en anglais) ; ces réseaux comportent plusieurs neurones et de nombreuses couches cachées hiérarchiques qui constituent une structure complexe. Ceci explique le sens du terme *profond* [9]. Ces neurones permettent de gérer une grande partie des données

brutes. À partir de ces dernières, et grâce aux multiples transformations non linéaires dans les multiples couches de traitement, ce modèle peut extraire des caractéristiques et apprendre à travers chaque couche, qui donne un haut niveau d'abstraction [23].

Dans le domaine de la cyber sécurité, les algorithmes d'apprentissage traditionnels sont beaucoup plus préférés, comparés à l'apprentissage profond [2]. Cela est dû à la nécessité d'interprétation rapide des prédictions des modèles, afin de permettre aux ingénieurs de sécurité d'intervenir est réparer les failles détectées pour défendre au mieux contre les différentes attaques. Pour cela nous avons opté pour l'utilisation des méthodes d'apprentissage traditionnelles, avec la présence des données étiquetées, donc nous avons choisi l'apprentissage supervisé. Différents classificateurs existants implémentent l'apprentissage automatique supervisé. Dans ce projet, nous testons les classificateurs suivants [25] :

1. **Arbre de décision (DT)** : Les arbres de décision sont très largement considérés comme étant les modèles d'apprentissage automatique les plus interprétables pour les problèmes de classement. Un arbre de décision est constitué d'un ensemble de nœuds où chacun est étiqueté par l'une des variables d'entrée (i.e., caractéristique), et pour lequel chacune des feuilles est étiquetée par 0 ou 1 (classe d'individu) [25].
2. **Foret d'arbres décisionnels (RF)**: Cet algorithme construit plusieurs arbres de décision, chaque arbre étant associé à différents scénarios et différentes variables initiales. Ce type d'algorithme est utilisé pour la modélisation prédictive de la classification et de la régression [25]. Dans ce projet, deux types de RF sont expérimentés : le premier à base de classification (**RF_C**) et l'autre à base de régression (**RF_R**).
3. **Machine à vecteurs de support (SVM)**: Le principe des SVM consiste à ramener un problème de classification à un hyperplan (feature space) dans lequel les données sont séparées en plusieurs classes dont la frontière est la plus éloignée possible des points de données. Le concept de frontière implique que les données soient linéairement séparables. Pour y parvenir, les SVMs font appel à des fonctions mathématiques permettant de projeter et de séparer les données dans l'espace vectoriel, les vecteurs de support étant les données les plus proches de la frontière. C'est la frontière

la plus éloignée de tous les points d'entraînement qui est optimale, et qui présente donc la meilleure capacité de généralisation [25].

4. **Voisin le plus proche (KNN)**: Dans un problème de classification, on retiendra la classe la plus représentée parmi les k sorties associées aux k entrées les plus proches de la nouvelle entrée x [25].
5. **Naïve Bayésiennes (NB)** : La classification naïve bayésienne est un type de classification probabiliste simple basé sur le théorème de Bayes avec une forte indépendance [25]. Deux types de classificateurs NB sont examinés dans ce projet : Gaussian et Bernoulli Naïve Bayes.
6. **Le perceptron multicouche (MLP)**: Est un type de réseau neuronal artificiel organisé en plusieurs couches au sein desquelles une information circule de la couche d'entrée vers la couche de sortie [25].
7. **Régression logistique (LR) :** Est un modèle statistique qui permet de déduire la relation entre un ensemble de variables prédictives (caractéristiques) avec une variable aléatoire binomiale (étiquette) ; il utilise pour cela une fonction logistique comme une fonction de lien. Il permet aussi de prédire la probabilité qu'une variable d'entrée est reliée à une valeur de sortie binaire (0 de 1) à partir de l'optimisation des coefficients de régression [25].

4. Évaluation de la performance des modèles

Pour l'évaluation des classificateurs, deux facteurs doivent être identifiés précisément : la méthode d'évaluation et les mesures de performance considérées.

4.1. Méthode d'évaluation

Dans notre étude, nous avons adopté la validation croisée (*cross-validation* en anglais) [24] comme méthode d'évaluation du modèle car elle est connue comme la méthode d'estimation la plus fiable. La méthode de validation croisée consiste à diviser l'ensemble des données collectées en *k* échantillons. Un des *k* échantillons est sélectionné pour validation alors que les *k-1* autres échantillons sont utilisés pour l'apprentissage. Le processus de validation se répète *k* fois, en sélectionnant à chaque fois un échantillon différent pour la validation. En effet, à chaque fois, un modèle

différent est généré et sa performance est mesurée et sauvegardée. La moyenne et l'écart type des *k* scores de performances peuvent être calculés pour estimer le biais et la variance de la performance de validation [23, 24]. La figure 2.9 montre un exemple sur la validation croisée avec k=5. L'ensemble de données est divisé en 5 parties égales. Les individus pour la partie du test sont désignés par des jetons blancs alors que les individus des parties d'apprentissage sont désignés par des jetons noirs. Dans le cas où le nombre des individus n'est pas divisible par le nombre des parties souhaité, la dernière partie doit inclure le reste de la division [23]. Dans notre projet nous adoptons la validation croisée avec k = 10.

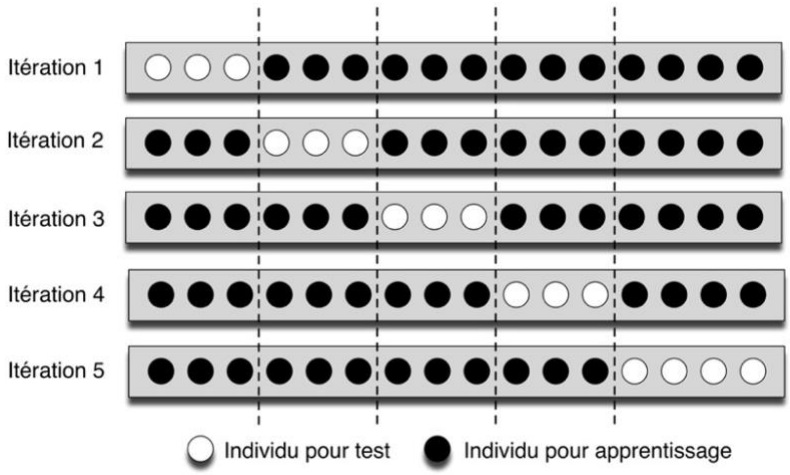


Figure 3.3. Validation croisée avec k=5 [23]

4.2. Mesures de performance

Concernant les mesures de performance, nous avons également sélectionné les mesures les plus couramment utilisées pour évaluer la performance des classificateurs. Toutes ces mesures sont basées sur la matrice de confusion décrite dans le tableau 3.2 :

		Prédiction	
		XSS	**Bénin**
Nature de script	**XSS**	Vrai positif (VP)	Faux négatif (FN)
	bénin	Faux positif (FP)	Vrai négatif (VN)

Tableau 3.4. Matrice de confusion.

Selon la matrice de confusion, un vrai positif (VP) indique qu'un script d'attaque XSS a été bien classé, tandis qu'un faux négatif (FN) indique qu'il a été classé comme bénin. Un vrai négatif (VN) indique qu'un script bénin est correctement classé, alors qu'un faux positif (FP) indique qu'il a été incorrectement classé comme un vecteur d'attaque XSS. Voici la liste des mesures adoptées pour la comparaison des classificateurs :

1. **Accuracy :** La métrique la plus couramment utilisée pour juger un modèle. C'est un rapport entre l'observation correctement prédite et le total des observations. Elle est la mesure de performance la plus intuitive surtout pour l'ensemble de données équilibrées comme celui de notre cas.

$$Accuracy = \frac{VP + VN}{VP + FP + FN + VN}$$

2. **Précision :** C'est la proportion des vecteurs d'attaque XSS effectivement identifiées corrects par le classificateur.

$$Précision = \frac{VP}{VP + FP}$$

3. **Rappel :** C'est la proportion des vecteurs d'attaque XSS réelles qui ont été effectivement identifiées par le classificateur.

$$Rappel = \frac{VP}{VP + FN}$$

4. **F1-score :** C'est la moyenne entre la précision et le rappel. Par conséquent, ce score prend en compte à la fois les faux positifs et les faux négatifs. F1-Score est généralement plus utile que l'Accuracy, surtout si la répartition des classes est inégale.

$$F1 - Score = 2 * \frac{Precision * Rappel}{precision + Rappel}$$

4.3. Résultats de l'évaluation

On a appliqué la méthode de validation croisée (avec k=10) sur les classificateurs choisis individuellement, sur l'ensemble de données collectées. Nous avons obtenu les résultats décrits dans le tableau 3.5 où les meilleures valeurs sont indiquées en **gras**.

Classificateur	Matrice de Confusion				Précision	Rappel	Accuracy	F1-Score
	VP (%)	FN (%)	VN (%)	FP (%)	(%)	(%)	(%)	(%)
DT	97.92	2.08	99.32	0.68	99.31	97.92	98.63	98.61
RF_R	97.81	2.19	99.48	0.52	99.40	97.81	98.64	98.63
RF_C	98.41	1.59	99.28	0.72	99.27	98.41	98.86	98.86
SVM	95.83	4.17	99.14	0.86	99.11	95.83	97.49	97.44
KNN	**98.82**	**1.18**	99.29	0.71	99.29	**98.82**	**99.06**	**99.05**
Gaussian NB	76.61	23.39	**99.91**	**0.09**	**99.88**	76.61	88.26	86.70
Bernoulli NB	96.49	3.51	78.60	21.40	81.86	96.49	87.54	88.57
MLP	98.64	1.36	99.12	0.88	99.12	98.64	98.89	98.88
LR	96.60	3.40	98.64	1.36	98.62	96.60	97.62	97.59

Tableau 3.5. Performance des différents classificateurs.

Le but de cette étude est d'éliminer, voire réduire le risque qu'un utilisateur soit affecté par des attaques XSS (c.-à-d. augmenter les vrais positifs). Après avoir testé plusieurs classificateurs, nous avons trouvé que l'algorithme du Voisin le plus proche (KNN) est le mieux adapté. Ce dernier donne une valeur très élevée de vrais positifs (99.82%) avec un taux raisonnable de faux négatifs (1.18 %). Le tableau 3.5 montre aussi la capacité des algorithmes de classification supervisés simples et interprétables à détecter les attaques XSS les plus courantes.

5. Implémentation

Dans cette section, nous détaillons les aspects techniques liées à la réalisation de notre projet. Cela inclut, l'ensemble des outils utilisés pour mettre en œuvre les expérimentations citées auparavant, la démarche à suivre pour la réalisation de notre étude et l'interface graphique de notre démo.

5.1. Outils d'implémentation

Pour l'expérimentation citée auparavant, nous avons utilisé le langage python avec ses différentes librairies implémentant l'apprentissage automatique. Le langage python est aussi utilisé pour le développement d'une démo qui permet le partage en ligne du modèle d'apprentissage construit. La plateforme Google Colab est utilisée pour l'exécution des programmes pour sa simplicité et intégralité.

- **Google Colab[1] :** Est un service cloud, offert par Google gratuitement, basé sur Jupyter Notebook et destiné à la formation et à la recherche dans l'apprentissage automatique. Cette plateforme permet d'entraîner des modèles d'apprentissage automatique directement dans le cloud, sans avoir besoin d'installer quoi que ce soit sur un ordinateur local à l'exception d'un navigateur.
- **Python[2] :** Un langage de programmation utilisé pour l'extraction des caractéristiques et l'implémentation du modèle d'apprentissage automatique sous Google Colab. Différentes librairies de Python ont été utilisées pour la réalisation de notre projet :
    - **Pandas[3]** : Une bibliothèque permettant la gestion des données sous forme de tableaux structurés. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles.
    - **Sklearn[4]** : Une bibliothèque d'outils dédiés à l'utilisation des algorithmes d'apprentissage automatique, des méthodes de validations et des mesures de performances.
    - **Imblearn[5]** : Une bibliothèque qui fournit les outils nécessaires pour traiter la classification avec des classes déséquilibrées.
    - **Joblib[6]** : Utile pour sauvegarder des structures de données dans un fichier pour pouvoir les recharger plus tard. Dans notre cas, elle a été

[1] Google Colab: https://colab.research.google.com/?utm_source=scs-index
[2] Python : https://docs.python.org/fr/3/tutorial/
[3] Bibliothèque pandas: https://pandas.pydata.org/
[4] Bibliothèque sklearn : https://scikit-learn.org/
[5] Bibliothèque imblearn : https://imbalanced-learn.org/stable/
[6] Bibliothèque joblib : https://joblib.readthedocs.io/

utilisée pour sauvegarder notre modèle d'apprentissage à base de KNN afin qu'on puisse le partager et l'utiliser dans la démo.

- **Gradio**[7] : Utile pour créer des démos et des applications Web d'apprentissage automatique et de la science des données. Avec Gradio, il est possible de créer rapidement des interfaces utilisateur autour des modèles d'apprentissage automatique construits et de les partager.

5.2. Processus de développement

Nous avons créé une démo qui permet aux navigateurs de distinguer les scripts malveillants qui peuvent être utilisés pour lancer des attaques XSS, de ceux bénins. Pour cela, nous avons suivi les étapes suivantes :

1. Collecter un ensemble de données contenant des scripts d'attaques XSS et d'autres bénins. Deux fichiers textes (**xss.txt** et **benign.txt**) sont créés contenant respectivement la liste des scripts XSS et bénins.
2. Filtrer les deux fichiers en supprimant les scripts dupliqués.
3. Faire l'extraction des caractéristiques des deux types de scripts et sauvegarder les résultats dans un fichier nommé **dataset_xss.csv**.
4. Appliquer la méthode **Shuffle()** pour mélanger les éléments de l'ensemble de données de manière aléatoire.
5. Appliquer la méthode **NearMiss()** pour générer un ensemble de données équilibré en utilisant le sous-échantillonnage.
6. Tester un ensemble de classificateurs en utilisant la méthode de validation croisée et déterminer le meilleur classificateur.
7. Sauvegarder le meilleur classificateur en utilisant la bibliothèque Joblib pour une utilisation ultérieure, dans le fichier **trained_model.pkl**.
8. Installer le package Gradio pour la prise en charge de l'interface graphique pour Python.
9. Préparer et lancer une interface graphique créée avec Gradio.

[7] Bibliothèque gradio : https://gradio.app/

5.3. Interface graphique de démo

L'image de la figure 3.4 présente la page Web qui correspond à notre démo nommée *XSS Checker*. Selon le type du script donné en entré (XSS ou bénin), XSS Checker fait appel à notre modèle enregistré et affiche sa prédiction comme le monte les figures 2.5 et 3.6 respectivement.

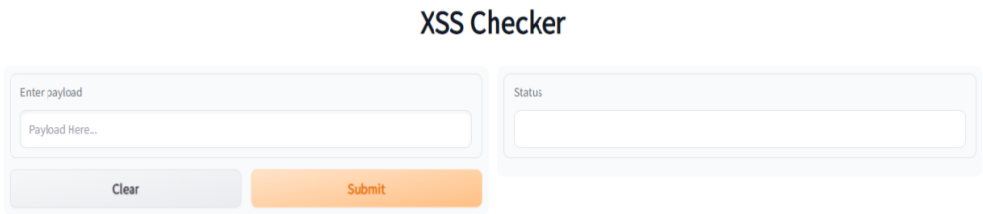


Figure 3.4. Page d'accueil de XSS Checker.

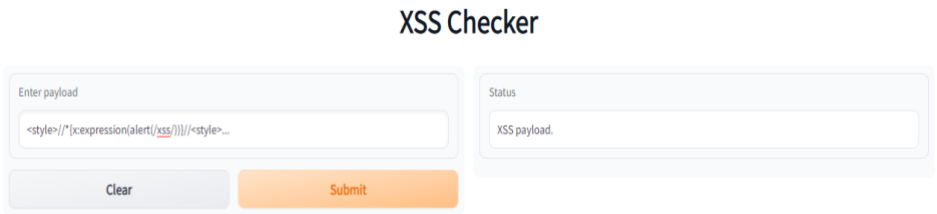


Figure 3.5. Cas d'un script XSS.

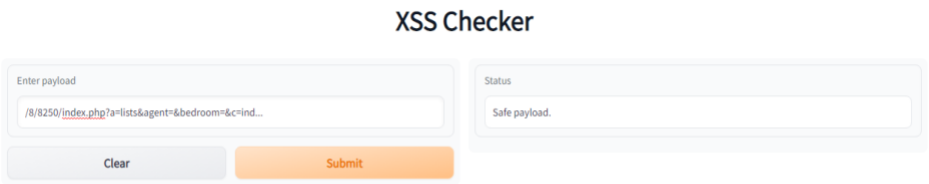


Figure 3.6. Cas d'un script bénin.

6. Conclusion

Dans ce chapitre, nous avons présenté la démarche à suivre pour le développement d'un modèle efficace pour la détection des attaques XSS. Après expérimentation, nous avons trouvé qu'un modèle à base de KNN peut distinguer efficacement les scripts XSS

des scripts bénins avec un F1-score qui dépasse 99%. Une interface graphique simple est utilisée pour l'expérimentation de notre modèle avec de nouveaux scripts et donc de nouvelles attaques XSS. L'apprentissage automatique a montré encore une fois sa capacité de défense contre une cyber attaque redoutable telle que l'attaque XSS.

CONCLUSION GENERALE

CONCLUSION GENERALE

Avec le développement de la technologie Web et l'augmentation explosive de la quantité de l'information, la sécurité web devient de plus en plus importante. Les vulnérabilités du web font partie des problèmes de sécurité ; les attaquants exploitent ces vulnérabilités pour lancer des attaques Web diverses. L'attaque XSS est l'une des plus anciennes et des plus reconnues par la communauté de la cyber sécurité. Malheureusement, l'émergence des technologies Web avancées et des nouveaux styles de programmation, rend leurs détections une des tâches les plus difficiles. Alors que les nouvelles technologies telles que les services Web et les API permettent des interactions et des échanges de données complexes entre les clients et les serveurs du réseau, les nouveaux styles de programmation telles que AJAX, CSS3 et HTML5 introduisent de nouvelles failles d'injection aux applications Web. Par conséquent, il est très important de détecter ces vulnérabilités aussi bien pour les applications en phase de création que pour les applications en cours d'exécution.

Dans ce cadre, nous avons proposé une approche pour la détection des attaques XSS. L'approche proposée se base sur l'utilisation des modèles d'apprentissage traditionnels. La réalisation de ces modèles a été faite à partir d'un ensemble de données collectés et nettoyées. Après plusieurs expérimentations, nous avons constaté l'efficacité des modèles d'apprentissage traditionnels, notamment le modèle à base de KNN, dans la détection des attaques XSS.

Au terme de ce travail, nous espérons continuer dans ce domaine en généralisant notre approche pour explorer d'autres caractéristiques discriminatives et améliorer la performance de notre modèle.

BIBLIOGRAPHIE & WEBOGRAPHIE

BIBLIOGRAPHIE

[1] Amy Shuen, *Web 2.0: A Strategy Guide Business thinking and strategies behind successful Web 2.0 implementations.* O'Reilly Media, Inc., 2008. ISBN: *0596529961.*

[2] Abdelhakim Hannousse, Salima Yahiouche, Mohamed Cherif Nait-Hamoud, Twenty-two years since revealing cross-site scripting attacks: a systematic mapping and a comprehensive survey. arXiv, CoRR abs/2205.08425, 2022. doi: 10.48550/arXiv.2205.08425.

[3] Abdelhakim Hannousse, Salima Yahiouche, *Towards benchmark datasets for machine learning based website phishing detection: An experimental study*, Engineering Applications of Artificial Intelligence, Vol. 104, 2021, 104347, doi: 10.1016/j.engappai.2021.104347.

[4] Guillaume Harry, Failles de sécurité des applications Web Principes, parades et bonnes pratiques de développement, Rapport technique, hal-00736013, 2012.

[5] Jeremiah Grossman, *XSS Attacks: Cross-site Scripting Exploits and Defense*, Syngress, 2007. ISBN: 1597491543.

[6] Park Joon S. and Ravi Sandhu, *Secure cookies on the Web,* IEEE internet computing, Vol. 4, N. 4, pp. 36-44, 2000.

[7] Keith Jeremy and Jeffrey Sambells. *DOM scripting: Web design with Javascript and the document object model*. Apress, Second Edition, 2011. ISBN: 1430233907.

[8] M. Johns, B. Engelmann, J. Posegga, *Xssds: Server-side detection of cross-site scripting attacks*, In proceedings of the Annual Computer Security Applications Conference, ACSAC, IEEE, Anaheim, CA, USA, 2008, pp. 335–344. doi: 10.1109/ACSAC.2008.36.

[9] S. Sundareswaran, A. Squicciarini, *Xss-dec: A hybrid solution to mitigate cross-site scripting attacks*, In proceedings of the 26th Annual IFIP Annual Conference on Data and Applications Security and Privacy, DBSec, Springer, Paris, France, 2012, pp. 223–238. doi:10.1007/978-3-642-31540-4_17.

[10] F. Sun, L. Xu, Z. Su, *Client-side detection of XSS worms by monitoring payload propagation*, In proceedings of the 2009 European Symposium on Research in Computer Security, ESORICS, Springer, Saint-Malo, France, 2009, pp. 539–554. doi:10.1007/978-3- 642- 04444- 1_33.

[11] A. Nunan, E. Souto, E. Dos Santos, E. Feitosa, *Automatic classification of cross-site scripting in web pages using document-based and URL-based features*, In proceedings of the IEEE Symposium on Computers and Communications, ISCC, IEEE, Cappadocia, Turkey, 2012, pp. 702–707. doi:10.1109/ISCC.2012.6249380.

[12] S. Goswami, N. Hoque, D. Bhattacharyya, J. Kalita, *An unsupervised method for detection of XSS attack*, International Journal of Network Security 19 (5) (2017), pp. 761–775. doi:10.6633/IJNS.201709.19(5).14.

[13] F. Mokbal, W. Dan, A. Imran, L. Jiuchuan, F. Akhtar, W. Xiaoxi, *MLPXSS: An integrated XSS-based attack detection scheme in web applications using multilayer perceptron technique*, IEEE Access 7 (2019), pp; 100567–100580. doi:10.1109/ ACCESS.2019.2927417.

[14] F. Mereani, J. Howe, *Preventing cross-site scripting attacks by combining classifiers*, In proceedings of the 10th International Joint Conference on Computational Intelligence, IJCCI, SciTePress, Seville, Spain, 2018, pp. 135–143. doi:10.5220/0006894901350143.

[15] J. Zhang, Y.-T. Jou, X. Li, *Cross-site scripting (xss) detection integrating evidences in multiple stages*, In Proceedings of the Annual Hawaii International Conference on System Sciences, HICSS, AIS Electronic Library, Grand Wailea, Maui, Hawaii, USA, 2019, pp. 7166–7175. doi:10.24251/HICSS.2019.860.

[16] X. Li, W. Ma, Z. Zhou, C. Xu, *XSS attack detection model based on semi-supervised learning algorithm with weighted neighbor purity*, In proceedings of the 19th International Conference on Ad-Hoc Networks and Wireless, ADHOC-NOW, Springer, Bari, Italy, 2020, pp. 198–213. doi:10.1007/978- 3- 030- 61746- 2_15.

[17] Y. Zhou, P. Wang*, An ensemble learning approach for XSS attack detection with domain knowledge and threat intelligence*, Computers & Security 82 (2019), pp. 261–269. doi:10.1016/j.cose.2018.12.016.

[18] R. Wang, X. Jia, Q. Li, S. Zhang, *Machine learning based cross-site scripting detection in online social network*, In proceedings of the 16th IEEE International Conference on High Performance Computing and Communications, HPCC, IEEE, Paris, France, 2014, pp. 823–826. doi:10.1109/HPCC.2014.137.

[19] S. Rathore, P. Sharma, J. Park, *XSSClassifier: An efficient XSS attack detection approach based on machine learning classifier on SNSS*, Journal of Information Processing Systems 13 (4) (2017), pp. 1014–1028. doi:10.3745/JIPS.03.0079.

[20] R. Kadhim, M. Gaata, *A hybrid of CNN and LSTM methods for securing web application against cross-site scripting attack*, Indonesian Journal of Electrical Engineering and Computer Science 21 (2) (2020) 1022–1029. doi:10.11591/ijeecs. v21.i2.pp1022- 1029.

[21] P. Chaudhary, B. B. Gupta, X. Chang, N. Nedjah, K. T. Chui, *Enhancing big data security through integrating XSS scanner into fog nodes for SMEs gain*, Technological Forecasting and Social Change 168 (2021) 120754. doi:10.1016/j. techfore.2021.120754.

[22] Z. Liu, Y. Fang, C. Huang, J. Han, *GraphXSS: An efficient XSS payload detection approach based on graph convolutional network*, Computers & Security (2022) 102597. doi:10.1016/j.cose.2021.102597.

[23] Aid Mohammed Amine, *La détection des webshells pour assurer la sécurité des serveurs web.* Mémoire Master, Université 8 Mai 1945, Guelma, Département Informatique, 2021.

[24] Michael W Browne, *Cross-Validation Methods*, Journal of Mathematical Psychology, 44(1), pp. 108-132, 2000.

[25] Dipanjan Sarkar, Raghav Bali, and Tushar Sharma. *Practical Machine Learning with Python: A Problem-Solver's Guide to Building Real-World Intelligent Systems.* APress, USA, 1st edion, 2017.

WEBOGRAPHIE

[W1] Open Web Application Security Project (OWASP), https://owasp.org, last access May 2022.

[W2] Amit Klein, DOM Based Cross Site Scripting or XSS of the Third Kind: A look at an overlooked flavor of XSS, webappsec, http://www.webappsec.org/projects/articles/071105.txt, July 2005, last access May 2022.

[W3] Kaspersky.org, Qu'est-ce qu'une attaque XSS (Cross-Site Scripting) ? Définition et explication, https://www.kaspersky.fr/resource-center/definitions/what-is-a-cross-site-scripting-attack, last access May 2022.

[W4] Guy Podjarny, *XSS Attacks: The Next Wave*, https://snyk.io/blog/xss-attacks-the-next-wave/, June 2017, last access May 2022.