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

1 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.

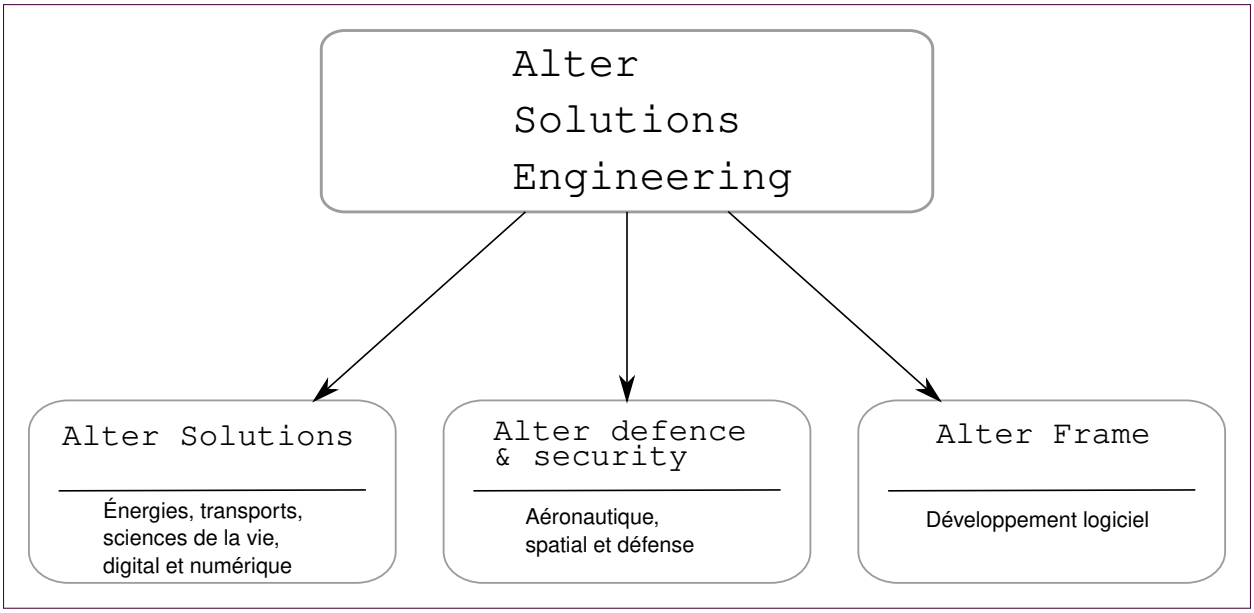
Ce sujet a l'avantage d'être ouvert et diversifié. Il me permet d'une part de travailler sur du pur développement et d'autre part de mettre en pratique la composante sécurité de la formation FSI[2] [2], tout en découvrant les concepts de CI qui m'étaient jusque là étrangers, ainsi que des technologies 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

Graphique 1 – Alter Solutions Engineering et ses filiales



2 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.

2.1 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 2.2).

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).

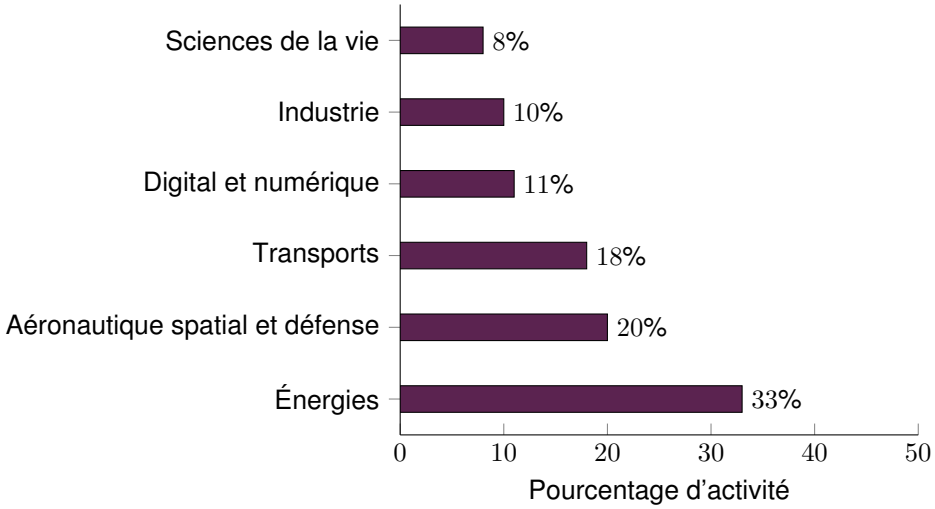
2.1.1 Alter Solutions

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.

2.1.2 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 8). 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.

2.1.3 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.

2.2 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 3.

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].

3 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.

3.1 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.

3.2 Développement Java

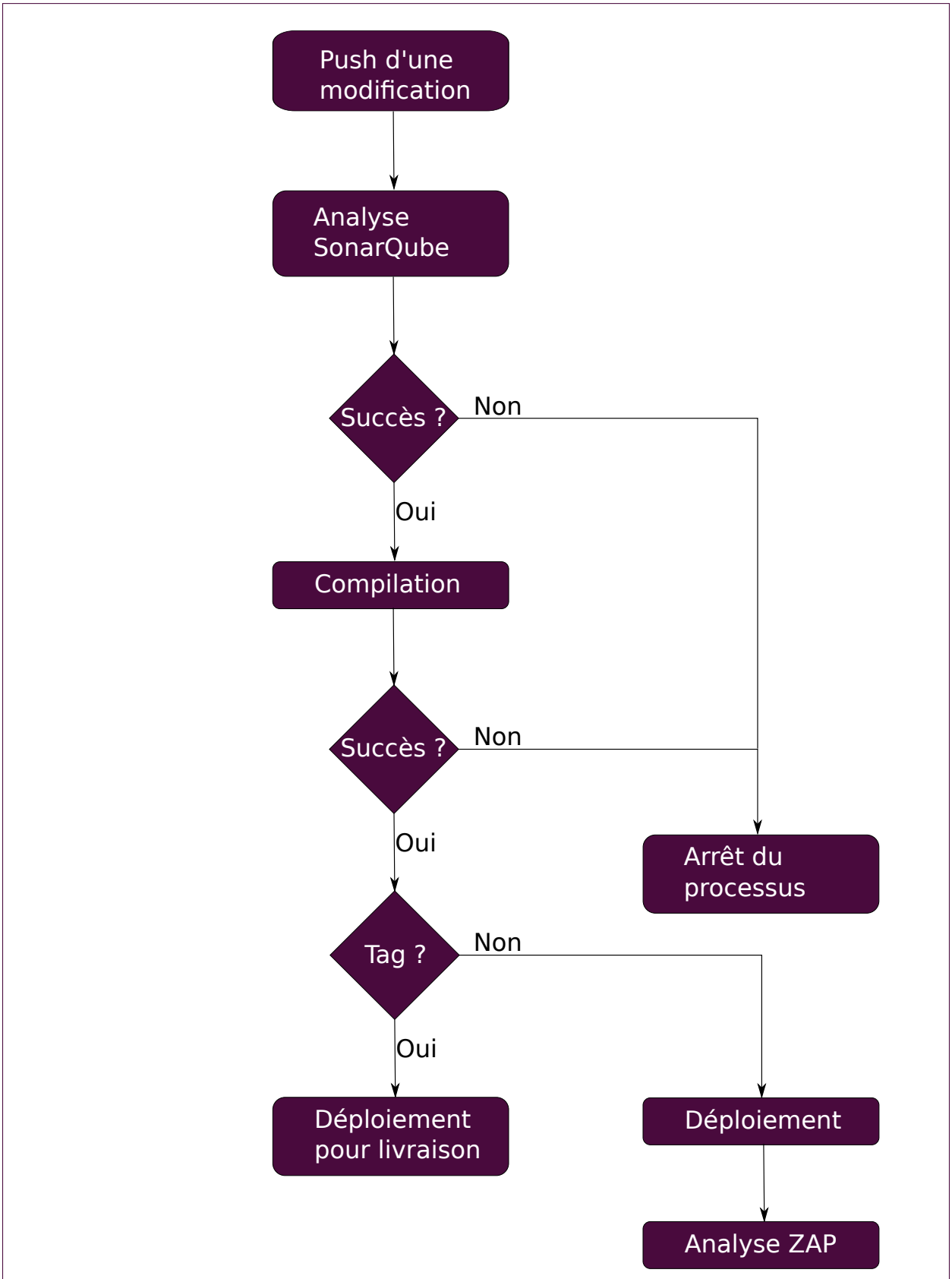
L'aspect Java, dans mon stage, était plus flou dans la mesure où il était sujet aux projets qui seraient en cours et en besoin de soutien au moment de mon arrivée dans l'entreprise. En pratique, cela a été majoritairement du développement sur un projet de gestions de tests sur des voitures (l'image 4 est une capture de la page d'accueil du logiciel), pour un constructeur automobile, ainsi que des interventions ponctuelles sur plusieurs autres projets destinés au même client.

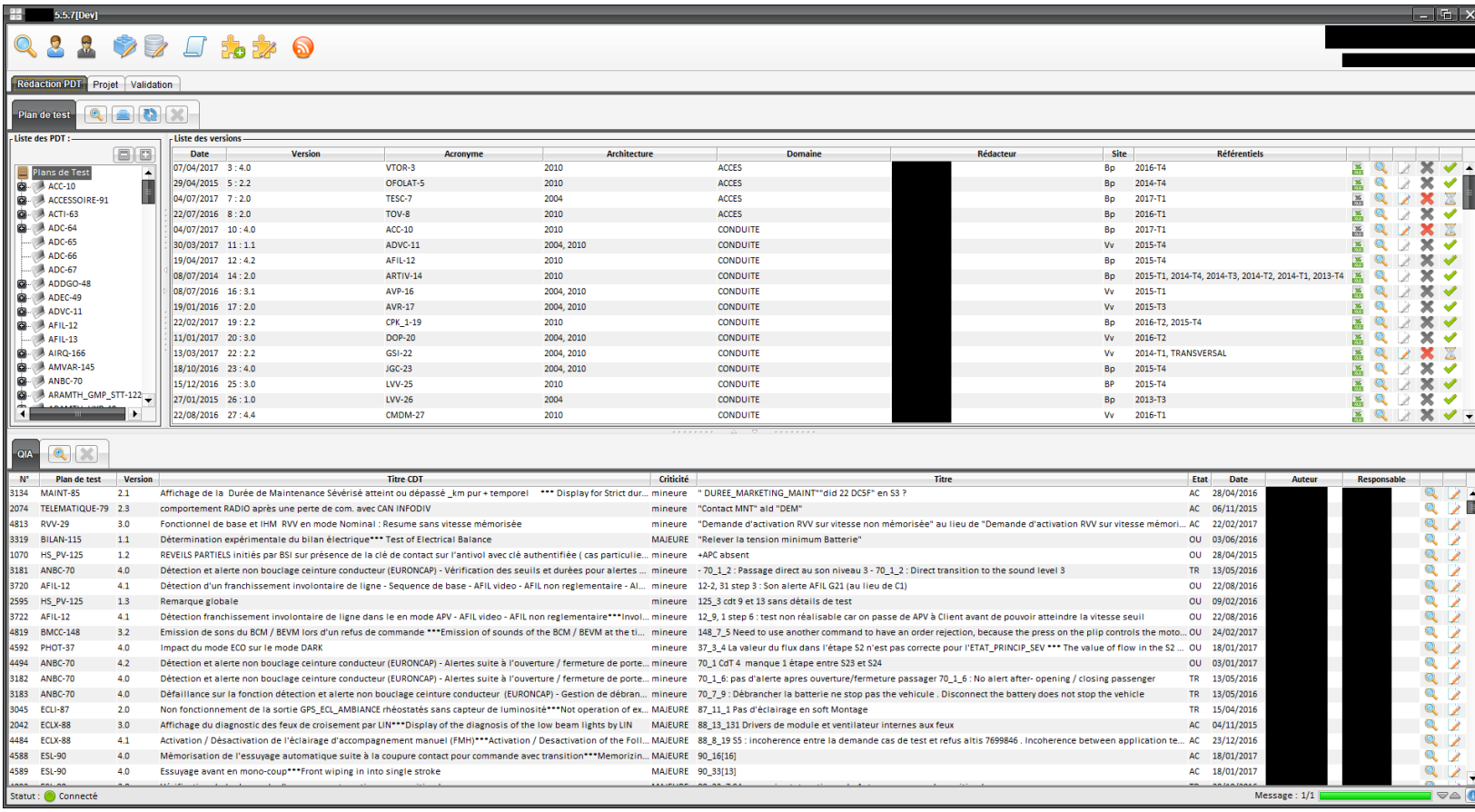
3.3 Interventions en fonction du besoin

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





Graphique 4 – Fenêtre de démarrage de l'outil

4 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 5, 6 et 7.

4.1 CI

4.1.1 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.

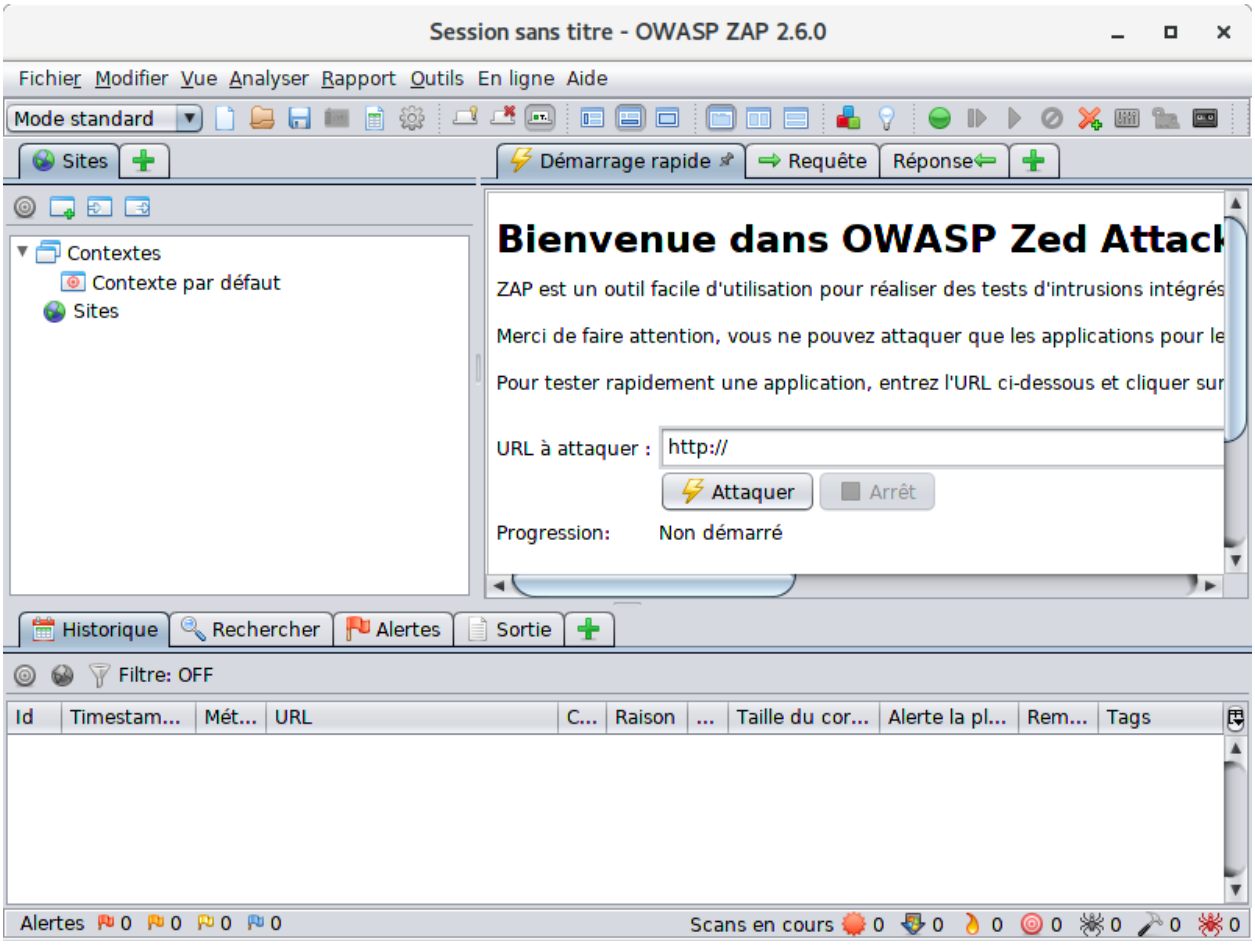
Le principe est que les actions décrites ci-dessus, compilation et analyse de code, sont effectuées à chaque push sur le serveur GitLab. Ce fonctionnement peut ensuite être affiné, pour ne se produire que lorsqu'un tag git est pushé ou sur certaines branches (branche master, tag de release, etc).

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.

4.1.2 ZAP : Zed Attack Proxy

ZAP[11] (voir figure 5) 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 5 – 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.

4.1.3 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.

4.1.4 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

4.2 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].

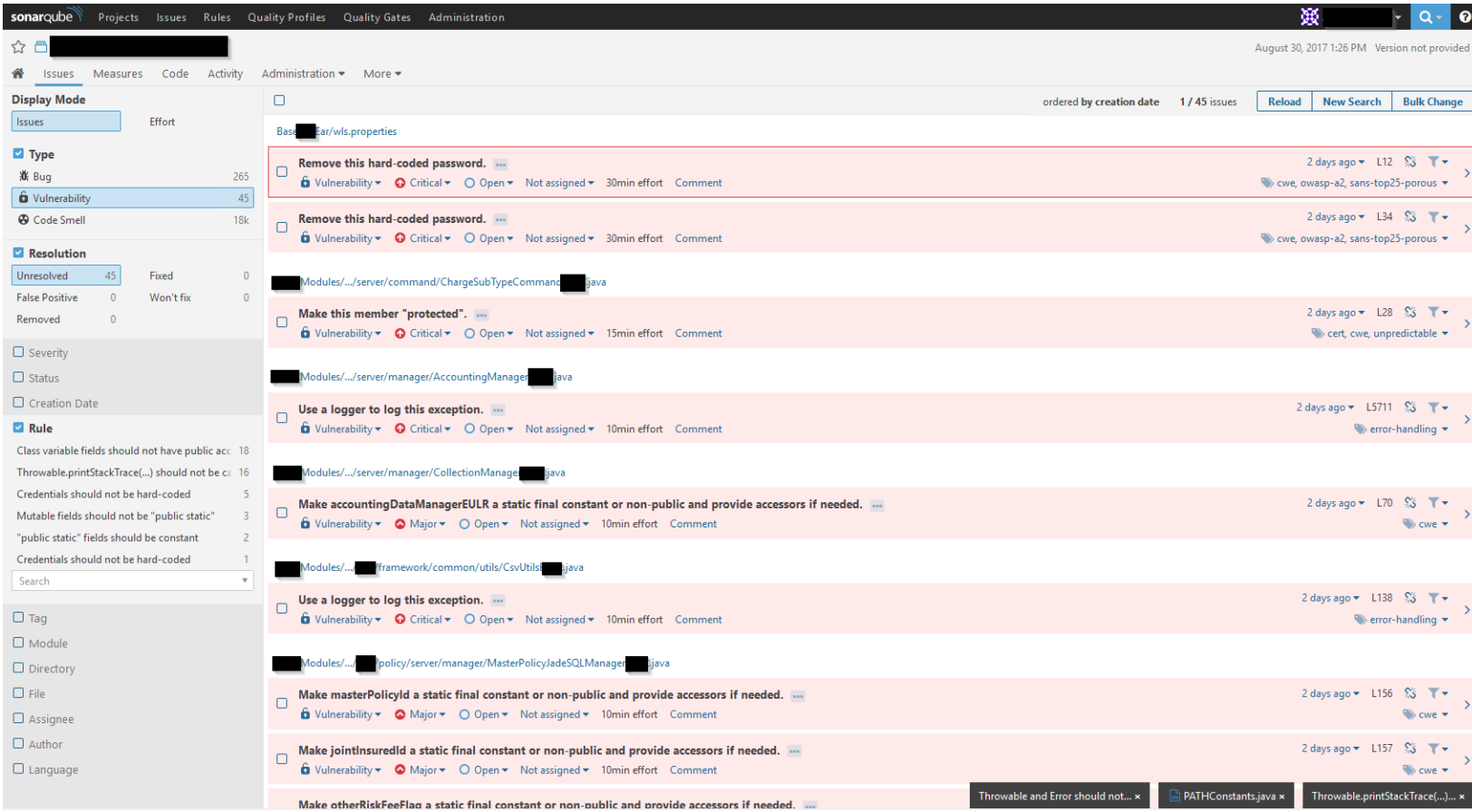
4.3 Audit technique

4.3.1 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 6 – Extrait des résultats de sécurité du scanner Sonar

4.3.2 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 6 liste une partie des vulnérabilités que sait détecter Sonar).

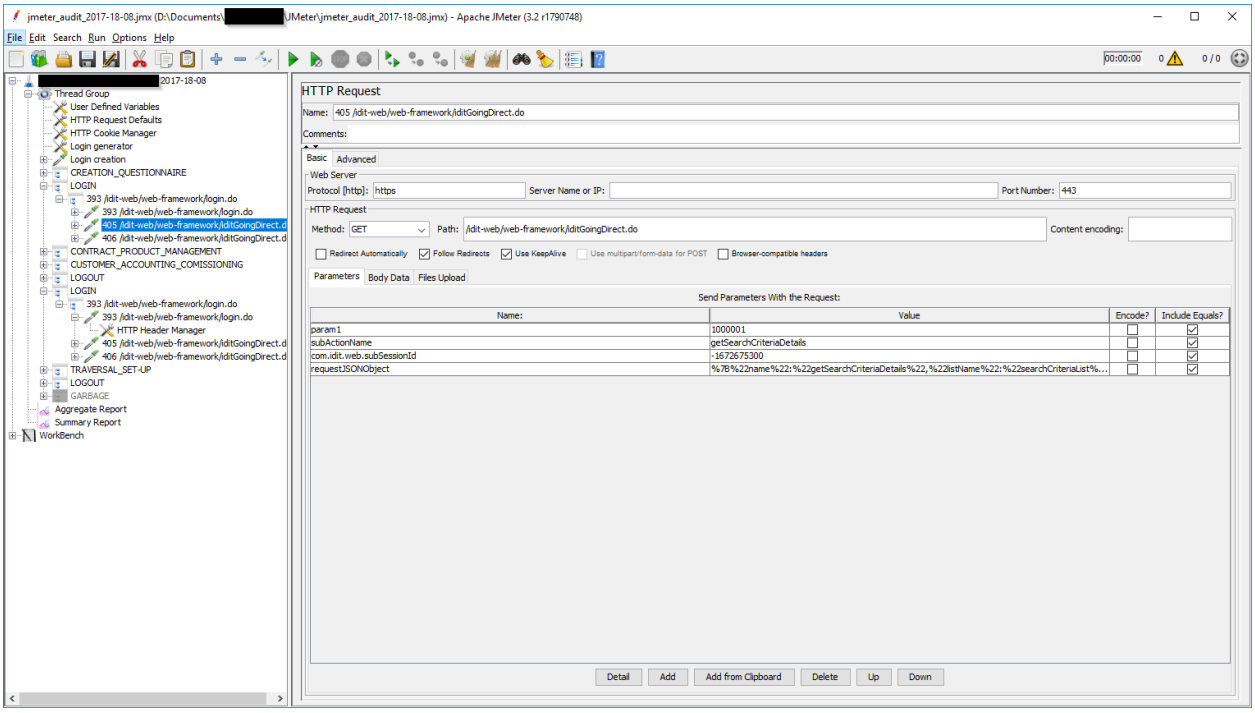
C'est durant l'audit que j'ai majoritairement utilisé Sonar, mais les scripts de CI qui étaient en plus avant mon arrivée chez Alter Frame incorporaient déjà des analyses Sonar effectuées sur le code. C'est d'ailleurs de ce fait (mon tuteur ayant déjà de l'expérience avec ce logiciel) ainsi que du faible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

4.3.3 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.

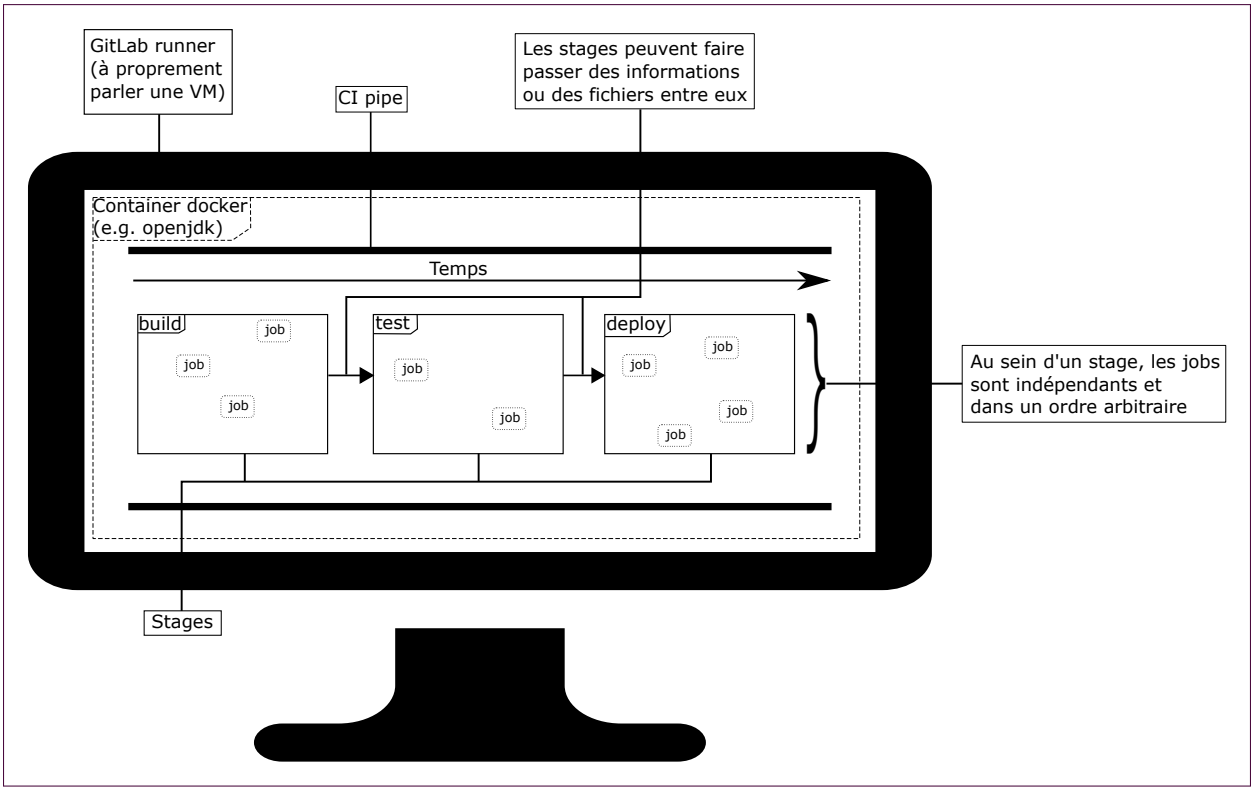
JMeter est un outil de test de performances pour applications web en général. Le panel de fonctionnalités



Graphique 7 – 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 7), puis rejouer celui-ci en boucle et plusieurs fois en parallèle pour tester les limites de l'application.

Graphique 8 – Résumé du processus de CI (*build*, *test* et *deploy* sont les stages par défaut)



5 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.

5.1 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
15
```

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.

5.2 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.

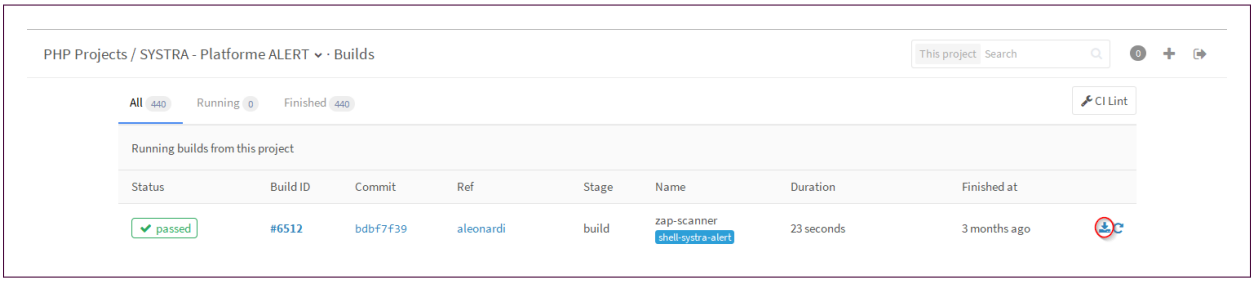
Néanmoins, la solution de l'API rajoute une contrainte : en plus de ZAP, il faut que l'environnement de CI dispose du langage de l'API. Les deux APIs principales maintenues par la communauté de développeurs "officiels" de ZAP sont celles en Java et en Python. J'ai retenu l'utilisation de Python d'une 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 8).

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 9 – 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
12
```

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 9).

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

5.3 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
25
```

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

6 Synthèse du développement Java

En termes de temps investi, le développement Java était à peu près équivalent à l'intégration continue dans mon stage. Le travail effectué sera plus court à synthétiser cependant, en ce sens qu'il se rapproche plus à ce que j'ai pu connaître de Java au cours de ma formation.

6.1 Rappel du contexte

J'ai travaillé sur plusieurs projets en Java, mais il n'y en a qu'un où mon intervention n'a pas été anecdotique : un projet de gestion de tests de voiture pour un constructeur automobile. Le client avait utilisé pendant longtemps un middleware propriétaire pour la communication client/serveur, et avait décidé de changer pour ActiveMQ en même temps que de changer de prestataire pour le développement du logiciel (c'est à cette occasion qu'Alter Frame a récupéré le contrat).

La solution initiale était Notify (je n'ai pas réussi à trouver de documentation là-dessus), mais celle-ci avait déjà disparu à mon arrivée sur le projet. En revanche, ActiveMQ n'était pas encore pleinement fonctionnel, son intégration ayant entraîné des régressions et des bugs nouveaux.

Malgré cela, ce changement était utile d'une part pour l'économie du prix de la licence (Apache ActiveMQ est librement distribué sous licence Apache 2.0... sans surprise) et d'autre part pour profiter de la stabilité d'une solution plus largement adoptée et maintenue.

En plus du développement à proprement parler, j'ai au à créer un installeur pour le logiciel en question, ce qui s'est avéré beaucoup moins trivial que ce que j'avais tout d'abord anticipé.

6.2 ActiveMQ

L'intégration d'ActiveMQ à un projet Java est simple : un des interlocuteurs crée une connexion, un message qui doit nécessairement avoir une destination, et un producteur qui est le processus qui va se charger de l'envoi à proprement parler. Les actions sont presque identiques pour le deuxième interlocuteur, excepté que celui-ci crée un consommateur[32] (voir listing 6 tiré de la documentation).

L'avantage du système de queue d'ActiveMQ est que le producteur et le receveur n'ont pas besoin d'être disponibles en même temps, le *broker* (le daemon qui reçoit et transmet les messages) les garde en mémoire jusqu'à ce qu'ils soient consommés. C'est la destination qui sert à savoir qui reçoit quoi : elle correspond à une queue, quand un expéditeur renseigne cette destination un message est ajouté à la file et quand c'est un consommateur, un message est retiré de la file (c'est un modèle FIFO).

ActiveMQ propose une interface web pour visualiser le traffic géré par le *broker*, les différentes queues, le nombre de producteurs/consommateurs, nombre de messages échangés, etc, à l'adresse `http://localhost:8161/admin/`.

```
  public static class HelloWorldProducer implements Runnable {
    public void run() {
        try {
            ActiveMQConnectionFactory connectionFactory = new ActiveMQConnectionFactory("vm
    ://localhost");

            Connection connection = connectionFactory.createConnection();
            connection.start();

            Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);

            Destination destination = session.createQueue("TEST.FOO");

            MessageProducer producer = session.createProducer(destination);
            producer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);

            String text = "Hello world! From: " + Thread.currentThread().getName() + " : " +
     this.hashCode();
            TextMessage message = session.createTextMessage(text);

            producer.send(message);

            session.close();
            connection.close();
        }
        catch (Exception e) {
            System.out.println("Caught: " + e);
            e.printStackTrace();
        }
    }
}

public static class HelloWorldConsumer implements Runnable, ExceptionListener {
    public void run() {
        try {
            ActiveMQConnectionFactory connectionFactory = new ActiveMQConnectionFactory("vm
    ://localhost");
            Connection connection = connectionFactory.createConnection();
            connection.start();
            connection.setExceptionListener(this);
            Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
            Destination destination = session.createQueue("TEST.FOO");
            MessageConsumer consumer = session.createConsumer(destination);

            Message message = consumer.receive(1000);

            if (message instanceof TextMessage) {
                TextMessage textMessage = (TextMessage) message;
                String text = textMessage.getText();
                System.out.println("Received: " + text);
            } else {
                System.out.println("Received: " + message);
            }

            consumer.close();
            session.close();
            connection.close();
        } catch (Exception e) {
            System.out.println("Caught: " + e);
            e.printStackTrace();
        }
    }
```


Extrait 6 – "Un exemple écourté d'échange de messages *via* ActiveMQ"

La mécanique de départ est assez simple et c'est tout l'intérêt de cet outil ; la vraie difficulté a résidé dans la recherche de toutes les utilisations qui étaient faites de l'ancienne librairie Notify, et leur remplacement sans générer d'effet de bord ; plus facile à dire qu'à faire comme toujours quand il s'agit de refactorer du code !

Il est difficile d'isoler un exemple de code sur lequel j'ai travaillé, tant cela a été de petites retouches à différents endroits des sources, suivis de longues sessions de tests avec le chef de projet pour exposer toutes les conséquences des modifications et, s'il y en avait d'inattendues, les corriger. Voici tout de même un exemple notable : listings 7 et 8.

```
public void onMessage(Message rawMsg) {
    switch (duo.getFirst()) {
        case BDD :
            switch (duo.getSecond()) {
                case NOTIFY_LOCK :
                    CustomLogEngine.debug("NotifyConnector - srvMsgProcess : notify lock");
                    callId = msg.getNextShort();
                    sender = msg.getNextString();
                    error = msg.getNextString();
                    CacheClient.getClientForNotify().notifyLockFromServer(error, callId);
                    break;
                case NOTIFY_UNLOCK :
                    CustomLogEngine.debug("NotifyConnector - srvMsgProcess : notify lock");
                    callId = msg.getNextShort();
                    sender = msg.getNextString();
                    short callIdToUnlock = msg.getNextShort();
                    error = msg.getNextString();
                    CacheClient.getClientForNotify().notifyUnlockFromServer(error,
    callIdToUnlock, callId);
                    break;

                // ...

            }
        case ALL_TYPE :
            switch (duo.getSecond()) {
                case LOCK :
                    // A chaque accès à une ressource
                    // synchronisée on reçoit la liste des lock que nous
                    // fournit le serveur
                    updateLocalLockMap(msg);
                    break;

                // ...

            }

        // ...

    }
}

```

Extrait 7 – "Gestion des verrous à la réception d'un message côté client"

```java
private void updateLocalLockMap(CustomMessage msg) {
    String dest = msg.getDest();
    if (this.applicationName.equals(dest)) {
        synchronized (MainModel.getInstance().getMapLock()) {
            try {
                if (msg.hasNextData()) {
                    CustomLocker mapl = (CustomLocker) msg.getNextObject();
                    if (mapl != null) {
                        MainModel.getInstance().setMapLock(mapl.getMapLock());
                    }
                }
            } catch (Exception e) {
                CustomLogEngine.error(CustomLogEngine.getStackTrace(e));
            }
        }
    }
}

```

Extrait 8 – "Méthode permettant de mettre à jour les ressources verrouillées"

Le logiciel en question gère la modification d'affaires, c'est-à-dire du récapitulatif de l'état d'un véhicule en termes de tests qui ont été effectués dessus, des résultats obtenus jusque là, etc. Tous les clients connectés à un même serveur peuvent visualiser ces affaires en même temps, mais un seul peut en modifier une à la fois pour éviter des modifications concurrentes de la base de données.

Échanger efficacement les informations sur un tel verrou, les maintenir à jour et les faire respecter, s'est avéré une tâche longue et difficile. Les listings cités plus haut représentent la partie client de ce code, que j'ai partiellement écrite : on maintient une *map* de verrous qui associe à une ressource l'utilisateur qui en possède le verrou (ou ne contient pas d'entrée, si la ressource est libre).

Quand un client essaie d'accéder à une telle ressource, on vérifie la présence de la ressource dans la *map* ; si elle est présente l'utilisateur voit un message d'erreur apparaître et sinon, il ouvre la ressource et envoie simultanément un *lock* au serveur qui va le diffuser à tous les utilisateurs connectés en plus de mettre à jour sa propre *map* des verrous pour pouvoir la transmettre aux nouveaux utilisateurs qui se connectent.

6.3 Création d'un installeur

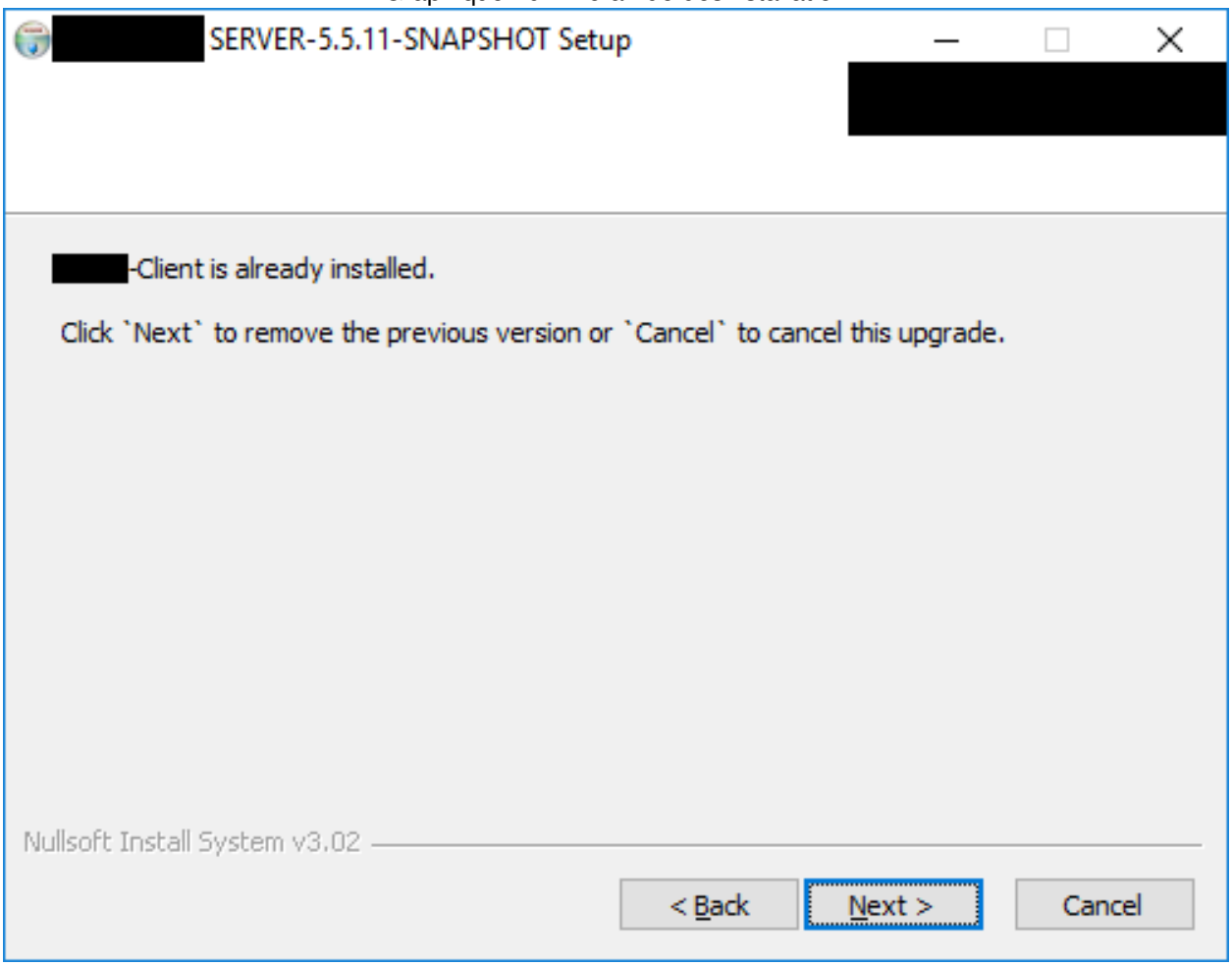
Il ne s'agit là pas de Java mais de NSIS[33] : un outil open source pour générer des (dés)installeurs sous Windows, indépendants du langage de l'application à installer. NSIS est également le nom du langage de script utilisé pour créer l'installeur ; complexe mais d'une grande flexibilité, il couvre un large éventail de fonctionnalités et a surtout une documentation de qualité, claire et détaillée[34].

Un résumé de la construction d'un script NSIS : un installeur est composé de page, ce sont les différentes fenêtres qui s'affichent au cours de l'installation. NSIS propose les plus classiques par défaut (e.g. choisir le chemin d'installation) et des pages personnalisées peuvent être créées. L'installeur affiche les pages dans l'ordre dans lequel elles sont déclarées dans le script.

Chaque page comporte une ou plusieurs fonctions qui détaillent son fonctionnement, peuvent annuler l'affichage de la page ou empêcher l'utilisateur de passer à la suivante sous certaines conditions (e.g. « Vous devez fermer l'application avant de la désinstaller »).

Enfin, des sections sont appelées pour l'installation à proprement parler des éléments. La différence avec les fonctions est que ces dernières définissent le comportement de la page d'un point de vue GUI. Les sections vont modifier le registre, copier des fichiers, créer des raccourcis, etc. La page par défaut `COMPONENTS` permet à l'utilisateur de choisir les composants qu'il installe, chaque composant correspond à une section du même nom, et la page par défaut `INSTFILES` installe chaque section correspondant à un composant sélectionné.

Graphique 10 – Écran de désinstallation



Des exemples typiques sont proposés avec l'installation de NSIS, l'installation des composants principaux de l'application n'a donc pas demandé de grandes modifications excepté le chemin vers les différents éléments. Un composant plus intéressant est la page proposant de désinstaller une version précédente de l'application avant d'installer la version actuelle (cf. capture d'écran **??**).

Un si simple message a demandé beaucoup de travail car il fallait d'une part comprendre le fonctionnement de NSIS pour ne l'afficher que sous certaines conditions, et d'autre part se pencher sur celui de Windows pour vérifier la présence d'une installation précédente.

```
1     Page custom checkIfAlreadyInstalled checkIfRunning
2     Page custom uninstallBeforeInstall ""
3
4     Var Tmp
5     Var AlreadyInstalled
6     Var UninstallerPath
7
8     Function checkIfAlreadyInstalled
9         StrCpy $AlreadyInstalled "false"
10
11        ReadRegStr $UninstallerPath HKCU \
12        "Software\Microsoft\Windows\CurrentVersion\Uninstall\${setup.regKey}" \
13        "UninstallString"
```

```
14          StrCmp $UninstallerPath "" abort
15
16          nsDialogs::Create 1018
17          Pop $Tmp
18
19          StrCpy $AlreadyInstalled "true"
20
21          ${If} $Tmp == error
22              Abort
23          ${EndIf}
24
25          ${NSD_CreateLabel} 0 0 100% 50% \
26          "Client is already installed. $\n$\nClick ‘Next‘ to remove the \
27          previous version or ‘Cancel‘ to cancel this upgrade."
28          Pop $Tmp
29          ${If} $Tmp == error
30              Abort
31          ${EndIf}
32
33          nsDialogs::Show
34          Goto done
35
36          abort:
37              Abort
38          done:
39      FunctionEnd
40
41      Function uninstallBeforeInstall
42          StrCmp $AlreadyInstalled "false" abort
43          HideWindow
44          ClearErrors
45          ExecWait '$UninstallerPath _?=$INSTDIR'
46          BringToFront
47          IfErrors no_remove_uninstaller done
48          Delete $UninstallerPath
49          RMDir $INSTDIR
50          no_remove_uninstaller:
51              Goto done
52          abort:
53              Abort
54          done:
55      FunctionEnd
56
57      Function checkIfRunning
58          FindProcDLL::FindProc "${exe}"
59          IntCmp $R0 1 0 notRunning
60              MessageBox MB_OK|MB_ICONEXCLAMATION "client is running. Please close it first" /
    SD IDOK
61              Abort
62          notRunning:
63      FunctionEnd
64
65      Section "Uninstall"
66          DeleteRegKey HKCU "Software\Microsoft\Windows\CurrentVersion\Uninstall\${setup.
    regKey}"
67          DeleteRegKey HKCU "Software\${setup.regKey}"
68
69          RMDir /r /REBOOTOK $INSTDIR
70
```

```
71          Delete "$DESKTOP\${diplayName}.lnk"
72
73          Delete "$SMPROGRAMS\${setup.dir}\*.*"
74
75          RMDir "$SMPROGRAMS\${setup.dir}"
76      SectionEnd
```

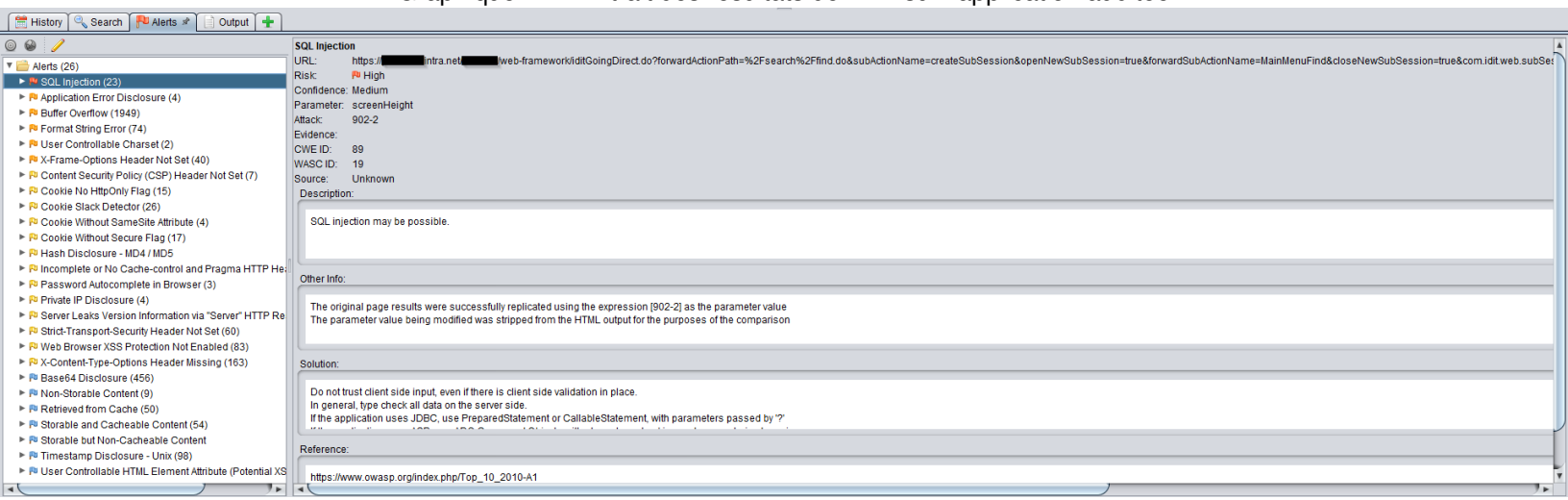
Extrait 9 – "Script NSIS permettant de désinstaller puis réinstaller une application"

Comme on peut le voir dans le listing 9, NSIS est un langage verbeux dans le sens où peu de choses sont gérées automatiquement à la génération de l'installeur : le script doit détailler chaque étape. Dans l'extrait présenté ici, je crée deux pages, une qui vérifie la présence d'une installation précédente, et le cas échéant qui vérifie si l'application est en cours d'utilisation, et une qui propose à l'utilisateur de désinstaller l'application.

Aucune de ces pages ne s'affiche si l'application n'est pas déjà présente sur le système : c'est l'effet de `abort` lignes 14 et 42 (bien qu'à proprement parler il s'agisse de jumps qui renvoient au véritable appel à `Abort` plus loin). On utilise le registre Windows pour détecter la présence de l'application sur le système, et une librairie (`FindProcDLL`) pour obtenir de Windows la liste des processus en cours d'utilisation et y chercher le nom du nôtre.

Les variables, de la forme `${nomVariable}`, qui ne sont pas définies explicitement ici sont importées du `pom.xml` du projet. Expliquer chaque point du script prendrait trop de temps ici (j'ai justement écrit une documentation bien plus complète sur NSIS, en Markdown, mais celle-ci n'est disponible que sur le serveur GitLab privé d'Alter Frame), mais cet extrait donne une idée générale du langage NSIS et de ses forces et faiblesses.

Graphique 11 – Extrait des résultats de ZAP sur l'application auditée



7 Synthèse de l'audit

Audit réalisé pour une compagnie travaillant dans le monde de l'assurance : celui-ci n'était pas prévu dans le sujet initial et entre dans la catégorie des « interventions en fonction du besoin ». L'application auditée était à la fois l'intranet de l'entreprise et un progiciel destiné au monde de l'assurance, et l'audit recouvrait plusieurs aspects :

- la sécurité, qui est importante aux yeux du client vu que la confiance de leurs propres clients en la sécurité des informations confidentielles est capitale pour assurer leur fidélité ;
- la performance, en effet il s'agit de la troisième itération de l'audit réalisée par Alter Frame et une des motivations initiales était que l'application était extrêmement lente (une page pouvait prendre plusieurs secondes à se charger dans des conditions normales) ;
- la qualité, au sens de la qualité d'écriture du code et de la maintenabilité, point qui par le passé était extrêmement coûteux pour le client à cause des nombreuses régressions et bugs liés à la piètre qualité du code, c'est donc leur principal intérêt dans l'audit.

Naturellement, en parallèle de l'aspect technique cet audit a impliqué une part non négligeable de rédaction de rapport et de documentation à l'intention des futurs auditeurs.

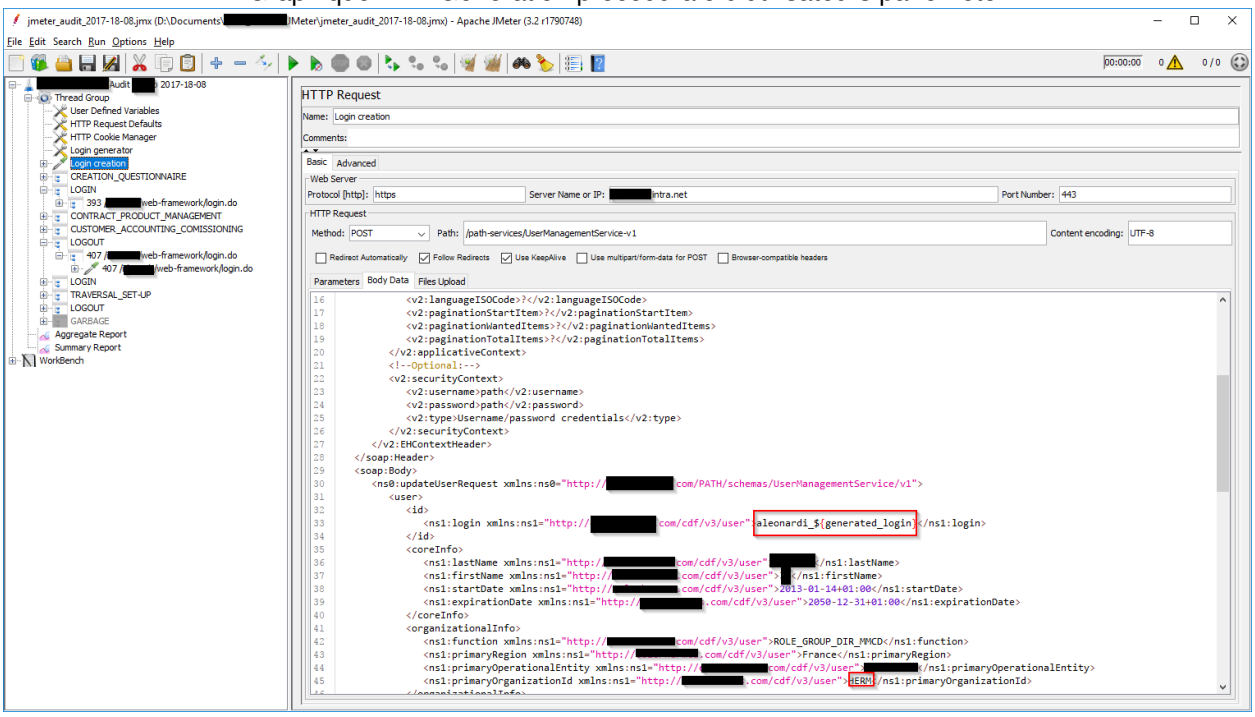
7.1 Sécurité

Le temps alloué à cette partie de l'audit n'a pas permis de mener un audit de sécurité complet comme ce que nous avons pu voir en projet. En pratique, il s'est plutôt agi d'utiliser les fonctionnalité proposées par ZAP pour avoir un survol des failles éventuelles de l'application, et de vérifier celles-ci « à la main »quand c'était possible.

L'avantage de cette approche est que ZAP, une fois configuré et lancer dans une série de scans, peut être laissé à travailler pendant que nous travaillions sur d'autres aspects de l'audit.

Les résultats étaient plutôt modérés une fois les faux positifs exclus (voir un aperçu dans la figure 11) : l'application est en Java ce qui limite d'emblée les possibilités de buffer overflow, nous n'avons pas réussi à reproduire les SQLi et XSS suspectées par ZAP, etc. Les divulgations d'informations techniques étaient légion en revanche et donc le problème le plus facile à remarquer et reproduire.

Graphique 12 – Génération procédurale d'utilisateurs par JMeter



Un des objectifs de l'audit était de comparer aux années précédentes pour avoir une mesure de l'évolution de l'application, et celle-ci était très positive : le nombre de failles de sécurité a drastiquement réduit, certaines ont entièrement disparu et d'autres énormément diminué en termes d'occurrences. On verra plus loin que ce constat s'applique aux autres aspects de l'audit aussi.

7.2 Performance

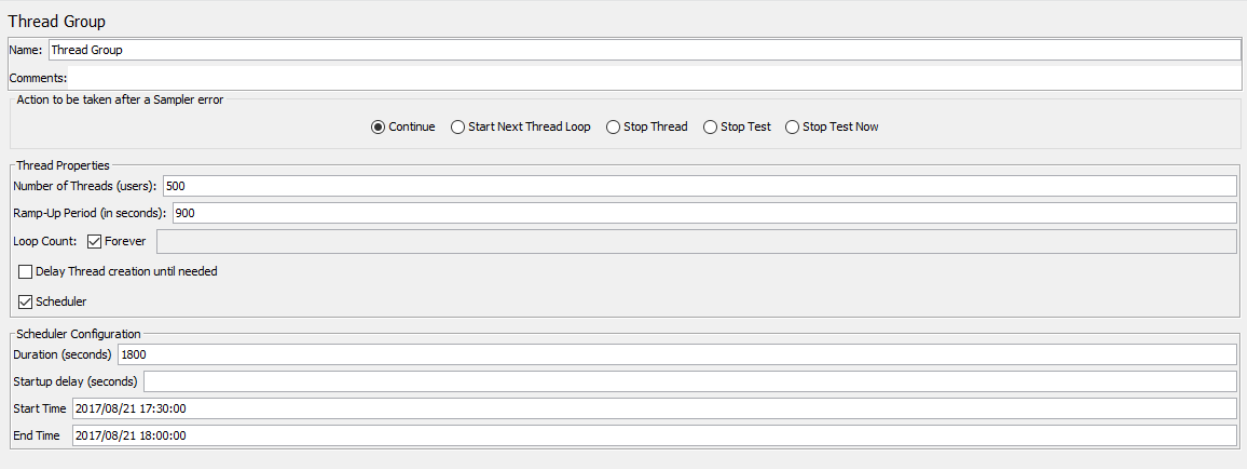
Cette partie de l'audit s'est basée sur JMeter et nmon (et SoapUI dans une moindre mesure, mais ce dernier ne permet pas de faire de mesures et il nous a simplement servi à mettre en place les requêtes envoyées par JMeter).

JMeter, une fois pris en main, est assez simple à configurer pour les fonctionnalités qui nous intéressaient. Un employé de l'entreprise cliente nous a assisté et a joué des scénarios qu'il rencontrait dans son travail de tous les jours, que nous enregistrions avec JMeter configuré comme un proxy. À quelques éléments de configuration près (e.g. la génération procédurale de nouveaux comptes sur l'intranet comme dans l'image 12 où un compteur `${generated_login}` est maintenu par JMeter et sert à générer des identifiants uniques) il n'y avait plus qu'à automatiser l'exécution des scénarios de test, et attendre puis analyser les résultats.

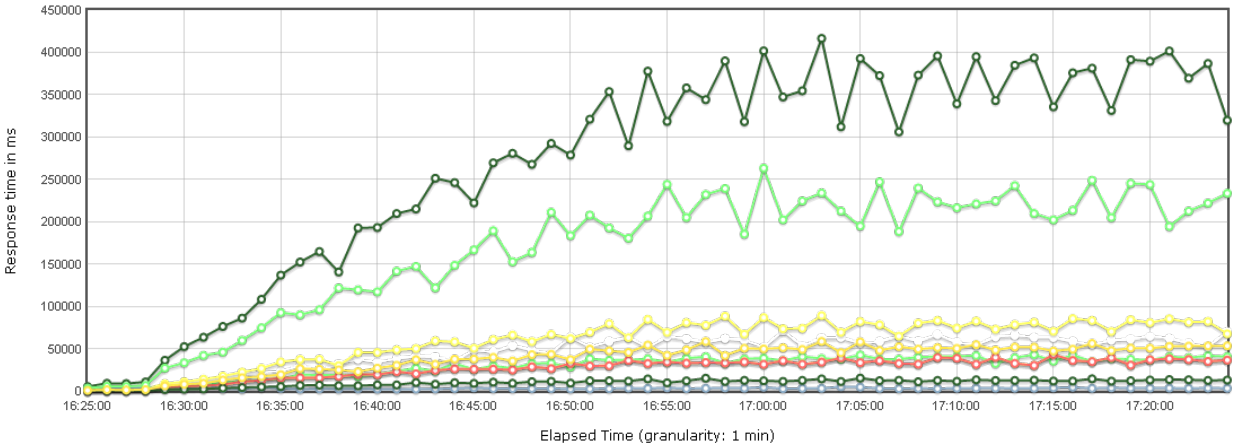
L'automatisation en question est triviale : on peut voir le panneau concerné dans l'image 13. On renseigne des informations telles que le nombre d'utilisateurs (correspondant à un thread qui va exécuter les tests en boucle), le nombre de tours de boucle que fait chaque utilisateur, la durée de la montée en charge (c'est-à-dire la fréquence de création des utilisateurs), etc.

La véritable valeur ajoutée que nous apportons arrive au moment de choisir les valeurs pour ces tests, et d'en interpréter les résultats. Outre des cas ne faisant intervenir que peu d'utilisateurs simultanés, que nous avons joué à titre d'exemple pour prendre en main l'outil, les deux configurations importantes sont une à 100 utilisateurs simultanés (ce qui est plus que l'affluence réelle que doit supporter l'application, mais reste vraisemblable) et une à 500 utilisateurs (pour voir à quel moment précis le serveur devenait surchargé et

Graphique 13 – Configuration des tests joués par JMeter



Graphique 14 – Extrait des résultats fournis par JMeter, pour le test à 100 utilisateurs



inaccessible).

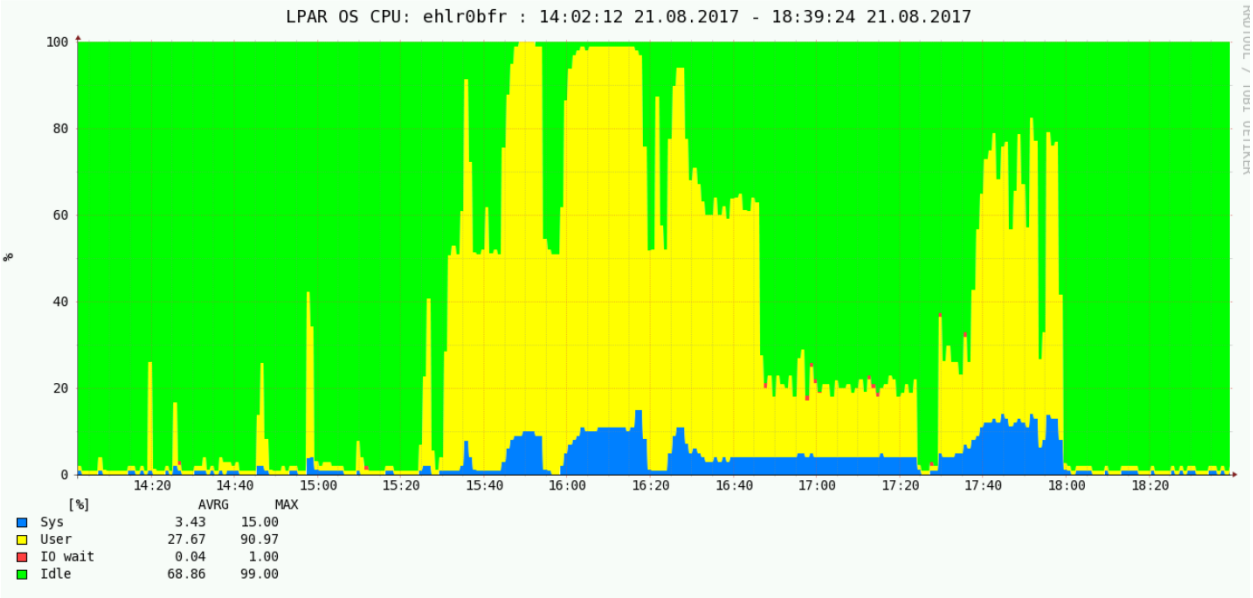
JMeter produit des résultats très complets et, avec la configuration par défaut, presque illisibles car affichant trop d'informations. La capture d'écran 14 est un extrait que j'ai affiné des temps de réponse du serveur en fonction du temps écoulé : les temps affichés peuvent sembler extrêmement longs (le pic est à 417s environ) mais ils sont à relativiser car chaque courbe représente un ensemble d'opérations (une *chaîne fonctionnelle*).

Malgré cela, les temps restent longs : compte tenu des opérations que regroupe chaque chaîne fonctionnelle, plus de cinq minutes pour la plus longue d'entre elles représente un temps excessif mais qui ne rend pas l'application inutilisable. Notre objectif, dans cet audit, était de mettre en avant les éléments les plus consommateurs de temps, mais pas d'analyser le code et fournir des suggestions d'amélioration. Le rapport d'audit s'est donc arrêté à souligner quels éléments faisaient goulot d'étranglement, laissant la charge d'améliorer la situation au client.

J'ai parlé plus tôt d'un autre outil : nmon. Il nous a servi à surveiller la consommation de ressources (mémoire et CPU) du seveur, ce qui était intéressant pour les corréler aux données de JMeter mais ne nous a pas apporté d'information neuve. Un exemple de résultats de nmon : la capture 15.

Une fois encore nous voulions mettre en relief les changements survenus depuis la précédente itération de l'audit, et encore une fois ils étaient positifs : d'une part les temps de réponse, bien qu'encore longs, se sont

Graphique 15 – Extrait des résultats fournis par nmon, pour le test à 500 utilisateurs : noter la saturation du CPU à partir de 16h



nettement améliorés (la page d'accueil, après la connexion, pouvait prendre plus d'une minute à charger).

Qui plus est, la résistance à la charge est sans commune mesure : le serveur était, dans l'audit précédent, surchargé par 100 utilisateurs simultanés (le taux de réponse aux requêtes diminuait dés 50 utilisateurs). Dans l'audit auquel j'ai participé, la première chute en temps de réponse se fait à 8 minutes soit environ à 260 utilisateurs simultanés.

7.3 Qualité

Pour partie, cet aspect de l'audit n'a pas encore été traité au moment où je rédige ce rapport. Je suis entrain de réaliser des analyses de code avec le chef de projet sous la tutelle de qui je travaille et il est trop tôt pour donner des résultats.

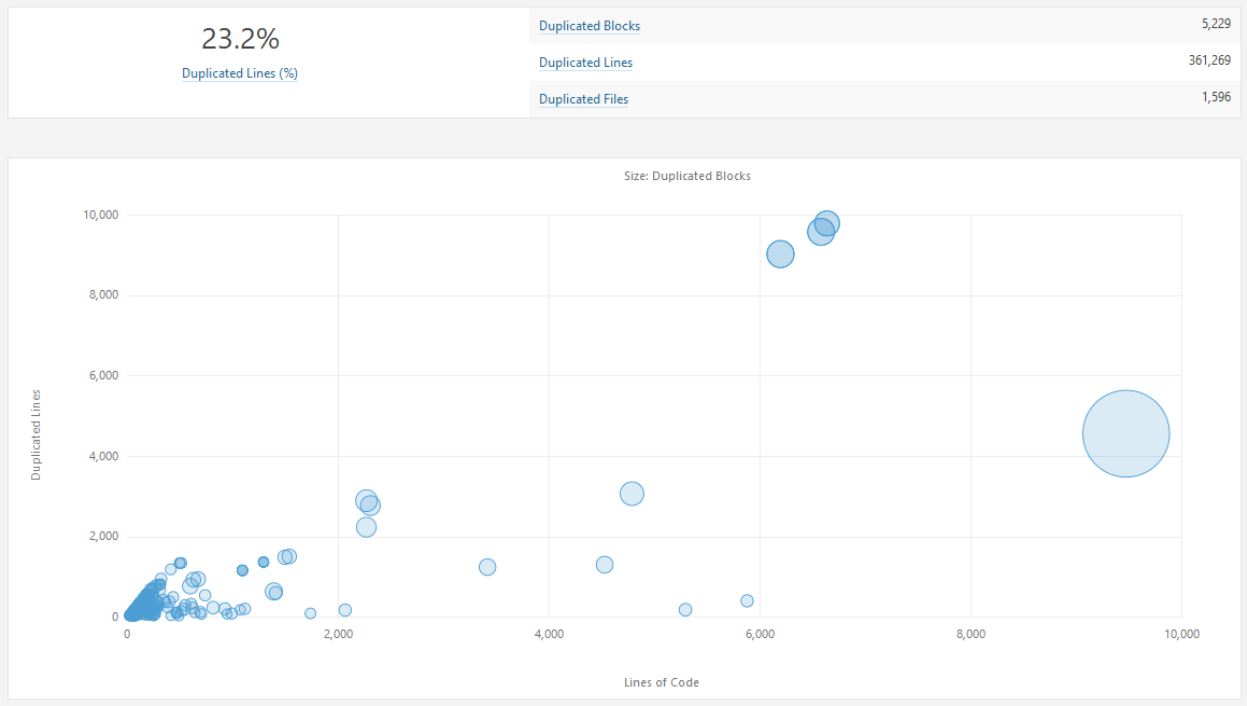
Il est néanmoins possible de présenter ce que le client demande précisément en la matière :

- analyser un certain nombre de problèmes récemment corrigés (les *root causes*) et statuer sur la stabilité des corrections apportées, sur le fait qu'elles n'oublient pas de cas limite et qu'elles soient faciles à étendre ;
- analyser certains aspects du code qui ne correspondent pas à un problème précis mais à des notions plus larges, telles que la duplication de code ou les accès à la base de données.

Pour l'instant, nous avons travaillé sur le point précis des duplications de code, avec SonarQube. L'outil propose, parmi ses règles par défaut, de trouver les blocs de code dupliqués dans le code. La capture 16 montre une partie des résultats : l'application auditée est séparée en deux parties, mais la partie « interface »ici est la plus concernée par les duplications.

Le récapitulatif sur la partie qualité va s'arrêter ici, car l'audit n'est pas assez avancé pour en parler plus avant. La prochaine étape de notre audit, néanmoins, va être de parcourir le code source à l'aide du récapitulatif des commits ayant corrigé les *root causes* et de donner un avis sur les éventuels oublis ou améliorations possibles.

Graphique 16 – Représentation graphique des relevés de code dupliqué par SonarQube



8 Conclusion & avenir

Ce stage aura été formateur de bien des manières et je vais tenter de mettre en avant les principales ici.

Découvrir l'intégration continue et ses applications à la sécurité, tout d'abord : c'est une notion dont j'avais entendu parler, et qui revenait régulièrement dans les offres de stages que j'ai parcourues, mais qui restait flou à mes yeux et qui s'est avérée, finalement, très intéressante à mettre en pratique.

L'intégration continue est une branche encore jeune de l'informatique en entreprise, et, semble-t-il, porteuse d'avenir, mais c'est aussi un domaine intéressant du point de vue de la sécurité tant il offre d'opportunités, que ce soit comme ce que j'ai pu faire automatiser des analyses de sécurité ou d'une manière générale auditer et améliorer le processus de CI en lui-même d'un point de vue cybersécurité (après tout, il s'agit d'accéder au net, éventuellement de télécharger ou téléverser des documents voire déployer une application, ou se connecter à distance à des machines... tout cela peut présenter des failles de sécurité et requérir des experts en la matière pour les éviter/corriger).

Me déplacer en personne chez un client, mener un audit, ensuite, a été nouvelle. J'ai pu participer à ce projet de son début (la planification de l'intervention, des différentes réunions avec le client, la répartition des tâches dans notre équipe) à sa fin (la rédaction des livrables et de la documentation) tout en intervenant évidemment sur la partie technique, les analyses à proprement parler.

Se plonger momentanément dans la façon de travailler, les méthodologies d'une entreprise dont on n'a pas l'habitude et dont les méthodes de travail et les besoins sont radicalement différents de ceux d'Alter Frame auxquels je m'étais habitué était d'une certaine manière déroutant, et d'une autre intéressant.

En fin de compte le seul bémol que je vois est de ne pas avoir eu le temps de mener aussi loin que je l'aurais aimé la partie CI du stage, mais celle-ci ne sera pas perdue dans la mesure où le code reste disponible pour des interventions futures, correctement commenté et documenté.

Pour rester sur la partie CI, et mettre un point final à ce rapport, on m'a proposé de rejoindre Alter Defense & Security (voir la figure 1 pour rappel) après la fin de mon stage, comme consultant en sécurité informatique, et il se trouve que mon tout premier contrat va être en lien avec l'intégration continue puisqu'il va s'agir de

mettre en place et sécuriser l'intégration et le déploiement continus d'une solution de threat intelligence[35] sur le Cloud pour un client d'Alter Defense.

9 Crédits

- Le template utilisé pour la mise en forme de ce document est l'oeuvre d'Andrew Hobbs, disponible ici sous licence Creative Commons Attribution 4.0 International.
- Le graphique 8 utilise une icône faite par Situ Herrera disponible sur www.flaticon.com sous licence Attribution 3.0 Unported.

Table des figures

Listings

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. 4).

[4] CNIL, éd. *Textes officiels européens protection des données*. URL : https://www.cnil.fr/fr/textes-officiels-europeens-protection-donnees (cf. p. 5).

[5] The Open Web Application Security Project (OWASP), éd. *The ZAP API*. URL : https://github.com/zaproxy/zaproxy/wiki/ApiDetails (cf. p. 6).

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

[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. 6).

[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] Apache Software FOUNDATION, éd. *Using ActiveMQ > Hello World*. URL : http://activemq.apache.org/hello-world.html (cf. p. 21).

[33] NULLSOFT, éd. *nullsoft scriptable install system*. Nullsoft ne donne plus signe de vie depuis 2014 mais le projet est toujours maintenu par des développeurs indépendants. URL : http://nsis.sourceforge.net/Main_Page (cf. p. 24).

[34] NULLSOFT, éd. *NSIS Users Manual*. URL : http://nsis.sourceforge.net/Docs/ (cf. p. 24).

[35] WIKIPEDIA, éd. *Threat Intelligence*. URL : https://fr.wikipedia.org/wiki/Threat_Intelligence (cf. p. 33).