07/03/2019

VAYSSIER Sylvain

[nom de la société]

Rapport de Projet

Offensive Security Console

# Préface

Table des matières

[Préface 1](#_Toc3386674)

[I- Introduction et présentation générale 3](#_Toc3386675)

[1) Présentation de l’entreprise 3](#_Toc3386676)

[2) Présentation de la mission 4](#_Toc3386677)

[3) Cahier des charges 4](#_Toc3386678)

[1 - Contexte et définition du projet 4](#_Toc3386679)

[2 - Objectif du projet 4](#_Toc3386680)

[3 - Périmètre du projet 4](#_Toc3386681)

[4 - Description fonctionnelle des besoins 5](#_Toc3386682)

[5 - Délais de réalisation 5](#_Toc3386683)

[II- Gestion du projet 5](#_Toc3386684)

[1) Présentation du contexte et définitions des objectifs 5](#_Toc3386685)

[2) Organisation 6](#_Toc3386686)

[1- Descriptif des tâches 6](#_Toc3386687)

[2- Modèle prévisionnel de Gant 7](#_Toc3386688)

[3- Analyse de risque 7](#_Toc3386689)

[4- Le choix des technologies 9](#_Toc3386690)

[5- Représentation UML 10](#_Toc3386691)

[III- OffSec : Back-End Python 12](#_Toc3386692)

[1) Python 12](#_Toc3386693)

[2) Le script 13](#_Toc3386694)

[IV- OffSec : Back-End Springboot 16](#_Toc3386695)

[1) Présentation de Springboot 16](#_Toc3386696)

[2) Création des classes 17](#_Toc3386697)

OffensiveSecurityConsole

# Introduction et présentation générale

## Présentation de l’entreprise

Pour la création de ce projet nous avons intégré l’entreprise Be-Kortalys, une filiale du groupe Be-YS. Cette filiale est une marque dédiée à l’ingénierie de la sécurité des systèmes d’informations et des centres de données.

Be-Ys est une entité créée dans le but d’offrir une offre universelle pour tout activité économique nécessitant la maitrise de l’identité numérique. Afin de pouvoir déployer l’éventails de ces compétences cette structure de plus de 1000 employés s’est redéployé en petit groupe chacun assurant sa spécialité :



Le groupe Clermontois Be-Almerys fondé en 2010, responsable des flux transactionnel des mutuelles de santé française, accueille dans ses locaux son petit frère, le groupe Be-Kortalys, le pôle spécialisé de cyber-sécurité. Son rôle ets la vente des prestations B2B, à savoir de l’hébergement sécurisé et des audits de pénétration réseau sur les système client tout en assurant la sécurité des système d’informations du groupe.

La cyber-sécurité est un enjeu majeur et les entreprises qui veulent investir dans ce domaine et se démarquer de la concurrence se doivent d’investir efficacement dans la sécurité des réseaux et dans la qualification de son personnel. C’est dans cette démarche que Be-Kortalys est passé maître dans l’art de perforer le marché en déployant un moyen d’hébergement ultra-sécurisé (même selon les standard actuel), tout en déployant l’infrastructure logicielle et matérielle nécessaire pour permettre à cette entreprise de bénéficier d’un fort soutien de l’État tant que par sa reconnaissance en tant qu’Organisme d’Importance Vitale mais aussi comme un des précurseurs et fer de lance dans le déploiement du marché français de la sécurité informatique.

## Présentation de la mission

Dans l’optique d’une prestation B2B, le Security Operation Center de Be-Kortalys nous a fait la requête de lui fournir, pour son usage interne une console de sécurité offensive capable de permettre d’automatiser tout ou une partie des tâches les plus fréquemment utilisé dans les audits de cyber-sécurité réalisé chez leur client. Ainsi il nous a été demandé à moi et mon groupe de travail de réalisé une application accessible depuis n’importe quel support qui pourrait effectué des actions (planifiés à l’avance ou non) de test d’intrusion réseau, avec notamment comme première implémentation un scanner de vulnérabilités.

Nous avons donc rencontré M Rémi Charbonnel, notre commanditaire de mission et le responsable des test d’intrusion réseau à travers un cahier des charges :

## Cahier des charges

### 1 - Contexte et définition du projet

Ce projet fait suite à un outil développé en interne permettant de réaliser des scans de vulnérabilités et de produire un rapport de manière automatisée.

Le script original se base sur des outils couramment utilisés dans le domaine de la cyber sécurité, permettant, à partir d’une liste d’IP, de scanner des systèmes afin de détecter les services disponibles et de détecter de potentielles failles de vulnérabilités.

### 2 - Objectif du projet

Ce projet se place comme l’évolution de cet outil, afin d’industrialiser son utilisation et de fournir des fonctionnalités supplémentaires. Le projet a pour objectif de fournir une application web afin de contrôler des actions en sécurité offensive et de générer des rapports, exporter au format PDF.

### 3 - Périmètre du projet

L’application doit permettre de gérer des actions – « jobs » – de sécurité offensive à l’encontre de systèmes cibles

### 4 - Description fonctionnelle des besoins

Ce qui est attendu pour la « gestion » c’est le fait de :

• Lancer des jobs

• Suivre le statut des jobs (en cours, terminé, en pause, en erreur)

• Accéder aux résultats des jobs Cette fonctionnalité étant le cœur de l’application, il est attendu que ces actions soient pilotables via une API afin de pouvoir être interfacé depuis différentes sources…

L’application devrait persister les données récoltées par les actions de sécurités afin de pouvoir traiter ultérieurement les données Le stockage de ces données devra être uniformisé afin de pouvoir ajouter différents formats en entrée (selon les types de jobs exécutés).

L’application devra comporter une application de type « front », proposant une UI permettant aux utilisateurs de contrôler la plateforme.

L’accès à la plateforme, API et application « front », devra être soumis à un contrôle d’accès afin de sécuriser les données.

### 5 - Délais de réalisation

La date de livraison attendue de notre projet est pour le 29 avril.

# Gestion du projet

## 1) Présentation du contexte et définitions des objectifs

Induit par notre formation de concepteur d’application / Expert en cyber-sécurité, le groupe nous a donc demandé par l’intermédiaire de notre commanditaire, une application intégrée dans l’environnement de travail du groupe. Pour cela, nous avons été affecté à un groupe de travail responsable de ce projet. Monsieur Yegor Nickolov, Monsieur Mayet Lionel ainsi que moi-même avons eu le plaisir de pouvoir travailler ensemble sur ce projet dans un contexte professionnel et encadré avec une grande place pour exprimer notre créativité et notre autonomie.

Durant notre formation j’ai pu être formé à Springboot, un Framework de développement Back-End à la très grande notoriété. Et c’est aussi sur cet environnement de travail que certaines des applications du groupe sont développées, et c’est pourquoi il fut simple de choisir sur quelle technologie s’appuierait le cœur de l’application.

Guidé par notre responsable pédagogique, Monsieur Benjamin Geogeault, et notre commanditaire, dans notre recherche documentaire, ils nous permirent de préparer les fondements de l’applications. Que cela soit à travers leur présences hebdomadaire et leur attention dès que nécessaire, ils nous recommandèrent d’adopter une démarche AGILE concernant le développement de notre application. En effet, certaines implémentations de fonctions étaient optionnelles et ils faillaient pouvoir livrer une application même inachevée avant la fin du délai imparti. De plus nous avons eu l’occasion d’avoir une formation sur la méthode AGILE SCRUM au début de notre formation, et il était évident que ce projet serait construit autour d’une méthode modulaire et adaptable.

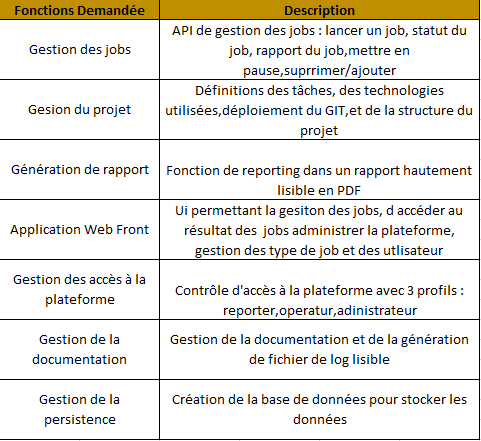
De plus, nous avions un programme complet quand à ce que nous allions voir comme cours durant notre formation, ce qui nous permis de choisir les technologies que nous allions étudier comme technologies de développement dans le projet.

Cependant, cette décision nous coûta beaucoup au lancement de ce projet, puisque nous ne connaissions à ce moment-là quasiment rien des technologies que nous allions utiliser. A contrario, cet obstacle nous permis de nous concentrer en amont sur une gestion de projet précise et sur une quantité importante de recherche documentaire quant aux meilleur moyens d’implémenter les fonctions demandées.

Pour nous permettre de garder trace de notre avancé et afin de pouvoir évaluer nos délais, nous devions rendre un modèle de compte rendu d’activité (*cf : Annexe : Fiche CRA*) de façon hebdomadaire, ainsi qu’une journée par semaine (le jeudi en général) dans les locaux du groupe afin de permettre aux différents groupes de travail de communiquer avec le commanditaire respectifs et afin de pouvoir échanger entre les élèves de la formations, qui s’avéra être le moyen le plus pédagogique de tous pour trouver les solutions.

## 2) Organisation

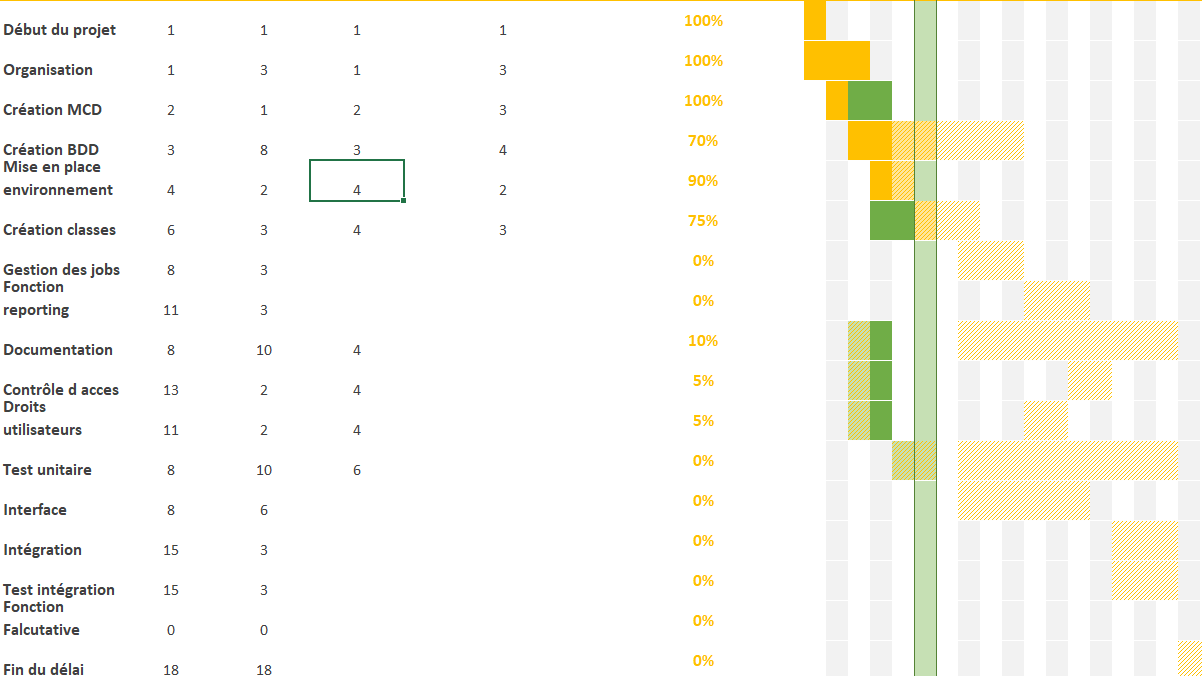
### 1- Descriptif des tâches

 La première partie de la gestion du projet a donc été de lister et décrire les tâches nécessaires pour la création de l’application. Afin de répondre au mieux, nous avons ensemble lister à nouveau les fonctions demandées :

*Le descriptif des tâches de l’application OffSec*

Suite à cela, il nous fallut convenir d’une durée pour les tâches dont nous avions convenu comme le montre le modèle de Gant ci-dessous.

### 2- Modèle prévisionnel de Gant



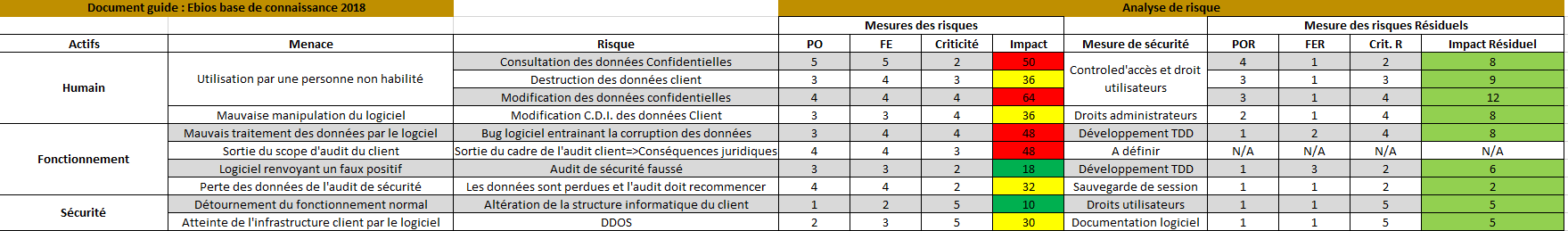
*Le modèle prévisionnel de données*

Ce modèle créer avec Excel était censé être notre « guideline » en matière de délai. Cependant nous avions les délais affectés à nos tâches en fonction des technologies que nous connaissions mais ne savions pas l’ampleur des tâches couverte par les Framework que nous utilisions. Ce manque d’expérience avec les technologies fera que notre modèle prévisionnel de Gant est très décalé par rapport à la réalité, et n’a pas du tout pu nous servir de support.

Au lieu de cette méthode, nous avons donc adopté une méthode un peu proche de la méthode AGILE puisqu’elle reposait sur nos rendez-vous hebdomadaire et sur le compte rendu d’activité. En effet, en devant rendre compte régulièrement, le groupe de travail devait donc se réunir pour faire le point à la manière des « SPRINT » de la méthode SCRUM. Il s’avéra tout de même difficile d’estimer les délais des tâches que nous choisissions de faire, et donc difficile d’estimer un modèle prévisionnel valide.

### 3- Analyse de risque

Lors de notre formation, nous avons eu une sensibilisation au risque inhérent dans le développement des applications. Inhérent à notre parcours, et à l’actualité, la sécurité de l’application et le contrôle des facteur environnementaux vulnérable se devait d’être une condition sine qua none. C’est dans cette optique que nous avons donc réalisé une analyse risque selon la méthode EBIOS 2018.



*Le rapport final de l’analyse de risque menée sur cette application*

Dans cette analyse, nous avons émis des scenarii pour étudier les menaces potentielles et les vulnérabilités qui en découlaient. Nous avons retenu que les actifs engendrant le plus de menace vis-à-vis de notre application était :

* L’actif Humain : Dans la sécurité l’actif Humain est la seule pièce qui reste complétement vulnérable
* L’actif Fonctionnement : Notre application pouvant héberger différents jobs de pénétration réseau, il faut émettre la possibilités d’accès à des données confidentielles, ou à des données clients rendant le fonctionnement même de l’application comme source de menace
* L’actif Sécurité : Les test d’intrusion peuvent parfois s’avérer être lourd pour l’infrastructure d’un client. Par exemple, le scanner que nous avons développé utilise MassScan et NMap, qui peuvent s’avérer une charge non négligeable pour un serveur. Les risques de cette catégories couvre les risque de sécurité envers la structure du client (la cible du test d’intrusion)

Nous avons évaluer les risques les plus probables et nous les avons ensuite noté selon l’échelle des risques (cf Annexe : Analyse de risque : source) selon leur probabilité d’occurrence (PO) leur faisabilité d’exploitation (FE) et leur criticité (évaluer sur l’impact en matière de Confidentialité Intégrité Disponibilité (cf Glossaire : CDI)). Ces valeurs, une fois multiplier entre elle permettent de démarquer les risques les plus important à pallier.

Selon la méthode d’analyse de risque EBIOS 2018, les mesures de sécurités contre les risques sont des « amortisseurs» a risques et il est nécessaire ainsi de réévaluer les risques sécuritaires avec les valeurs corrigés. Il est nécessaire de réitérer ce processus jusqu’à ce que les risques passent dans la zone d’acceptabilité.

Nous pouvons constater que les mesures de sécurité pour réduire les menaces sont des moyens efficaces et plutôt simple à mettre en place :

* Le développement de l’application se fera avec la méthode des Test Driven Development
* Le développement d’un système de login et de droits d’accès différents en fonction des rôles
* Le développement d’un moyen de persister les données ou de sauvegarder les sessions
* Une documentation logiciel efficace

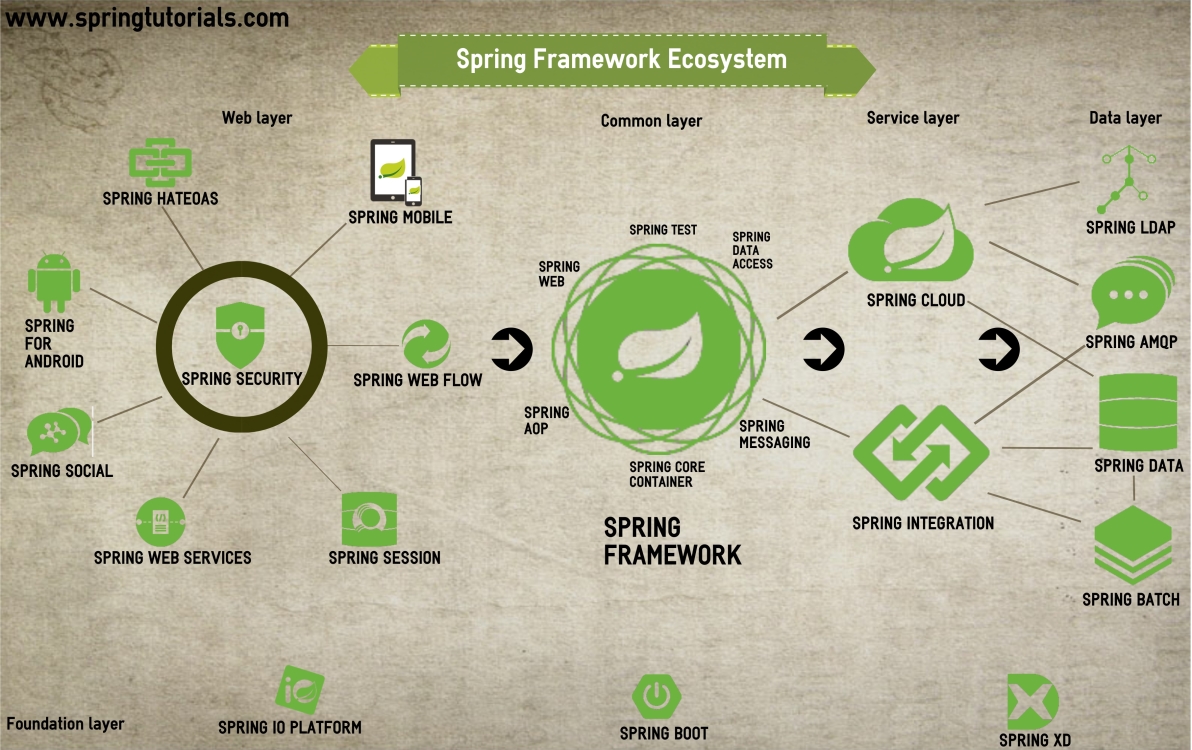
Il faut remarquer que la ligne concernant la menace « sorti du scope de l’audit client » bien qu’induite par notre application, la source en est indépendante, puisque l’utilisateur est censé être celui qui connait le périmètre légale quant à l’utilisation de cette application.

### 4- Le choix des technologies

Maintenant que la plupart de l’encadrement était disposé il ne nous restait qu’à choisir ls technologies.

Le Framework Springboot est devenu rapidement un Framework incontournable dans le développement Back-End d’application. Reposant sur la Java Virtual Machine, cet environnement de travail pour le JAVA, permet notamment de déployer son application sur n’importe quel système et peut très bien s’adapter à un déploiement Ansible ou encore Docker. Surcouche du déjà efficace Spring Framework, il s’accommode du gestionnaire de dépendance Maven (et peut même utiliser Gradle) permet d’adapter son application avec les composant nécessaires, rendant ainsi tout projet développer avec ce Framework parfaitement modulable.

Cependant son principal atout ne réside pas là mais bien dans sa capacité à créer des API RESTfull (cf Glossaire : *REST*) exposé en HTTP(S) . En effet l’archietcture MVC s’est vite imposé comme une quasi-norme pour les API (on peut aussi trouver du SOAP(cf Glossaire : SOAP)). Avec des échanges d’objet sous le format JSON,cet environnement de travail permet de travailler avec sécurité et aisance, notamment par l’implantation d’une multitude de fonctionnalités. Il intègre d’ailleur une configuration par défaut que le développeur peut surcharger à sa guise.



*Spring boot et le Spring Framework*

JAVA est devenu depuis une quinzaine d’années un langage qui a vu son ascension perdurée, cependant les vieux défauts de ce langage ne peuvent pas être comblé et il est vrai que déployer des commandes system lancé dans un runtime, ce n’est pas ce qu’il y a de plus adaptés. Notamment si l’on veut des taches de pénétration réseau qui sont en générale des script Bash ou Python

C’est dans cet optique qu’une partie de l’application allait reposait sur un moteur en python. Après avoir eu l’aval de notre commanditaire, il nous conseilla d’utiliser la bibliothèque Hug en python qui permet d’exposer son script en http à travers des REST Controlleur comme le ferai une API.

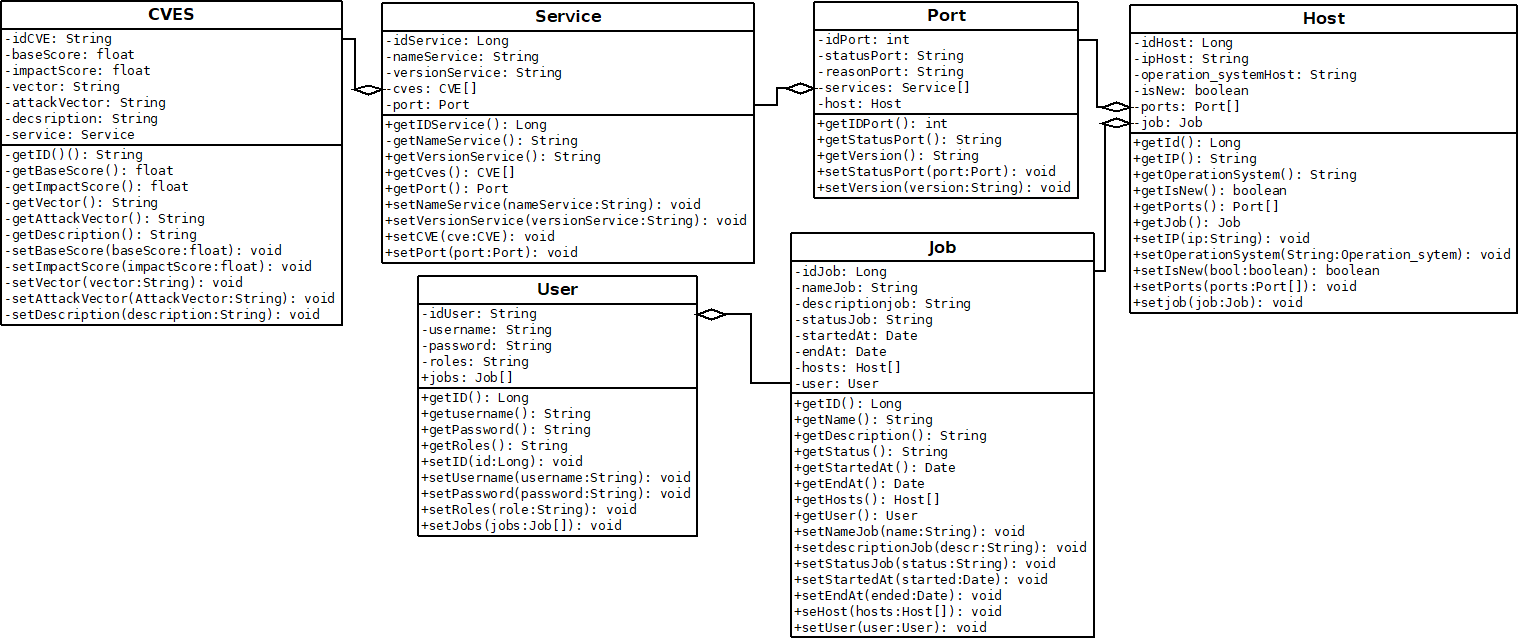
La technologie concernant le Back-End s’est décidé quasiment en même temps que celle que OffSec allait utiliser pour la persistance des données. MySQL est une des technologies pour porter les base de données les plus répandues et une des plus documentés. Il s’intègre parfaitement avec Hibernate et donc avec Springboot.

Enfin, il nous fallait un Front-End performant. Capable d’afficher les données en temps réel, portable sur tout appareil, smartphone, rasberry. Il existe de nombreux Framework Javascript extrêmement populaire : Angular, React, Amber, Vue mais nous choisirons Angular. Notamment parce qu’il est développé par Google et qu’il est donc pérenne. Ensuite parce qu’il utilise du TypeScript(cf Glossaire : ECMA-Script) qui lui a été développé par Microsoft. Le Type Script est un sur-ensemble (« superset ») de JavaScript qui permet de le Trans compiler. Cela permet de nouvelle fonctionnalités comparé au JS comme l’apparition des classes, des interfaces, du lambda calcul et du typage strict des variables.

Pour résumer voici l’architecture globale de l’application OffSec :

### 5- Représentation UML

Afin de pouvoir implémenter les fonctions nécessaires , il nous a fallu construire un diagramme UML permettant de modéliser les relations entre les différentes entités de notre applications

****

*UML de modélisation de l’application OffSec*

Notre diagramme commence avec un utilisateur (au moment du développement les rôles (utilisateur, reporter et admin) sont implémentés sous forme de tableau) identifier par ses identifiants (identifiant de l’utilisateur, son nom d’utilisateur ainsi que son mot de passe) ainsi qu’une collection de jobs. En effet, chaque utilisateur possède sa propre table de jobs pour que l’utilisateur de profils « admin » puisse avoir une liste des tâches commandes par les autres utilisateur ou reporters.

Les jobs sont des tâches préprogrammées qui ont un but offensif. Pour l’instant, il n’existe que le scanner de vulnérabilités comme job de sécurité offensif. Mais dans une optique de projet AGIL il est à noter que l’implémentation des autres tâches pourra se faire en aval du développement du prototype. Les jobs sont définis par un identifiant qui est un nombre aléatoire Long généré par Springboot. Ils ont un nom une description, une date de début, une date de fin et un statut qui permet de suivre l’évolution du job (« started », « pending »,  « cancelled », « paused »).

Afin de permettre l’évolution de l’API, il fut necessaire d’établir des tables qui peuvent êtres implémentée en fonctions de n’importe quelles taches offensives, c’est pourquoi nous avons implémentées les tables suivantes qui sont communes à toutes les tâches de sécurités offensives possibles. C’est ainsi que les tables suivantes nous ont semblés plutôt commune à toutes tâches offensives : Host, Port, Service, Cve. Host étant la table stockant les hôtes ciblés par les jobs, Port la table stockant les ports des hôtes par leur numéro de port. A la demande de notre commanditaire nous n’avons pas intégrés le type de protocole de ports utilisés. En effet nous voulions intégrer le fait de savoir si les ports tester par le scanner sont actifs sur le protocole UDP ou TCP mais en effet, les faits intéressants sont en général sur les ports TCP, c’est dans cette optique que nous décidé de ne pas l’implémenter. Dans le champ « Reason », nous avons la manière dont le port à été interrogé (en général par la méthode SYN-ACK).

Le champ Service quant à lui permet de savoir quel service était actif sur le port lorsque le port a été interrogé. Et enfin nous avons la liste des cves qui sont liés au services qui a été scanné.

# OffSec : Back-End Python

## Python

Python est un langage de programmation interprété, multi-paradigme et multiplateforme. Il permet la programmation impérative, structurée, fonctionnelles, et orienté objet avec un typage fort et bénéficie d’un ramasse-miettes et permet d’éviter l’utilisation des pointeurs du C. Python est un des langages préférés des développeurs en matières de sécurités notamment grâce à son énorme communauté et sa liste de dépendances complétement abyssale qui permet de customiser son environnement de développement et sa manipulation particulièrement aisée

Pour les besoin de notre application nous savions que nous devions avoir besoin de certaines bibliothèque :

* Json : Une bibliothèque permettant de gérer l’encodage et le décodage Json
* Requests : Une bibliothèque permettant de réaliser des requêtes http
* Sqlite3 : Une bibliothèque écrite en c permettant de gérer des petites bases de données
* Hug : Une bibliothèque permettant de porter l’application selon le modèle REST
* Os : Une bibliothèque permettant de gérer les appels systèmes
* Re : Une bibliothèque permettant de faciliter l’usage des expressions régulières
* Time : Une bibliothèque permettant la gestion du temps
* Urllib.request : une bibliothèque permettant aussi les requêtes http mais de manière différente
* Shutil : Permettant les opérations de haut-niveau sur les fichiers
* Zipfile : Une bibliothèque permettant de gérer les archives

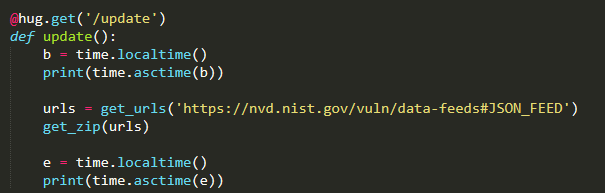
En effet, pour les besoins de notre applications nous allons utiliser Hug pour déployer cette application (ou plutôt ce module de notre application) à la façon d’une application REST, c’est-à-dire permettant l’accès à cette application en passant par des requêtes http.

Cette application renverra un objet Json et pour cela aura besoin de pouvoir effectuer une requête http sur les REST controller de l’application en Springboot

L’application devra gérer des bases de données NVD, la base de données de vulnérabilités américaines avec une méthode exposée en REST permettant de télécharger les archives les plus récentes et de mettre à jour la base de données NVD en local. Elle permet aussi de gérer la base de données Vulners du script Nmap du même nom pour permettre à l’application de toujours avoir une base de données sur laquelle travaillé

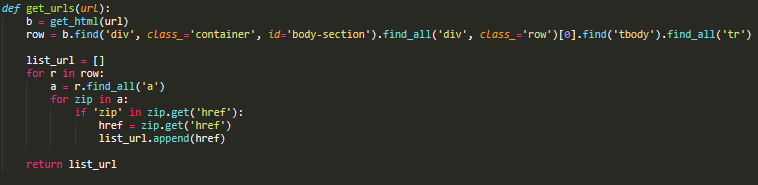
## Le script

La première méthode du script repose sur la mise à jour de la base de données en local :



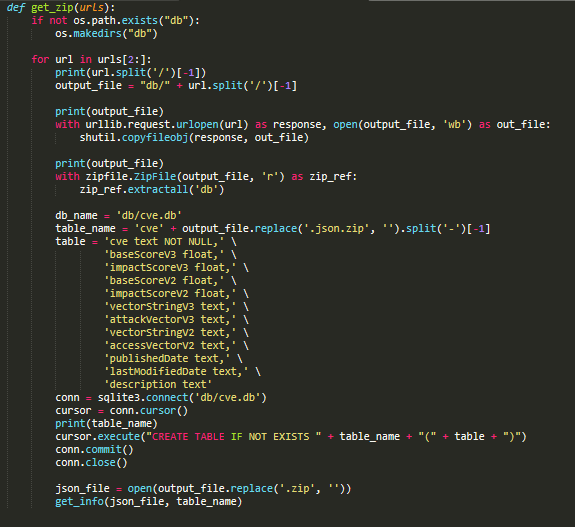
*La méthode REST pour mettre à jour la base de vulnérabilités*

Pour faire qu’une fonction Python soit exposé en REST et donc accessible en http il faut la faire précéder du décorateur ***Hug.get*** permettant de répondre au méthode GET sur ce end point.

**

*La méthode get\_urls permettant de récupérer les urls des bases de données*

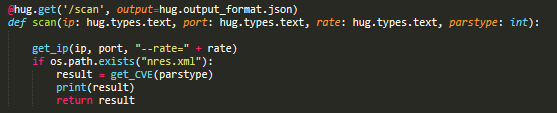
La méthode get\_Url va permettre au script de chercher les balises Href dans la page et ainsi de suivre ces href pour passer les urls à la méthode get\_zip :



*La méthode get\_zip permettant de récupérer les archives de la base de données*

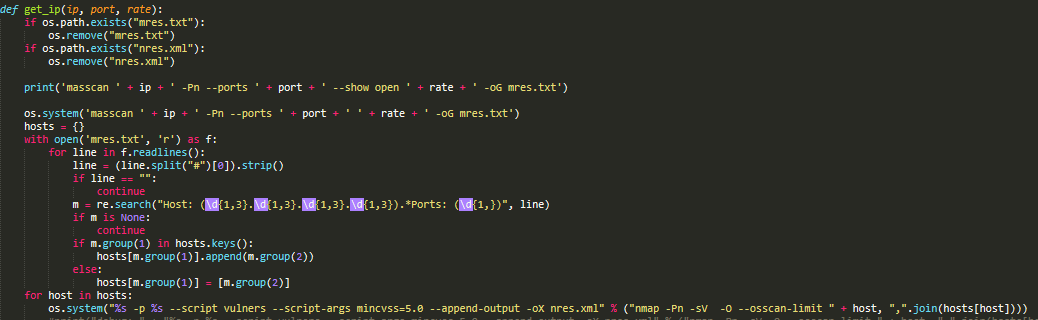
Cette fonction elle va permettre de télécharger les mise à jour de la bases de données en fonctions de celle qui sont présentes ou non sur l’ordinateur local hébergeant ce script.

Maintenant que la base de données permettant d’identifier les vulnérabilités, il nous faut une fonction permettant de lancer un scan exposé via un REST controller :



*Un REST Controller permettant le scan d’une cible*

Avec le décorateur suivant nous spécifions un end point http avec la requête GET (normalement une méthode POST serait la norme) mais pour les besoins du développement nous avons utilisés la méthode GET pour avoir les résultats visibles depuis un browser Web. Cette fonction appellera une fonction responsable du lancement des scans qui stockera les résultats dans un fichier texte. Les résultats seront eux parseur par la fonction get\_cves responsable de « décrypter » les balises de résultats pour stocker juste les résultats qui nous intéresse.

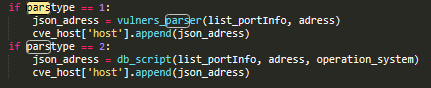


*La fonction get IP permettant de lancer MassScan et Nmap*

Cette fonction, représentative de notre implémentation globales quant à la gestion des tâches offensives, va lancer un scan de l’application Massscan. Masscan est un scanner de port extrêmement rapide et performant conçue essentiellement pour scanner de large bande d’IP et de large spectre de Port. D’après notre commanditaire,  « si aucun matériel ne réduit votre cadence vous pourriez scanner tous les ports de l’internet en 45 minutes », et c’est pour cette raison que Massscan va « dé-grossir » le travail du script suivant Nmap en prenant le fichier texte de résultat de Massscan.

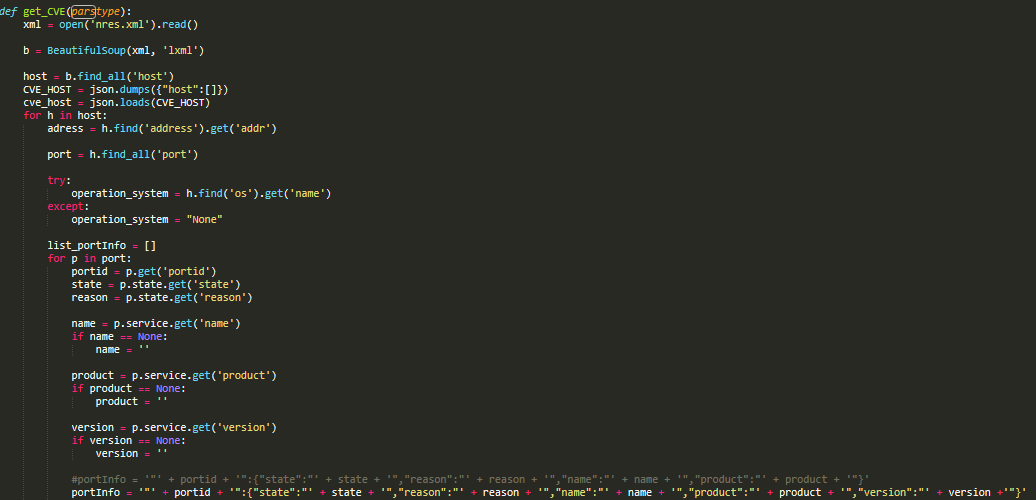
Nmap de son côtés se lancera sur les hôtes ayant été trouvés en ligne et exécutera son scan armé du script vulners qui permet de scanner les vulnérabilités sur une hôte en fonction des services qui tournent dessus

Pour récupérer les données nous avons implémenter 2 type de parsing comme dit précédemment. Un qui « scrap » le site de vulnérabilités de vulners.com et un qui consulte la base de données NVD télécharger en local. Nous préfèrerons utilisé la base de données NVD car mise à jour régulièrement.



*Le choix de la source*

Les résultats de Nmap sont ensuite stockés dans le fichier nrest.xml et passé à la fonction suivante :



*Récupération des informations de vulnérabilités sorties par Nmap*

Pour parser les résultats nous utilisons la bibliothèque très connue : BeautifulSoup, permettant d’extraire de la donnée des fichiers HTML et XML.

# OffSec : Back-End Springboot

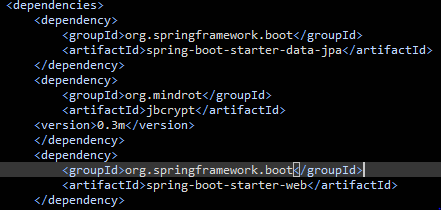
## Présentation de Springboot

Le Back-End en Springboot est le cœur de l’application. Il gère les échanges entre l’API en python et le Front-end via des objets Json sur des requêtes effectuées sur les « *end points »* des contrôleur Rest. Il est aussi le composant principal car c’est dans ce composant que la logique métier est incorporé :

* Parsage des données dans la base de données
* La logique concernant les jobs
* La mise en relation des différents composants par HTTP
* La logique entre les différentes classes
* La gestion du SGBDR
* La sécurisation de l’application
* Le contrôle d’accès
* Les droits utilisateurs
* La gestion des tests unitaire avec la dépendance Junit
* La gestion des tests d’intégrations
* La documentation de l’application via la dépendance Swagger

Les besoins de notre client stipulaient au minimum un scanner de vulnérabilités en tant que job offensif, il a donc fallu implémenter les classes remontées par l’API en Python, et la table de gestion des jobs ainsi que la classe User qui nous permettra d’utiliser certaines classes cruciales de Springboot en ce qui concerne l’authentification

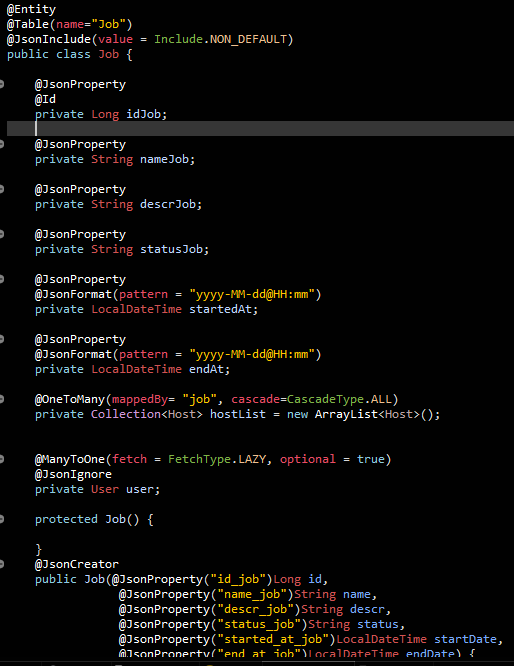
Springboot fourni un environnement de travail complet avec le gestionnaire de dépendances MAVEN lui permettant d’ajouter des modules et des fonctionnalités au Framework déjà bien complet. Son gestionnaire de dépendance Maven est un repository en ligne regroupant tous les liens à jours et sécurisés, mais pour des notions de choix Springboot s’intègre très bien avec le gestionnaire de dépendance Gradle. Les dépendances sont liées à l’environnement de travail grâce au fichier « pom.xml »



*Des dépendances dans le pom.xml*

## Création des classes

Afin de pouvoir implémenter toutes les fonctionnalités, nous devions commencer par créer les classes. Elles se construisent de la même manière que des autres classes en JAVA, à l’exception près d’annotations indiqué de cette manière « @Annotation » permettant de surcharger le comportement par défaut de Springboot :

*La création de la classe Job avec Springboot*

Les annotations permettent de décrire le comportement de Springboot quand il est confronté à une donnée. Par exemple la classe Entity permet de spécifier que cette classe sera inscrite dans la base de données alors que « JsonProperty » permettent de spécifier que ces champs sont à inscrire dans la base de données. On pourrait spécifier le champ mais Springboot prend le nom des variables comme nom de champ par défaut dans la base de données.

Une catégories d’annotations particulières est a remarqué : les annotations relationnelles « @OneToMany » et « «@ManyToOne ». Ces relations permettent de créer les liens entre les différentes entités de la même manière que les jointure SQL le font entre les différentes tables de la base de données. Le paramètre « mappedBy » permet de préciser le End-Point qui est primordiale dans les relations de type « @ManyToMany » tandis que les autres paramètres permettent à Springboot de savoir quelle comportement adoptés lords de la suppression ou lors de la création. Enfin l’annotation « @JsonCreator » permet de spécifier quel est le constructeur de l’objet.