Faculté
des Sciences
Aix*Marseille Université

ALTERFRAME

Rapport de stage de fin d'études

Sécurité, CI & développement Java

Alexandre Léonardi

M2FSIL-FSI

Année 2016/2017

Encadrants : Clément Fleury &
Idir Meziani

Enseignant : Emmanuel Godard

Table des matières

Introduction

Développement Java et développement d'une solution d'analyse statique de sécurité : ce sont les deux branches de mon stage. Il s'agit pour partie de prendre part aux contrats en Java d'Alter Frame, l'entreprise qui m'accueille pour la durée du stage, et d'autre part d'intervenir sur un projet en interne visant à mettre en place une analyse de sécurité systématique des projets Web au-travers de pratiques de CI[1][1]. La présentation d'Alter Frame sera donc naturellement la toute première partie de ce rapport.

Ce sujet a l'avantage d'être ouvert et diversifié. Il me permet d'une part de travailler sur du pur développement et d'autre part de mettre en pratique la composante sécurité de la formation FSI[2][2], tout en découvrant les concepts de CI qui m'étaient jusque là étrangers, ainsi que des technologies qui vont de pair telles que Docker. Présenter ces deux pans de mon travail à Alter Frame composera la suite du rapport.

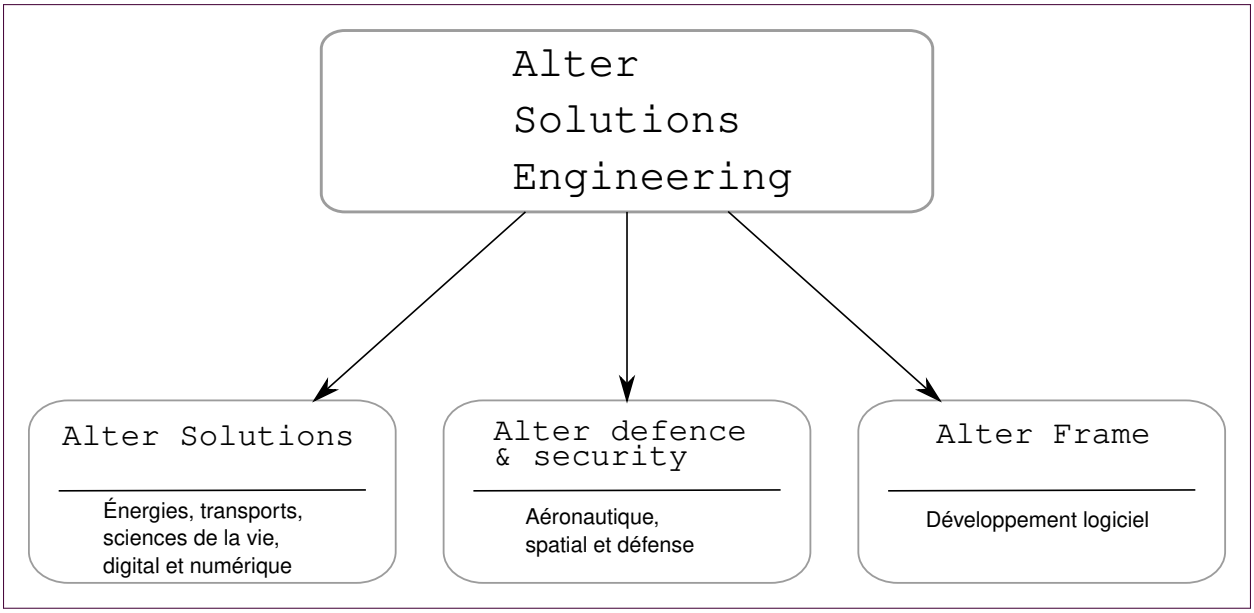
Celui-ci se clôturera en analysant et résumant les apprentissages que j'ai retiré de ce stage, et les perspectives d'avenir qu'il m'ouvre.

À noter que, par discrétion à leur égard, les noms des clients d'Alter Frame ne seront pas mentionnés et seront effacés des captures d'écran que vous trouverez dans ce document. Il en ira de même pour les différents projets et les noms de personnes physiques.

1. Continuous Integration ou intégration continue
2. Fiabilité et sécurité informatique

Rapport de synthèse

Graphique 1 – Alter Solutions Engineering et ses filiales



1 Présentation d'Alter Solutions Engineering

Alter Solutions Engineering, et plus particulièrement sa filiale Alter Frame, est l'entreprise qui m'a accueilli pour la durée de mon stage de fin d'études, nous allons donc commencer par la présenter rapidement.

1.1 Les subdivisions d'Alter Solutions Engineering et leurs secteurs d'activité

Alter Solutions Engineering est une entreprise relativement jeune : elle a été créée en 2006 et, si elle n'entre plus maintenant dans la catégorie des PME en termes de nombre de collaborateurs, elle reste une structure de petite taille.

Le siège social de l'entreprise se trouve à Versailles et c'est là où travaille l'équipe de développement française dont je fais partie. En pratique, il s'agit de l'équipe de développement d'Alter Frame qui est une entité enfant d'Alter Solutions Engineering (cf. section 1.2).

Alter Solutions Engineering est une société de conseil en hautes technologies mais en pratique, elle est composée de trois filières qui ont chacune une spécialité bien distinctes (cf. graphique 1 et graphique 2).

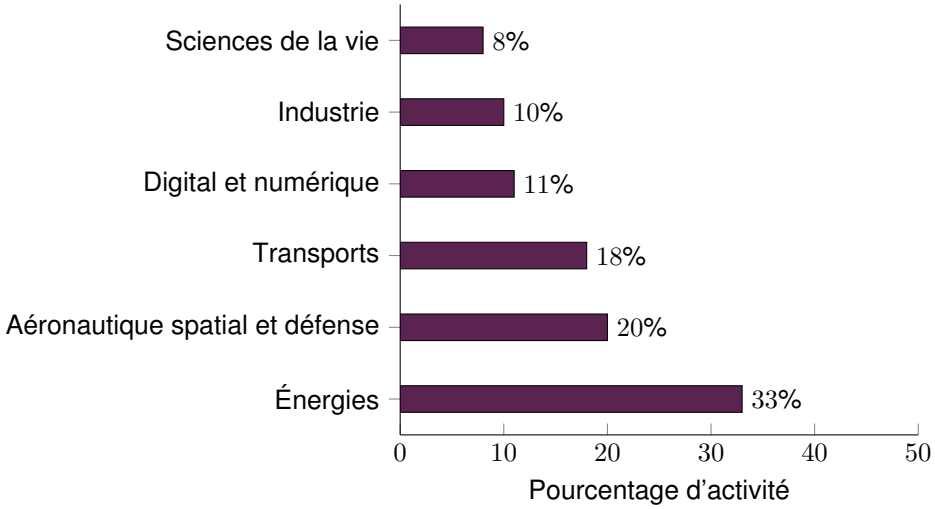
1.1.1 Alter Solutions

Cette filiale est spécialisée dans le conseil en ingénierie, notamment dans les domaines de l'énergie, des transports, des sciences de la vie, du digital et du numérique.

1.1.2 Alter defence & security

Alter defence est également orientée vers le conseil, mais cette fois plus particulièrement dans l'aéronautique, le spatial et la défense.

Alter Defence and Security est également la filiale d'Alter Solutions Engineering qui m'accueillera une fois mon stage terminé (cf. section 10). L'objectif du groupe Alter au-travers de ce stage est de me donner le



Graphique 2 – Répartition de l'activité des différentes filiales d'Alter Solutions Engineering

bagage technique et l'entraînement nécessaires pour pouvoir à terme me déployer comme expert technique auprès de clients de l'entreprise, or les missions de cyber-sécurité sont le domaine d'activité de Defence & Security.

1.1.3 Alter Frame

Alter Frame enfin est la branche spécialisée dans l'édition de logiciels et celle que j'ai rejoint durant mon stage. C'est une ESN[3][3] dont l'activité est elle-même répartie en deux catégories :

- le conseil, c'est-à-dire le fait de fournir des spécialistes d'un domaine du numérique pour la durée d'un contrat à un client ;
- le développement de logiciels au forfait, c'est-à-dire le fait de prendre commande d'un logiciel à réaliser en interne et de le livrer à la fin du contrat.

1.2 Un peu plus de détails sur Alter Frame

Bien qu'Alter Frame ait des clients et des domaines d'intervention variés, en termes de technologies il y a trois pôles de compétences qui sont caractéristiques de l'entreprise et reviennent le plus régulièrement :

- Java ;
- .NET ;
- PHP.

Bien que mon stage se soit divisé en deux grands axes, mon travail a été dans tous les cas lié au pôle de développement Java et au responsable technique sous la direction de qui j'ai travaillé. En conséquence j'ai participé à plusieurs projets Java de manière anecdotique, en plus du projet principal que je détaillerai en partie 2.

Quelques projets notables d'Alter Frame, mais sur lesquels je n'ai pas eu l'occasion de travailler :

3. Entreprise de Services du Numérique

- interface de *monitoring* de plateformes pétrolières pour tablettes ;
- plateforme web permettant d'accéder facilement à des exécutables des différents projets d'Alter Frame, à destination des commerciaux des autres branches de la compagnie ;
- outil de vérification de conformité vis-à-vis du futur règlement européen sur la protection des données[4].

2 Détail du sujet de stage

Le sujet de mon stage était ainsi formulé : « Intégration de tests de sécurité dans le processus d'intégration continue, et développement Java selon les besoins de l'entreprise. »Nous avions convenu, lors de l'entretien d'embauche, que mon travail serait également réparti entre ces deux aspects.

En pratique, cela a représenté plusieurs projets différents et une intervention en tant que consultant.

2.1 Sécurité & CI

La partie la plus précisément définie de mon stage : Alter Frame dispose d'un système d'intégration continu qui, à mon arrivée, incorporait de l'analyse qualité et la compilation du code à chaque *push* sur leur plateforme git.

Mon travail serait donc d'incorporer un aspect sécurité à la configuration déjà existante, de manière à obtenir le processus du graphique 3.

Éléments à implémenter :

- analyse dynamique, uniquement dans le cas d'application web, en automatisant des tests de sécurité avec ZAP ;
- analyse statique, si le temps le permettait, en réutilisant les analyses de code *via* Sonar déjà en place et les affinant d'un point de vue sécurité.

Ces tâches de départ dépendent de plusieurs autres qui faisaient, de fait, également partie de mon sujet de stage. Réaliser des analyses de sécurité contre les applis web d'Alter Frame impliquaient que celles-ci soient accessibles en ligne, au minimum sur un serveur privé pour les besoins du développement. Automatiser ce déploiement, et idéalement pouvoir l'étendre pour réaliser une livraison dans certaines conditions (telles qu'un *push* tagué), devrait donc être la première étape à réaliser, avant d'ensuite programmer les tests, ZAP ayant le bon goût d'être très finalement contrôlable par des APIs dans plusieurs langages[5] et en ligne de commande[6][7].

Par ailleurs, si la partie analyse statique devait se réaliser, cela impliquait d'apprendre le fonctionnement des plugins sonars et comment étendre les règles créées par défaut par l'analyseur, pour ensuite à proprement parler isoler des problématiques de sécurité qui peuvent être adressées et les faire vérifier.

2.2 Développement Java

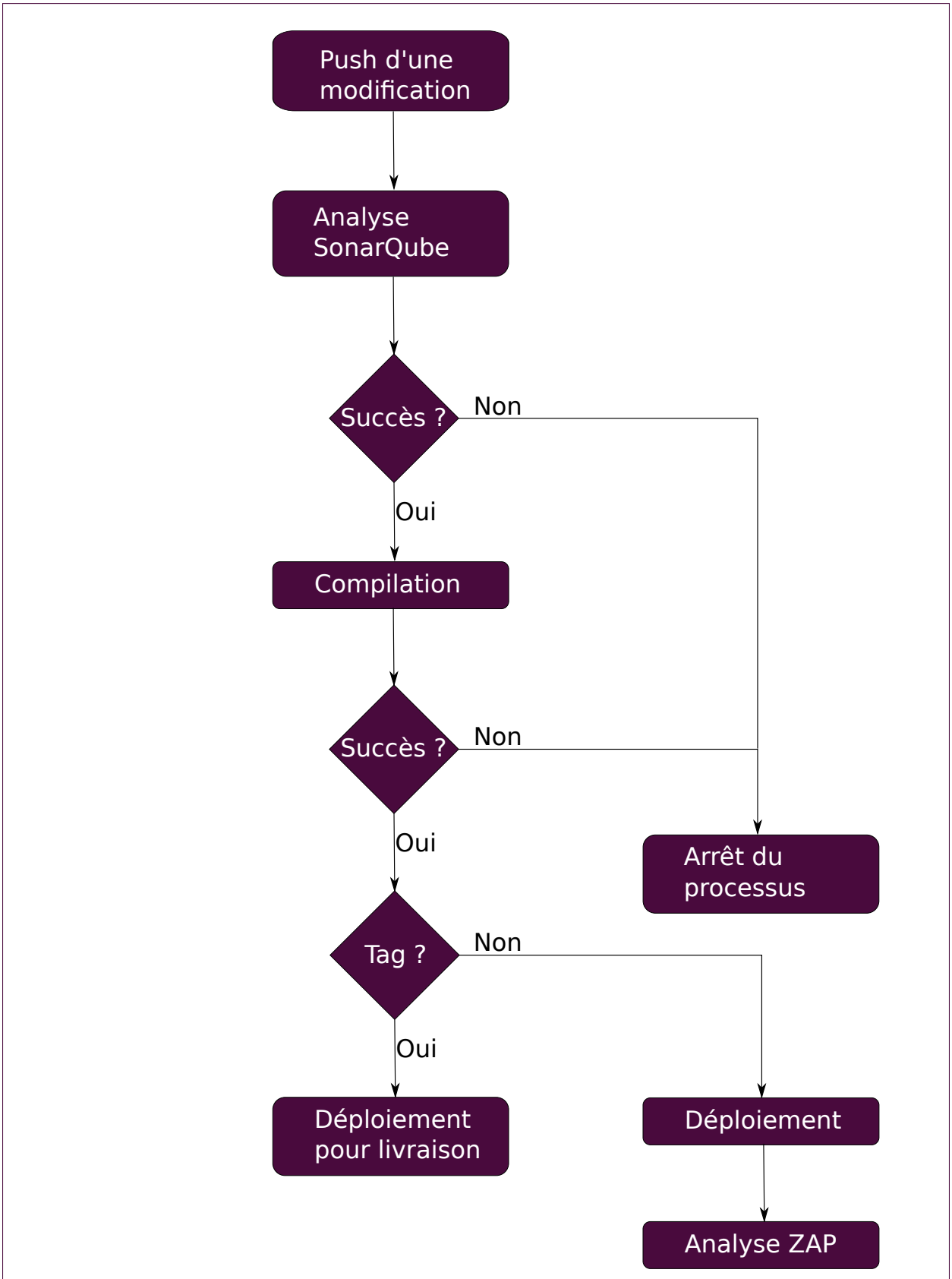
L'aspect Java, dans mon stage, était plus flou dans la mesure où il était sujet aux projets qui seraient en cours et en besoin de soutien au moment de mon arrivée dans l'entreprise. En pratique, cela a été majoritairement du développement sur un projet de gestions de tests sur des voitures, pour un constructeur automobile, ainsi que des interventions ponctuelles sur plusieurs autres projets destinés au même client.

2.3 Interventions en fonction du besoin

De même que le développement Java, cette partie de mon travail était sujette à évolution en fonction du besoin. En fin de stage, elle a pris la forme d'un audit technique pour une compagnie d'assurance qui souhaitait améliorer son service d'Intranet, tant d'un point de vue sécurité que qualité de code ou performances d'exécution.

Durant cet audit j'ai eu à gérer une partie de l'aspect sécurité de notre intervention, et une partie de l'aspect performance, cela au cours d'une mission de 3 jours chez le client.

Graphique 3 – Déroulement du processus d'intégration continue



3 Environnement de travail & solutions retenues

Je ne suis intervenu que sur des projets qui étaient déjà commencés, et de ce fait il n'y a eu que peu de choix en termes de solutions retenues. Je vais néanmoins détailler ici l'environnement de travail, les différentes solutions techniques qui étaient déjà en place à mon arrivée et avec lesquelles j'ai travaillé pendant 6 mois.

Pour autant, il est intéressant de résumer l'environnement dans lequel j'ai évolué pendant 6 mois, pour chacun des pans très variés sur lesquels je suis intervenu. Les sous-sections seront volontairement brèves ; plus de détails seront disponibles dans les parties 4, 5 et 6.

3.1 CI

3.1.1 GitLab & GitLab-CI

L'intégration continue dans les projets d'Alter Frame se fait à l'aide d'un service proposé par la plateforme d'hébergement de projets informatiques GitLab[8]. Le service en question, GitLab-CI[9], propose de mettre en place de l'intégration continue sur les projets hébergés sur GitLab.

Au moment de mon arrivée chez Alter Frame la partie CI des projets consistait majoritairement en la compilation des projets et une analyse de code à l'aide d'un plugin SonarQube[10], mise en place depuis environ 2 ans.

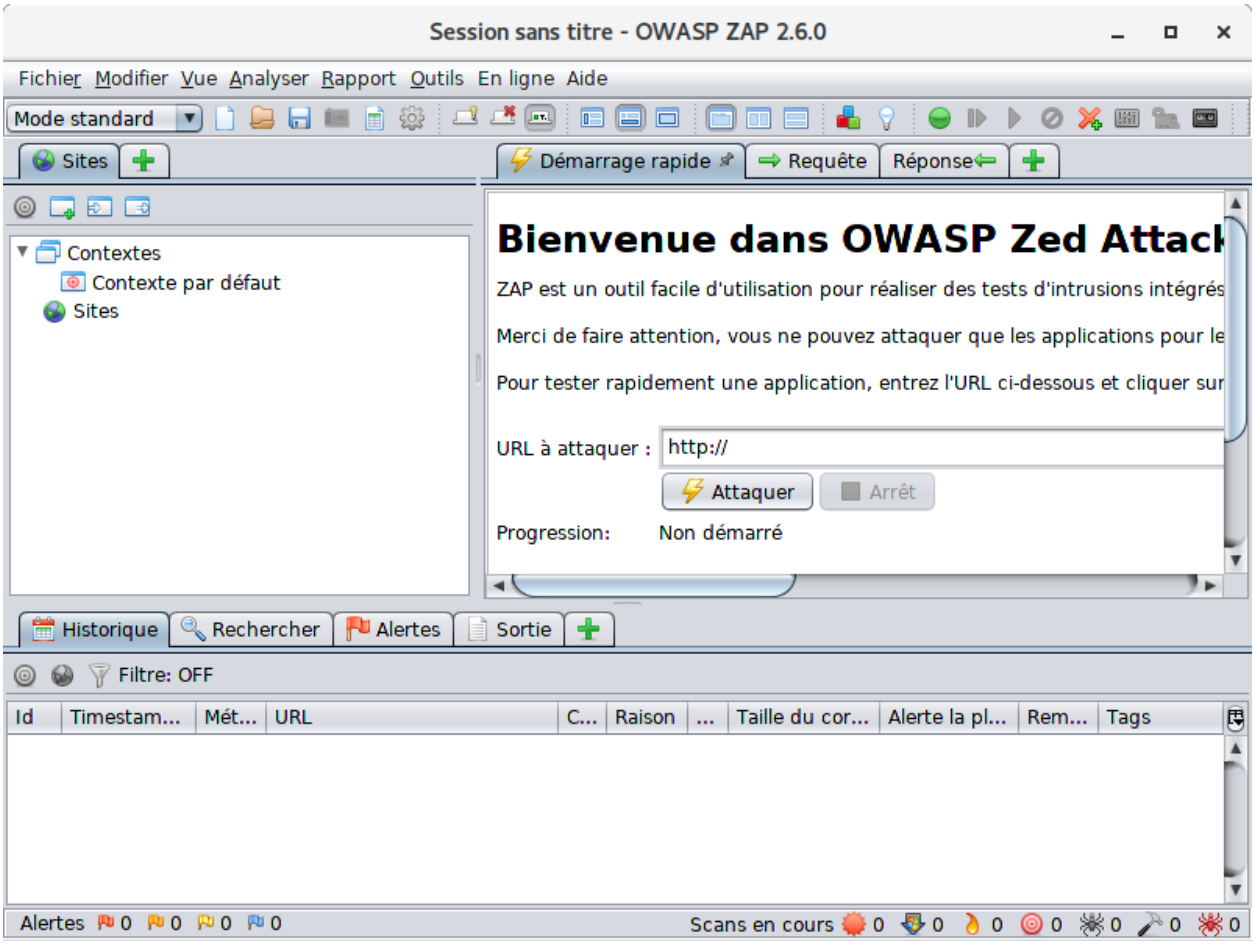
Le principe est que les actions décrites ci-dessus, compilation et analyse de code, sont effectuées à chaque push sur le serveur GitLab. Ce fonctionnement peut ensuite être affiné, pour ne se produire que lorsqu'un tag git est pushé ou sur certaines branches (branche master, tag de release, etc).

Il n'y avait néanmoins pas de composante cybersécurité dans le processus de CI d'Alter Frame et c'est donc ce sur quoi je suis intervenu en priorité. Néanmoins, mon travail ne s'est pas limité à cela et j'ai aussi pu intervenir sur d'autres aspects du CI et améliorer l'existant.

3.1.2 ZAP : Zed Attack Proxy

ZAP[11] (voir figure 4) est un projet open source développé par l'OWASP[12]. Il s'agit un proxy qui peut intercepter et analyse le trafic qui traverse la machine hôte. ZAP est un outil de sécurité très intéressant et ce pour un grand nombre de raisons :

- activement développé[13] ;
- open source et cross-platform ;
- OWASP est une référence dans le monde de la sécurité ;
- une large communauté, et donc une grande quantité de ressources sur laquelle s'appuyer ;
- ZAP est contrôlable en ligne de commande (voir extrait 1) et *via* des APIs en plusieurs langages.



Graphique 4 – Fenêtre de démarrage de ZAP

```
$ zap.sh -cmd -help
Usage:
    zap.sh [Options]
Core options:
    -version                  Reports the ZAP version
    -cmd                      Run inline (exits when command line options complete)
    -daemon                   Starts ZAP in daemon mode, ie without a UI
    -config <kvpair>          Overrides the specified key=value pair in the configuration
    file
    -configfile <path>        Overrides the key=value pairs with those in the specified
    properties file
    -dir <dir>                Uses the specified directory instead of the default one
    -installdir <dir>         Overrides the code that detects where ZAP has been installed
    with the specified directory
    -h                        Shows all of the command line options available, including
    those added by add-ons
    -help                     The same as -h
    -newsession <path>        Creates a new session at the given location
    -session <path>           Opens the given session after starting ZAP
    -host <host>              Overrides the host used for proxying specified in the
    configuration file
    -port <port>              Overrides the port used for proxying specified in the
    configuration file
    -lowmem                   Use the database instead of memory as much as possible - this
    is still experimental
    -experimentaldb           Use the experimental generic database code, which is not
    surprisingly also still experimental
    -nostdout                 Disables the default logging through standard output
Add-on options:
    -script <script>          Run the specified script from commandline or load in GUI
    -addoninstall <addon>     Install the specified add-on from the ZAP Marketplace
    -addoninstallall          Install all available add-ons from the ZAP Marketplace
    -addonuninstall <addon>   Uninstall the specified add-on
    -addonupdate              Update all changed add-ons from the ZAP Marketplace
    -addonlist                List all of the installed add-ons
    -quickurl [target url]: The URL to attack, eg http://www.example.com
    -quickout [output filename]: The file to write the XML results to
    -quickprogress: Display progress bars while scanning
    -last_scan_report <path> Generate the 'Last Scan Report' into the specified path
```

Extrait 1 – Options de ZAP en ligne de commande

Je n'avais, avant mon stage, que brièvement eu l'occasion d'utiliser ZAP, au-travers du sous-projet de tests d'intrusion avec M. Pachy. Pouvoir m'entraîner plus longuement avec représentait donc à la fois un intérêt personnel, car cela me permettait d'en apprendre plus sur les vulnérabilités web les plus répandues, et professionnel car c'est un outil dont l'usage pourrait être pertinent pour mes futurs emplois.

ZAP est le seul outil sur lequel il y a vraiment eu un choix à faire car les tests de sécurité n'étaient pas encore implémentés à mon arrivée. Le principal concurrent de ZAP est Burp Suite[14], une solution non-libre mais qui dispose d'une version gratuite.

Les arguments qui ont fait pencher la balance en la faveur de ZAP sont :

- le fait que l'OWASP est une référence dans le monde de la sécurité ;
- le développement ouvert qui est une assurance de qualité dans le monde de la sécurité (possibilité de relever les failles/oublis/erreurs dans le code) ;
- le fait que moi comme mon tuteur ayons déjà eu une expérience avec ZAP et pas avec Burp.

3.1.3 Docker

Docker[15] est une technologie de virtualisation basée sur des conteneurs, qui vient se place en opposition aux hyperviseurs et machines virtuelles [4]. En plus d'une charte graphique à base de faune marine des plus plaisantes [5], la technologie Docker présente plusieurs fonctionnalités qui la rendent intéressante dans le monde de l'industrie informatique :

- un conteneur est plus léger qu'une VM ;
- un conteneur s'exécute de la même façon sur n'importe quelle machine où Docker est installé ;
- un conteneur peut embarquer toute la configuration nécessaire au bon fonctionnement de l'application, et c'est là le point le plus important. L'étape de configuration de l'environnement n'a à être effectuée qu'une seule fois, à la création de l'image [6]. De plus le système de Docker Store[16], proche de celui d'un gestionnaire de paquets, permet au client d'avoir facilement la dernière version possible d'un logiciel, encore une fois en s'abstrayant des changements de configuration qui vont avec la mise-à-jour.

On assiste donc à une généralisation de l'utilisation de Docker depuis sa première version en 2013, avec de nombreux cas d'utilisation[17], mais aussi à une multiplication des outils en lien avec la technologie Docker comme des outils de gestion de groupes de containers[18][19].

GitLab-CI est étroitement lié à Docker : lors d'un push, un container est lancé dans lequel tout le processus de CI est exécuté, en isolation. De ce fait, il n'y avait pas de choix à faire quant à la technologie de virtualisation. Le comportement du processus peut être configuré au-travers d'un script en YAML, il est par exemple possible de sélectionner l'image Docker servant d'environnement d'exécution.

3.1.4 YAML

YAML Ain't Markup Language[20], de son nom complet, est un « standard de sérialisation de données ». L'objectif de ce langage est de permettre de représenter des donnes à la fois clairement et simplement, principalement en les formatant comme des listes ou des maps.

La syntaxe de YAML est donc sans surprise concise et facile de prise en main[21] ; qui plus est le code YAML est assez proche de l'anglais pour être compréhensible même par quelqu'un qui n'y est pas familier.

Dans le cas qui nous occupe, YAML est le langage permettant de contrôler le processus de CI proposé par GitLab-CI grâce à un script nommé `.gitlab-ci.yml` placé à la racine du projet (l'extrait 2 est un exemple écourté de script YAML sur lequel j'ai travaillé).

4. Ou VMs pour Virtual Machines
5. `https://www.docker.com/sites/default/files/group_5622_0.png`
6. On ne parle de conteneur qu'une fois l'image en cours d'exécution, cf. différence entre processus et programme

```yaml
variables:
    MYSQL_DATABASE: database_name

services:
    - mysql:latest
    - redis:latest

deploy:
    stage: build
    image: aleonardi/symfony-mysqlclient
    stage: build
    script:
        - bash .gitlab-ci.sh
        - chmod a+x vendor/bin/phpunit
        - php vendor/bin/phpunit --colors

zap-docker:
    stage: test
    image: owasp/zap2docker-weekly:latest
    script:
        - python zap-baseline.py -t http://rick.alter-frame.fr/ -c zap.conf
    artifacts:
        - ./zap-report
    only:
        - master
        - tags

```

Extrait 2 – Script de contrôle de processus de CI en YAML

3.2 Développement Java

Il y a moins à dire sur le développement Java car il s'agit d'un pan de travail très proche de ce que j'ai déjà rencontré par le passé, que ce soit dans des cours ou dans mon précédent stage. Chez Alter Frame, j'ai développé en Java 7 couplé à Swing pour la GUI, le tout dans un environnement Windows en utilisant les classiques Maven et git.

Encore une fois, le projet ne m'ayant pas attendu pour en premier lieu, il n'y avait pas de grande marge de manoeuvre quand à quelles technologies utiliser. L'environnement Windows est indépendant de Java à proprement parler, mais dû à l'utilisation de VBScript pour créer et modifier des classeurs Microsoft Excel.

L'incompatibilité entre le projet et Java 8 reste, elle, un mystère [7].

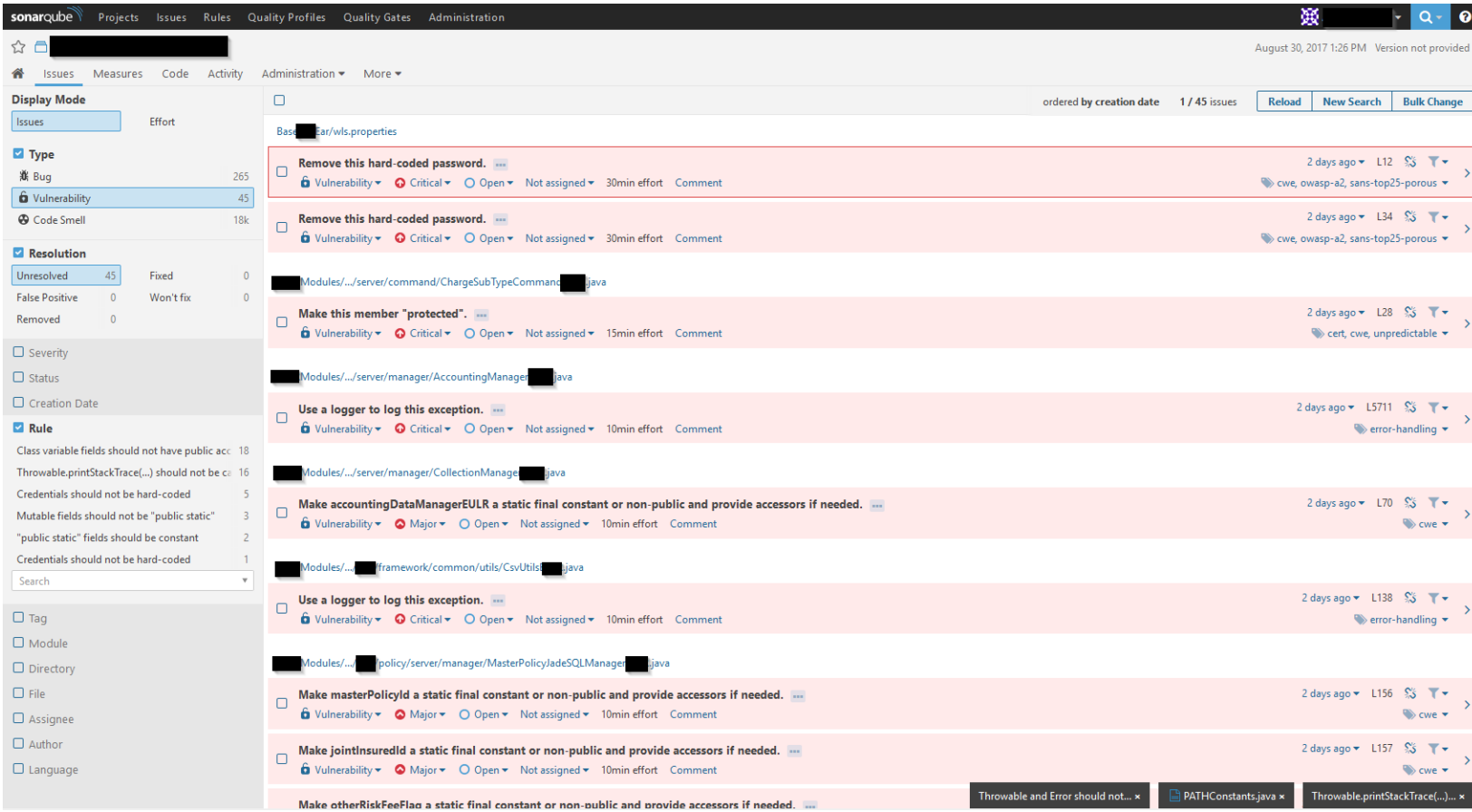
3.3 Audit technique

3.3.1 ZAP : Zed Attack Proxy

J'ai à nouveau eu l'occasion d'utiliser ZAP pendant le déroulement de l'audit. Cette fois-ci il ne s'agissait pas d'en automatiser l'usage et donc de le contrôler au-travers d'une API, mais bien d'un cas d'usage plus classique où j'ai configuré ZAP comme proxy Internet, et observé le trafic lors de l'utilisation de l'application auditée grâce à son interface graphique.

Néanmoins, les avantages de ZAP qui ont fait que mon tuteur et moi l'avons retenu pour l'intégration continue s'appliquent toujours ici (excepté pour le contrôle par API/ligne de commande) et le raison du choix de ZAP est donc la même.

7. https://i.pinimg.com/564x/66/ba/a0/66baa08b16a192d752959fa4c29bc96a.jpg



Graphique 5 – Extrait des résultats de sécurité du scanner Sonar

3.3.2 SonarQube

SonarQube est un outil d'analyse de code bien connu. Bien que sa vocation première soit d'améliorer la qualité et la maintenabilité du code analysé, Sonar incorpore aussi des règles de sécurité dans ses patrons de détection. C'est dans cette optique que je l'ai utilisé (la figure 5 liste une partie des vulnérabilités que sait détecter Sonar).

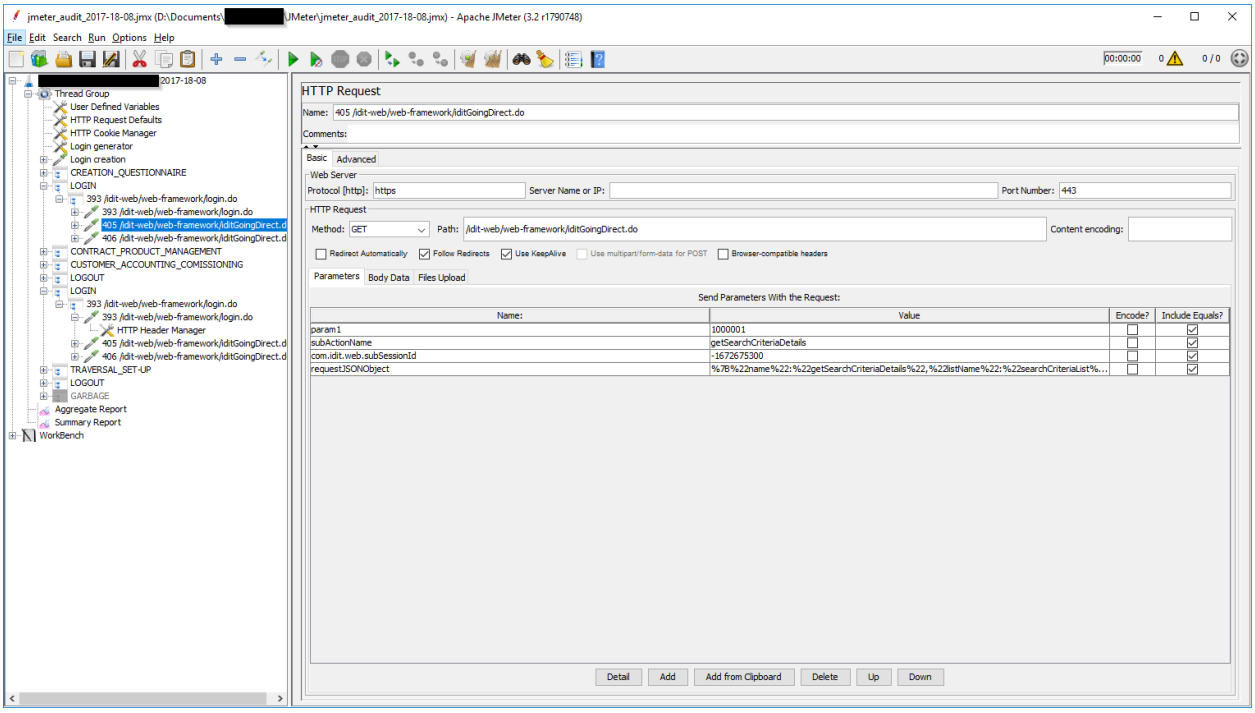
C'est durant l'audit que j'ai majoritairement utilisé Sonar, mais les scripts de CI qui étaient en plus avant mon arrivée chez Alter Frame incorporaient déjà des analyses Sonar effectuées sur le code. C'est d'ailleurs de ce fait (mon tuteur ayant déjà de l'expérience avec ce logiciel) ainsi que du faible nombre de concurrents gratuits[22] que nous avons choisi d'utiliser Sonar dans ce cas de figure.

3.3.3 JMeter & SoapUI

Ces deux outils, de même que ZAP, avaient déjà été utilisés lors des itérations précédentes de l'audit, et les conserver permettait de comparer facilement les résultats que nous obtiendrions avec les résultats antérieurs. Aussi, en l'absence de raison de ne *pas* les conserver, nous les avons conservés.

SoapUI[23] est un outil de test pour applications REST ou SOAP. L'application auditée utilisait le protocole SOAP, et SoapUI nous a servi à modifier et rejouer des requêtes isolées, sans tester les performances de l'application mais pour en saisir le fonctionnement et préparer le véritable test, en plus de vérifier que nous avions bien les droits pour réaliser toutes les opérations prévues.

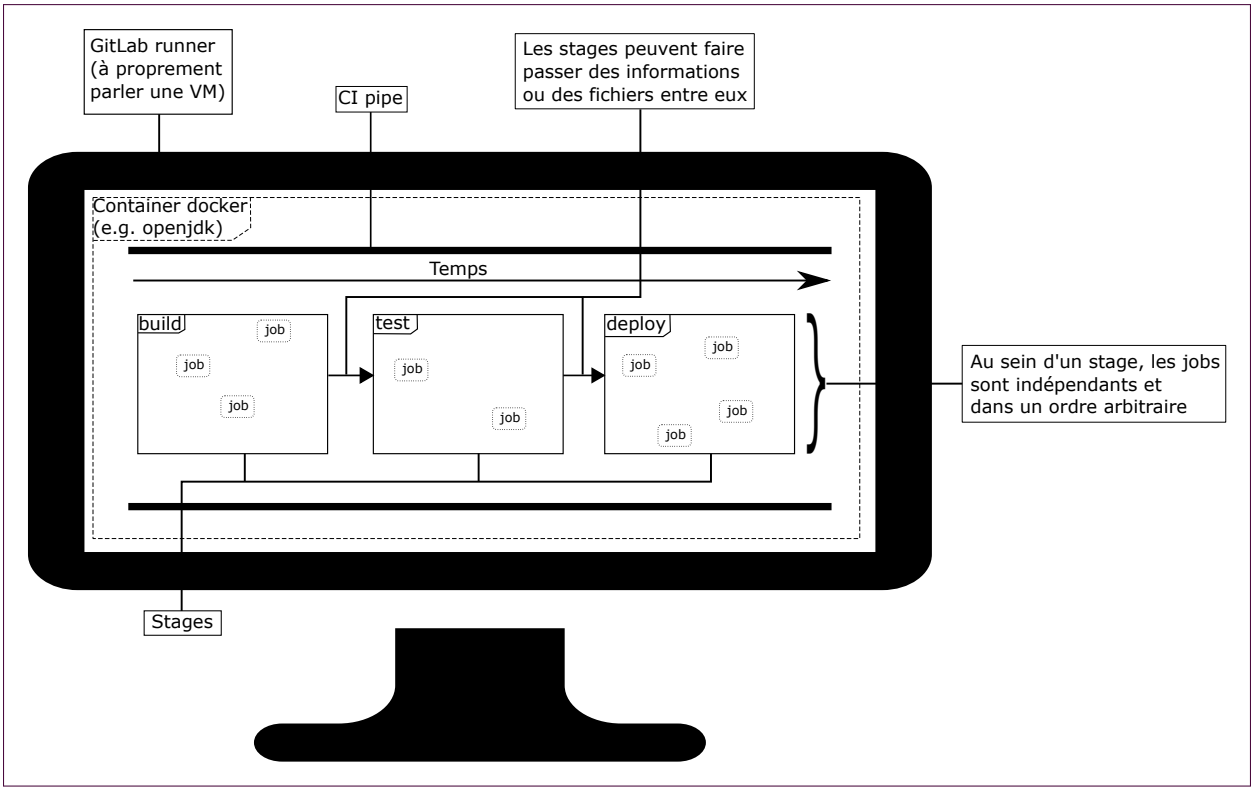
JMeter est un outil de test de performances pour applications web en général. Le panel de fonctionnalités



Graphique 6 – JMeter, une fois configuré

qu'il offre est extrêmement large (et en conséquence, sa prise en main est loin d'être triviale) et nous n'en avons utilisé qu'une partie : configurer JMeter comme proxy pour enregistrer un cas d'utilisation typique qui servira de scénario de test (on peut voir les différentes étapes d'un tel scénario dans l'interface de JMeter dans la figure 6), puis rejouer celui-ci en boucle et plusieurs fois en parallèle pour tester les limites de l'application.

Graphique 7 – Résumé du processus de CI (*build*, *test* et *deploy* sont les stages par défaut)



4 Synthèse du travail d'intégration continue

L'amélioration du CI d'Alter-Frame et l'ajout de la composante sécurité a été la tâche principale de mon stage.

Un point sur le fonctionnement et la nomenclature de GitLab-CI[24] : le script de contrôle du processus de CI est le `.gitlab-ci.yml`, ou le `.yml` pour faire court. Le comportement par défaut pour ce script est de définir une image Docker, avec la balise `image` (on aurait par exemple `image: openjdk:latest` pour aller chercher l'image openjdk sur le hub docker, et récupérer la version taguée "latest").

Cette image va être instanciée en un container au début de l'exécution du script, et ce dernier se déroulera dans l'environnement du container. Utiliser l'image openjdk par exemple donne accès à une JDK et permet donc d'appeler des fonctions telles que javac ou d'installer des utilitaires qui en dépendent tels que SonarQube.

On appelle "runner" l'hôte sur lequel s'exécute l'image docker, et "job" chacune des fonctions en lesquelles le script peut être séparé. Eux-mêmes peuvent être regroupés en stages : les stages s'exécutent dans l'ordre dans lequel ils sont déclarés et si un stage échoue, le runner s'arrête. En revanche, l'ordre des jobs au sein d'un stage est à la discrétion du runner et ne peut pas être connu.

La figure 3 résume le fonctionnement de GitLab-CI.

4.1 Déployer automatiquement les applications web

Mon intervention sur cette partie a été en commun avec un autre ingénieur d'Alter Frame. Nous devions faire en sorte de compiler les applications web en PHP, et générer à partir de là une image Docker conte-

nant l'application et toute la configuration requise, puis déployer cette image sur un serveur appartenant à Alter Frame.

Voilà pour la théorie. La pratique a été une succession d'approches différentes qui n'ont pas toujours été fructueuses, et beaucoup d'essai et échec permettant d'avancer petit à petit ; il faut savoir que la documentation de GitLab-CI sur le sujet[25] n'est pas parfaitement complète et fait la supposition que l'utilisateur possède une bonne connaissance de Docker.

La première approche fut d'exécuter le runner en mode shell plutôt que Docker. C'est la méthode la plus facile mais aussi celle qui s'éloigne le plus du comportement par défaut et qui fait perdre la grande flexibilité qu'offre le fonctionnement par image Docker : les différents outils doivent être définitivement installés sur le serveur qui héberge les runners.

En plus de ce défaut, cette méthode est discutable d'un point de vue sécurité car elle exige que l'utilisateur qui exécute le script ait des privilèges administrateur, donc indirectement toute personne qui intervient sur les scripts .yml[26].

Au final et malgré ses points négatifs, cette méthode nous a permis d'arriver à nos fins, mais nous ne l'avons pas retenue, pour partie pour les raisons exposées plus haut et pour partie pour des raisons propres à la compilation de l'application en PHP qui bloquaient l'ingénieur avec lequel j'ai travaillé.

Deuxième approche, plus en accord avec la philosophie de GitLab-CI : utiliser docker-in-docker (dind). L'idée est simple et la mise en oeuvre plus complexe (pour changer). Il s'agit de fournir au runner une image disposant des utilitaires nécessaires pour construire une image docker, et finalement de réaliser le même processus qu'en mode shell, mais dans le contexte isolé du container docker parent.

Les inconvénients de la méthode shell disparaissent : plus besoin d'installer en dur sur le serveur des utilitaires qui sont spécifiques à un projet (le serveur héberge plusieurs runners qui se répartissent tous les projets d'Alter Frame selon les besoins) ; plus besoin non plus d'avoir des privilièges administrateur sur le serveur.

Bien entendu cette méthode aussi avait un coût. Tout d'abord, GitLab-CI met en place des protections pour éviter que tout développeur puisse, par défaut, avoir accès à ces fonctionnalités et il faut donc une étape de configuration supplémentaire au niveau des runners pour utiliser dind.

Ensuite, déployer et configurer une application est un procédé lourd qui peut impliquer l'installation de dépendances. En réalisant cela dans un container, deux cas de figure se présentent :

- soit on utilise une image générique proposant l'outil de base (e.g. openjdk :latest pour un projet Java) et on installe dans le .yml les différentes dépendance. Ce n'est pas compliqué de mise en oeuvre, mais chaque exécution du script sera longue, et la lenteur est vite limitante dans le domaine de l'intégration continue (le développeur n'a pas toujours le temps d'attendre qu'un procédé long se termine) ;
- soit on crée une image personnalisée qui embarque déjà les dépendances requises, cela accélère le processus de CI mais rajoute du travail en amont avec la création et la configuration de l'image, ainsi que son stockage sur un registre Docker. De plus, le processus perd en transparence car la configuration de l'environnement devient cachée au développeur qui ne voit que le nom de l'image utilisée. Bien sûr, cela peut aussi être vu comme un avantage car le .yml s'en trouve d'autant allégé, et ne contient au final plus que l'essentiel du point de vue intégration et déploiement.

```
1  FROM nouchka/symfony:7.0
2  LABEL maintainer="aleonardi@alter-frame.com"
3
4  ARG mysql_apt_version
5
6  ENV DEBIAN_FRONTEND=noninteractive
7  ENV DEBCONF_NONINTERACTIVE_SEEN=true
8
9  RUN apt-get update -yqq
10 RUN apt-get install --assume-yes wget lsb-release gnupg
11 RUN wget https://dev.mysql.com/get/mysql-apt-config_${mysql_apt_version}_all.deb
12 RUN dpkg -i mysql-apt-config_${mysql_apt_version}_all.deb
13 RUN apt-get update -yqq
14 RUN apt-get install --assume-yes mysql-client
```

Extrait 3 – Dockerfile utilisé pour le job de déploiement de l'application

C'est la deuxième approche que j'ai conservée (voir le code de l'image en question dans le listing 3), mais non sans avoir testé la première au préalable. Avoir la configuration directement dans le .yml était source de complexité (appeler un .sh à l'intérieur du .yml pour externaliser de gros blocs de code) et de bugs (de petits changements pouvaient entraîner des résultats inattendus, surtout compte tenu du fait que nous étions plusieurs à intervenir sur le .yml).

Au final, cette méthode a tenu ses promesses : nous nous retrouvions avec une configuration sauvegardée à part, facile et rapide à importer, et qui comportait toutes les dépendances requises pour le projet. De plus un runner avait été spécialement configuré pour permettre l'utilisation de dind, restreinte par défaut, et générer l'image qui encapsulait l'appli web développée par Alter Frame devenait possible.

4.2 Intégrer ZAP et les analyses de sécurité

ZAP a pour lui d'intégrer par défaut les cas d'usage qui m'importaient, à savoir de pouvoir être utilisé en mode ligne de commade et/ou contrôlé par une API sans interaction avec l'utilisateur au moment de l'exécution.

De ces deux solutions, c'est celle de l'API qui est la plus complète : ZAP en ligne de commande propose quelques options (cf. listing 1) mais qui sont vite limitées : se connecter à un site web, lancer certaines des commandes de base de l'application comme le crawler ou le scan actif, et générer un fichier de résultats. Le scan actif de ZAP est tout de même assez intéressant pour que cette méthode soit pertinente, mais les options proposées par les APIs sont bien plus foisonnantes.

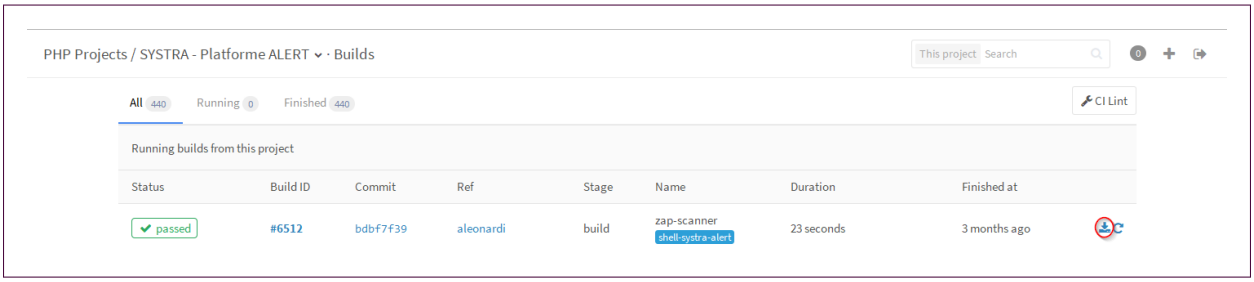
Néanmoins, la solution de l'API rajoute une contrainte : en plus de ZAP, il faut que l'environnement de CI dispose du langage de l'API. Les deux APIs principales maintenues par la communauté de développeurs "officiels" de ZAP sont celles en Java et en Python. J'ai retenu l'utilisation de Python d'une part car ce langage est concis et se prête très bien à l'écriture de courts scripts, d'autre part pour avoir l'occasion de m'en servir et développer des compétences qui me seraient utiles plus tard dans mon parcours professionnel (ce qui sera le cas dès octobre, cf section 10).

Plutôt que de rajouter encore un élément à l'image Docker créée spécialement au point précédent, j'ai préféré utiliser une fonctionnalité très intéressante de GitLab-CI, celle de pouvoir spécifier une image différente pour chaque job. Elle rajoute en verbosité au script car l'image doit donc être spécifiée pour chaque job sans exception (impossible d'en utiliser une par défaut et de ne la remplacer que pour le job de l'analyse ZAP), mais elle évite de surcharger l'image Docker utilisée pour le déploiement. Qui plus est, cette image de déploiement est spécifique à chaque projet et doit donc être réécrire, alors que le script commandant l'exécution de ZAP peut être générique (exception faite de l'URL cible). Inclure l'image à utiliser à ce script renforce cette autonomie, le bloc de code peut littéralement être copié collé d'un projet à l'autre et fonctionner (encore une fois sous réserve de changer l'URL à attaquer).

Nous nous retrouvons avec une configuration versatile : elle peut être exportée facilement et profite de toute l'expressivité de l'API Python pour ZAP. L'OWASP propose des images Docker de ZAP[27][28] mais elles ne correspondent pas à mon besoin, elles exposent un script nommé zap-scanner qui permet, comme son nom l'indique, de lancer un scanner ZAP en ligne de commande. Pour une utilisation dans le cadre de GitLab-CI, on cherche plutôt des images qui exposent /bin/sh, car cela signifie qu'une fois le container lancé on à accès à /bin/sh et on peut donc utiliser la ligne de commande.

Bien sûr il existe des images non officielles de ZAP[29] mais qui, à chaque fois, ont été créées avec un usage précis en tête plutôt que dans un but de généricité ou n'étaient plus maintenues et ne correspondaient donc, encore une fois, pas à mon besoin. La conséquence a été que j'ai utilisé une image générique qui ne contient que Python et une installation Linux basique[8], et j'ai installé *via* le .yml ZAP. Le script Python pour contrôler l'attaque était inclus dans le dépôt git, et il ne restait plus qu'à l'appeler en spécifiant, en paramètre, l'URL à attaquer.

Graphique 8 – Récapitulatif du *build* et icône de téléchargement du rapport ZAP



La dernière étape était de générer, et rendre disponible, un rapport pour que les développeurs puissent prendre des mesures. L'idéal aurait été d'envoyer automatiquement un mail à l'auteur du commit et qui contiendrait le rapport en XML, néanmoins le temps ne m'a pas permis de me pencher sur la faisabilité de cette fonctionnalité. Ce que j'ai fait en revanche est de créer un artefact, c'est-à-dire de marquer un fichier généré pendant le job comme persistant pour que GitLab le sauvegarder sur le serveur qui héberge les runners et le rende téléchargeable dans son interface web. Un example de la syntaxe utilisée est visible dans le listing 4.

```
1   zap-scanner:
2   stage: test
3   script:
4     - # ...
5       # The output file has to be specified here, otherwise it will be printed on standard
     output
6     - docker exec zap-sh zap-cli -p 8090 active-scan 'http://itsecgames.com/' > zap-report
    .xml
7     - # ...
8   artifacts:
9     paths:
10      # The path to the report has to be specified here to be downloadable
11    - ./zap-report.xml
```

Extrait 4 – Extrait de code qui lance un scanner ZAP en ligne de commande et rend le rapport disponible sur GitLab

Dans l'interface web de GitLab, le détail des résultats des builds sont visibles dans une page à part où le rapport peut-être téléchargé (l'icône entourée en rouge dans la figure 8).

8. à savoir python :3.3.6-alpine3.4, qui a l'avantage d'être plus légère que python :latest qui utilise Debian.

4.3 Amélioration de l'existant

Les deux points précédents étaient des ajouts et, de ce fait, la part la plus visible de mon travail sur le processus de CI, mais une partie en était déjà établie à mon arrivée et j'ai aussi travaillé sur cette partie, pour l'améliorer : principalement du point de vue de la lisibilité et de la maintenabilité.

On peut résumer mon intervention en trois points importants :

- regrouper le code dupliqué et utiliser des ancres[30] et des templates (cf. listing 5) ;
- utiliser des images Docker pré-configurées plutôt que télécharger et installer un logiciel, autant que possible ;
- stocker les valeurs variables (numéro de version, identifiants de connexion, etc) dans des variables (précisément). À noter que GitLab-CI offre une mécanique de variables secrètes[31].

```
1  ## Template for building code
2  .build_template: &maven_clean_install
3    script:
4      ## Install maven
5      - wget -q http://wwwftp.ciril.fr/pub/apache/maven/maven-${MAVEN_MAJOR_VERSION}/${
   MAVEN_FULL_VERSION}/binaries/apache-maven-${MAVEN_FULL_VERSION}-bin.zip
6      - unzip -qq apache-maven-${MAVEN_FULL_VERSION}-bin.zip
7      - rm apache-maven-${MAVEN_FULL_VERSION}-bin.zip
8
9      ## Install and populate database
10     - export
11     - apt-get update && apt-get --assume-yes install mysql-client
12     - mysql --user=$MYSQL_ROOT_USERNAME --password="$MYSQL_ROOT_PASSWORD" --host=mysql "
   $MYSQL_DATABASE" < ./server/sql/db-structure.sql
13
14     ## Build application
15     - ./apache-maven-${MAVEN_FULL_VERSION}/bin/mvn -f ./root/pom.xml clean install
16
17     mvn:
18     stage: build
19     ## Importing services
20     services: *mysql
21     ## Merging anchored code with current job
22     <<: *maven_clean_install
23     only:
24       - develop
```

Extrait 5 – Template d'installation et utilisation de Maven, et son appel dans un job

5 Synthèse du développement Java

5.1 Méthodologie

5.2 Travail effectué

6 Synthèse de l'audit

6.1 Méthodologie

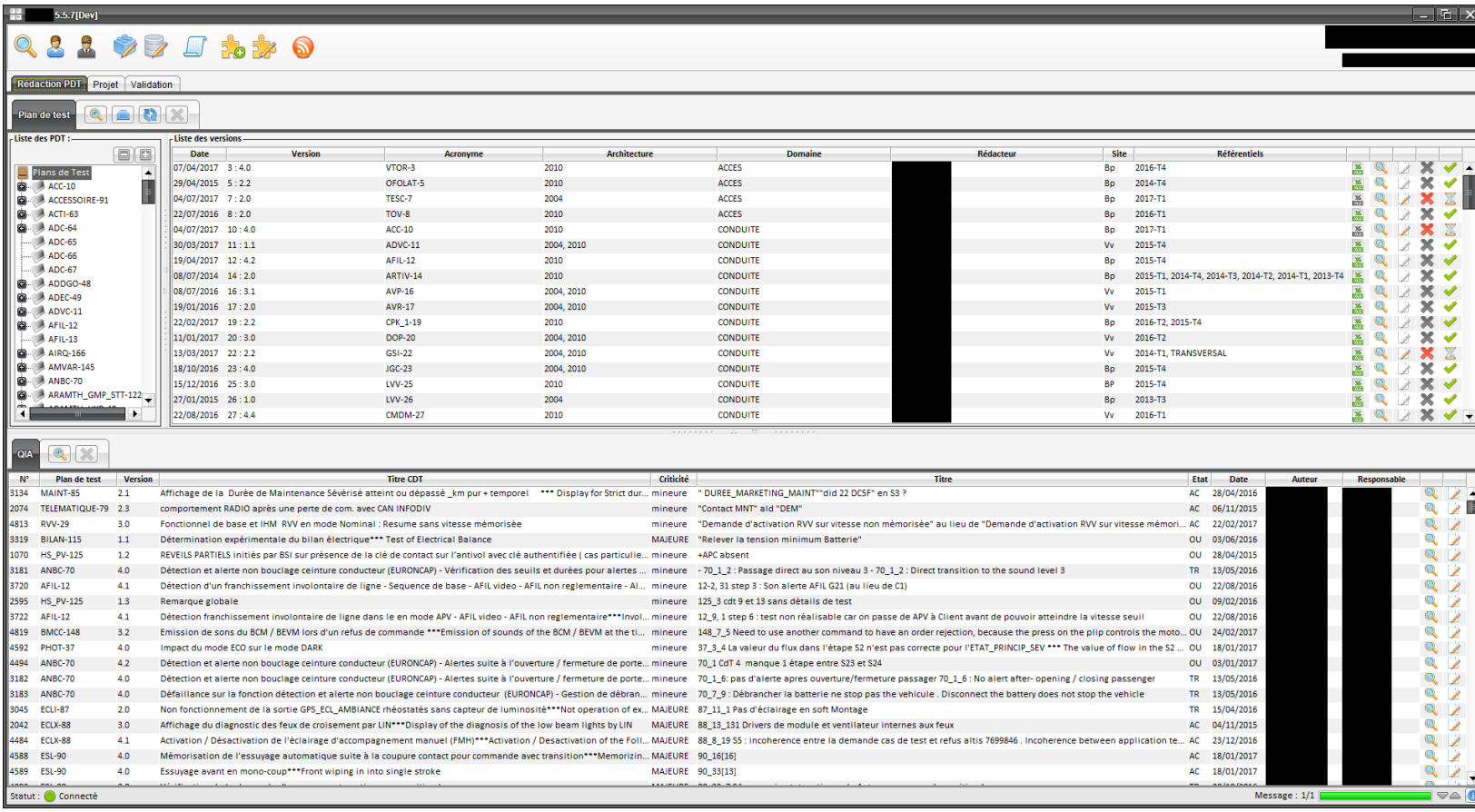
6.2 Travail effectué

Rapport technique

7 Sécurité & intégration continue

La seconde partie de mon stage a été majoritairement consacrée à l'amélioration du processus de CI au sein d'Alter Frame. Il y a beaucoup à dire sur le sujet, tant en fait que je n'ai fait qu'effleurer la surface de ce qui est possible, d'une part parce qu'il s'agit d'un secteur de l'informatique encore jeune et d'autre part parce que je me suis naturellement concentré sur l'aspect "sécurité" qui est loin d'être le seul intérêt de l'intégration continue.

7.1 Mon action sur le sujet



Graphique 9 – Fenêtre de démarrage de l'outil

8 Développement Java : Outil de gestion de tests pour un constructeur automobile

La pemière partie de mon stage a été occupé par beaucoup de développement en Java. L'idée de cette part du stage était de participer aux contrats remplis par Alter Frame et avoir ainsi une idée précise d'à quoi ressemblait le travail dans l'entreprise. Je vais profiter de cette partie pour résumer les projets auxquels j'ai participé et dans quelle mesure, sans pour autant le commanditaire dans un soucis de discrétion.

8.1 Le contexte

Produire et mettre en vente une voiture requière que celle-ci soit passée par une batterie de tests intensifs. Ce premier projet était un logiciel permettant de gérer ces tests de bout en bout : ajouter un véhicule à la base de données, établir une liste d'organes à tester, une liste de tests pour chaque organe, puis ensuite enregistrer les résultats des tests qui ont été effectués (voir graphique 9). Cela représente le cœur de la logique métier impliquée dans le logiciel, mais naturellement il y avait autour de ce noyau de nombreuses fonctionnalités plus "classiques" telles que la gestion d'authentification des utilisateurs, les différents privilèges (administrateur, utilisateur simple), etc.

De manière intéressante, j'ai remarqué en travaillant sur le workflow des tests automobile implémenté dans cet outil que celui-ci est semblable au cycle en V qui fait partie intégrante de la gestion de projet en informatique. Les détails étaient, naturellement, très différents mais ce qui semblait être les grandes

lignes de la façon dont un projet de tests devait être géré était au contraire très proche de ce que nous connaissons dans le monde de l'informatique.

Une particularité de ce projet est qu'il s'agit d'un logiciel déjà existant qui a été récupéré par Alter Frame pour de la mise à jour et de la maintenance. L'ESN initialement en charge du projet avait perdu le contrat et mon travail a, de ce fait, été majoritairement constitué de correction de bugs et, dans une moindre mesure, d'ajout de fonctionnalités mineures (voir section 8.3).

8.2 Environnement technique

- *Java 7 :* Le projet étant vieux de plusieurs années déjà, il utilise la version de Java qui était en vigueur à l'époque de sa genèse à savoir Java 7.
- *Java Swing[32] :* Pour les mêmes raisons que Java 7, le framework utilisé pour la partie graphique de l'application est Java Swing.
- *Apache ActiveMQ[33] :* L'outil fonctionne selon une architecture client-serveur classique et la communication entre le serveur et les différents clients se fait au-travers d'un système de queues et du système open source ActiveMQ.
- *MyBatis[34] :* La liaison base de données/application se fait grâce à cet ORM open source qui a la particularité de mapper des méthodes Java à des requêtes SQL[35].

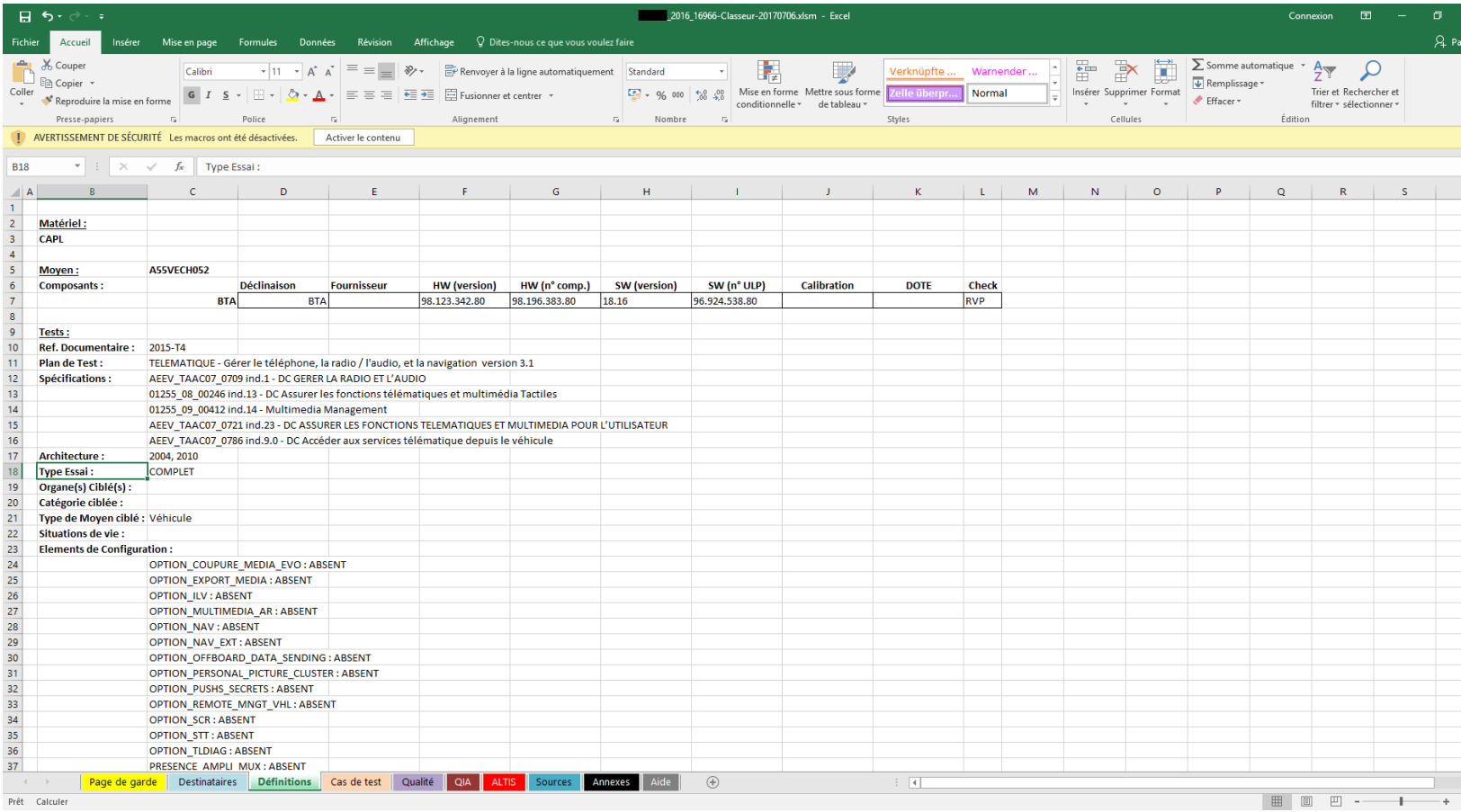
8.3 Fonctionnalités ajoutées

Comme mentionné plus haut, le gros du travail sur cet outil consistait à reprendre et débugger un code au départ écrit par une autre société. Il s'agissait donc tout à la fois de trouver des erreurs, améliorer ce qui pouvait l'être (notamment en termes de performances) et comprendre intrinséquement le fonctionnement de l'application. Tout cela représente un temps de développement considérable.

Les fonctionnalités ajoutées sont donc, comparativement, peu nombreuses. Ma première réalisation originale sur ce projet a été de modifier la génération de tableau Excel (voir figure 10) pour prendre en compte de nouvelles spécifications.

La génération en elle-même existait déjà à ce moment à l'aide de Microsoft Visual Basic, Scripting Edition[36] [9]. Ce qui était demandé par le client était d'ajouter une nouvelle catégorie d'informations à ce classeur, à savoir les *Situations de vie*.

9. VBScript pour les intimes



Matériel :										
CAPL										
Moyen :	**A5SVECH052**									
Composants :		**Déclinaison**	**Fournisseur**	**HW (version)**	**HW (n° comp.)**	**SW (version)**	**SW (n° ULP)**	**Calibration**	**DOTE**	**Check**
	BTA	BTA		98.123.342.80	98.196.383.80	18.16	96.924.538.80			RVP
Tests :										
Ref. Documentaire :	2015-T4									
Plan de Test :	TELEMATIQUE - Gérer le téléphone, la radio / l'audio, et la navigation version 3.1									
Spécifications :	AEEV_TAAC07_0709 ind.1 - DC GERER LA RADIO ET L'AUDIO									
	01255_08_00246 ind.13 - DC Assurer les fonctions télématiques et multimédia Tactiles									
	01255_09_00412 ind.14 - Multimedia Management									
	AEEV_TAAC07_0721 ind.23 - DC ASSURER LES FONCTIONS TELEMATIQUES ET MULTIMEDIA POUR L'UTILISATEUR									
	AEEV_TAAC07_0786 ind.9.0 - DC Accéder aux services télématique depuis le véhicule									
Architecture :	2004, 2010									
Type Essai :	COMPLET									
Organe(s) Ciblé(s) :										
Catégorie ciblée :										
Type de Moyen ciblé :	Véhicule									
Situations de vie :										
Elements de Configuration :										
	OPTION_COUPURE_MEDIA_EVO : ABSENT									
	OPTION_EXPORT_MEDIA : ABSENT									
	OPTION_ILV : ABSENT									
	OPTION_MULTIMEDIA_AR : ABSENT									
	OPTION_NAV : ABSENT									
	OPTION_NAV_EXT : ABSENT									
	OPTION_OFFBOARD_DATA_SENDING : ABSENT									
	OPTION_PERSONAL_PICTURE_CLUSTER : ABSENT									
	OPTION_PUSHS_SECRETS : ABSENT									
	OPTION_REMOTE_MNGT_VHL : ABSENT									
	OPTION_SCR : ABSENT									
	OPTION_STT : ABSENT									
	OPTION_TLDIAG : ABSENT									
	PRESENCE_AMPLI_MUX : ABSENT									

Graphique 10 – Classeur Excel généré par l'outil

9 Audit de sécurité et de performances

J'ai eu l'occasion, sur la fin de mon stage, de réaliser un audit pour une société travaillant dans le monde de l'assurance. Ce genre de projets sort du cadre habituel de ceux entrepris par Alter Frame. C'est un contrat historiquement entretenu par Alter Frame depuis plusieurs années, et une opportunité intéressante de s'éloigner du développement et de faire une mission de conseil.

L'audit comportait trois axes importants : la qualité, la sécurité et la performance.

9.1 Méthodologie

Mon intervention a naturellement concerné ces trois aspects, et ce de trois façons différentes.

9.1.1 Sécurité

Il s'agit naturellement de la partie où mon intervention a été la plus notable (NOTE : est-ce que ça fait pas pompeux ?!!!). Les analyses de séurité se sont faites sur site, à partir d'une machine configurée par le client : l'intervention concerne une application web disponible uniquement en intranet chez le client et leurs procédures et protocoles de sécurité rendent très compliqué d'exporter l'appplication pour l'auditer à partir des locaux d'Alter Frame.

Le test a consisté à mener des analyses avec OWASP ZAP et à analyser et rejouer les résultats, trier les faux positifs des vraies failles de sécurité et faire un état de l'évolution de l'application depuis le précédent audit qui date d'un an.

Nous avons utilisé les fonctionnalités classiques de ZAP pour obtenir un résultat le plus exhaustif possible compte tenu du peu de temps alloué à l'audit de sécurité : d'abord *spider* l'application cible pour que le proxy en découvre la majorité, avant de lancer un scan actif qui, compte tenu de la taille de l'application auditée, a pris plusieurs heures de temps d'exécution.

Le test a également incorporé une partie audit de code, supportée par l'utilisation de SonarQube et qui se recoupera avec la partie qualité. Sonar peut, en effet, être configuré pour retourner des alertes de sécurité en ce qui concerne le code analysé. Comme pour ZAP il requière une intervention humaine pour écarter les faux positifs et n'offre aucune garantie d'exhaustivité, mais cela représentait un point d'entrée efficace pour chercher des traces de risques dans le code audité.

9.1.2 Performance

La partie audit de performance a elle aussi compris deux sous parties, l'une dynamique basée sur l'utilisation de JMeter[37] et de nmon[38], l'autre statique (à savoir, une analyse de code). Je ne suis personnellement intervenu que sur la partie dynamique qui s'est effectuée sur site pour les mêmes raisons que l'analyse de sécurité.

JMeter est un outil développé par Apache qui permet de faire des tests de charge d'applications Web. Les fonctionnalités de l'outil sont impressionantes de diversité et de profondeur et comme encore une fois le temps alloué à l'auit était court, je n'ai pu qu'en effleurer la surface : mon travail a consisté à enregistrer une série de cas d'utilisation typiques en plaçant JMeter en proxy de navigation, puis à les configurer pour qu'ils soient reproductibles automatiquement (par exemple en générant des utilisateurs de manière procédurale) et à les rejouer en boucle.

JMeter produit des résultats très complets formatés en HTML qui affichent plusieurs métriques sous formes de graphiques ou de tableaux récapitulatifs, et nous avons effectué plusieurs jeux de test en augmentant à chaque fois la charge à laquelle le serveur était soumis pour observser son comportement jusq'au point où il "tombe".

Nmon pour sa part est un outil de monitoring

9.1.3

9.2 Résultats obtenus

10 Avenir

Conclusion

Penser à mettre les crédits pour le template

Références

[1] WIKIPEDIA, éd. *Intégration continue*. URL : https://fr.wikipedia.org/wiki/Int%C3%A9gration_continue (cf. p. 2).

[2] Aix-Marseille UNIVERSITÉ, éd. *Fiabilité et sécurité informatique (FSI)*. URL : http://masterinfo.univ-mrs.fr/FSI.html (cf. p. 2).

[3] WIKIPEDIA, éd. *Entreprise de services du numérique*. URL : https://fr.wikipedia.org/wiki/Entreprise_de_services_du_num%C3%A9rique (cf. p. 5).

[4] CNIL, éd. *Textes officiels européens protection des données*. URL : https://www.cnil.fr/fr/textes-officiels-europeens-protection-donnees (cf. p. 6).

[5] The Open Web Application Security Project (OWASP), éd. *The ZAP API*. URL : https://github.com/zaproxy/zaproxy/wiki/ApiDetails (cf. p. 7).

[6] The Open Web Application Security Project (OWASP), éd. *Command Line*. URL : https://github.com/zaproxy/zap-core-help/wiki/HelpCmdline (cf. p. 7).

[7] Daniel GRUNWELL, éd. *A simple tool for interacting with OWASP ZAP from the commandline.* Un wrapper pour utiliser l'API Python à travers la ligne de commande, non-officiel. URL : https://github.com/Grunny/zap-cli (cf. p. 7).

[8] GITLAB, éd. *About GitLab*. URL : https://about.gitlab.com/ (cf. p. 9).

[9] GITLAB, éd. *GitLab Continuous Integration & Deployment*. URL : https://about.gitlab.com/features/gitlab-ci-cd/ (cf. p. 9).

[10] SONARSOURCE, éd. *SonarQube*. URL : https://www.sonarqube.org (cf. p. 9).

[11] The Open Web Application Security Project (OWASP), éd. *OWASP Zed Attack Proxy*. URL : https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project (cf. p. 9).

[12] The Open Web Application Security Project (OWASP), éd. *OWASP, the free and open software security community*. URL : https://www.owasp.org/index.php/Main_Page (cf. p. 9).

[13] The Open Web Application Security Project (OWASP), éd. *The OWASP ZAP core project*. Plus de 60 commits en juin 2017. URL : https://github.com/zaproxy/zaproxy (cf. p. 9).

[14] PortSwigger LTD, éd. *Burp suite editions and features*. URL : https://portswigger.net/burp (cf. p. 11).

[15] Docker INC., éd. *Docker*. URL : https://www.docker.com/ (cf. p. 12).

[16] Docker INC., éd. *The Docker Store*. URL : https://store.docker.com/ (cf. p. 12).

[17] AIRPAIR.COM, éd. *8 Proven Real-World Ways to Use Docker*. URL : https://www.airpair.com/docker/posts/8-proven-real-world-ways-to-use-docker (cf. p. 12).

[18] The Linux Foundation (initialement GOOGLE), éd. *Kubernetes : Production-Grade Container Orchestration*. URL : https://kubernetes.io/ (cf. p. 12).

[19] Docker INC., éd. *Swarm mode overview*. URL : https://docs.docker.com/engine/swarm/ (cf. p. 12).

[20] Clark EVANS, Brian INGERSON et Oren BEN-KIKI. *YAML : YAML Ain't Markup Language*. URL : http://yaml.org/ (cf. p. 12).

[21] YAML.ORG, éd. *YAML 1.1 Reference card*. URL : http://www.yaml.org/refcard.html (cf. p. 12).

[22] Squale PROJECT, éd. *Squale : Software QUALity Enhancement*. URL : http://www.squale.org (cf. p. 14).

[23] SMARTBEAR, éd. *SoapUI*. URL : https://www.soapui.org/ (cf. p. 14).

[24] GITLAB, éd. *GitLab Workflow : an overview*. URL : https://about.gitlab.com/2016/10/25/gitlab-workflow-an-overview/ (cf. p. 16).

[25] GITLAB, éd. *Using docker build*. URL : https://docs.gitlab.com/ee/ci/docker/using_docker_build.html (cf. p. 17).

[26] Andreas JUNG. *On Docker security : 'docker' group considered harmful*. URL : https://www.andreas-jung.com/contents/on-docker-security-docker-group-considered-harmful (cf. p. 17).

[27] The Open Web Application Security Project (OWASP), éd. *Docker*. Page du wiki de ZAP concernant les images Docker. URL : https://github.com/zaproxy/zaproxy/wiki/Docker (cf. p. 19).

[28] The Open Web Application Security Project (OWASP), éd. *zaproxy/build/docker/*. Dépôt contenant les Dockerfiles de ZAP maintenus par l'OWASP. URL : https://github.com/zaproxy/zaproxy/tree/develop/build/docker (cf. p. 19).

[29] SOFTPLAN, éd. *OWASP Zed Attack Proxy (ZAP) Maven plugin*. Une image pour utiliser ZAP comme un *goal* Maven. URL : https://github.com/pdsoftplan/zap-maven-plugin (cf. p. 19).

[30] GITLAB, éd. *https ://docs.gitlab.com/ee/ci/yaml/#anchors*. URL : https://docs.gitlab.com/ee/ci/yaml/%5C#anchors (cf. p. 20).

[31] GITLAB, éd. *Secret variables*. URL : https://docs.gitlab.com/ee/ci/variables/%5C#secret-variables (cf. p. 20).

[32] WIKIPEDIA, éd. *Swing (Java)*. URL : https://en.wikipedia.org/wiki/Swing_%28Java%29 (cf. p. 26).

[33] Apache Software FOUNDATION, éd. *Apache ActiveMQ*. URL : http://activemq.apache.org/ (cf. p. 26).

[34] Clinton BEGIN. *MyBatis*. URL : http://www.mybatis.org/mybatis-3/ (cf. p. 26).

[35] DBOUKELMOUNE. *Présentation de MyBatis*. Comparaison de MyBatis à JDBC et Hiberte. URL : https://blog.zenika.com/2012/03/28/presentation-de-mybatis/ (cf. p. 26).

[36] Microsoft CORPORATION. *How to automate Excel from a client-side VBScript*. URL : https://support.microsoft.com/en-us/help/198703/how-to-automate-excel-from-a-client-side-vbscript (cf. p. 26).

[37] Apache Software FOUNDATION, éd. *Apache JMeter*. URL : http://jmeter.apache.org/ (cf. p. 28).

[38] Nigel GRIFFITHS. *Nigel's performance monitor for Linux*. URL : http://nmon.sourceforge.net/pmwiki.php (cf. p. 28).