Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences
Western Switzerland

FILIÈRE INFORMATIQUE DE GESTION
TRAVAIL DE BACHELOR

Tests d'intrusions IT génériques et unifiés en Django/Python

Août 2019

Réalisé par Roque Daniel
Professeur responsable Barmaz Xavier
Document déposé le 7 août 2019

www.hevs.ch

RÉSUMÉ

Ce projet d'intégration a pour objectif de concevoir une application web en Django Python permettant d'exécuter des tests d'intrusions de type réseau, applicatif et social engineering. Pour atteindre cette finalité, les fonctionnalités et outils de trois anciens travaux de bachelor dans la même thématique ont été assemblés au sein d'une solution unifiée. En vue de ce qui précède, un état de l'art sur les frameworks open source et gratuit a été mené au préalable. Le résultat de cette investigation a donné suite à une analyse qui a permis au moyen d'une matrice décisionnelle d'évaluer les divers frameworks découverts. L'analyse a confirmé qu'aucun de ces frameworks ne permet d'exécuter les trois catégories de tests d'intrusions. Par conséquent, une stratégie d'intégration a été définie pour intégrer les trois anciens travaux de Bachelor. Les spécifications de chacun de ces travaux ont été prises en considération dans la base de cette stratégie. Finalement, l'approche d'intégration a permis de consolider la base de données et les fonctionnalités de chaque travail tout en y apportant des améliorations. De plus, tous les outils de tests d'intrusion applicatifs et social engineering ont été intégrés.

Mots-clés :	Django Python, application web, test d'intrusions, orchestration sécurité

AVANT-PROPOS

Dans un article du journal suisse Le Temps, Anouch Seydtaghia présente les enjeux de la sécurité informatique au vu de « la croissance exponentielle des données » provenant de divers systèmes et appareils, tels que les smartphones, les plateformes numériques ou encore les objets connectés (Seydtaghia, 2019). La sécurité est un thème primordial pour les entreprises qui doivent se prémunir contre les cyberattaques et cybercriminels.

La meilleure façon de prévenir des failles de sécurité d'un système ou d'un groupe de personnes est de les mettre à l'épreuve avec des tests d'intrusions. Il existe divers outils open source et gratuit pour procéder à ces tests. Cependant, chacun de ces outils permet de réaliser un type de test d'intrusion spécifique. L'idéal serait d'avoir un framework qui réunisse des outils d'intrusions de différents types (réseau, applicatif, et social-engineering).

Ce travail a pour objectif de rechercher un framework de ce type ou, le cas échéant, de le développer en intégrant trois anciens travaux de Bachelor qui contiennent les outils souhaités. Le thème de ce travail m'a séduit, car mes connaissances dans le domaine du web sont restreintes et que le framework Django utilise le langage Python, un langage que je souhaite apprendre. De plus, la sécurité est un sujet qui prend de l'importance dans mon activité professionnelle.

STRUCTURE DE LA CLÉ USB

En annexe de ce rapport, vous trouverez une clé USB contenant les fichiers suivants :

- 00_CAHIER_DES_CHARGES.pdf
- 02_GUIDE_INSTALLATION.pdf
- 03_GUIDE_DEVELOPPEUR.pdf
- 04_RAPPORT_DES_HEURES.pdf
- 05_POSTER.ppt

REMERCIEMENTS

J'adresse tous mes remerciements pour le soutien que toutes les personnes de mon entourage m'ont donné durant ce travail.

Je tiens à remercier chaleureusement Xavier Barmaz, professeur à la Haute École Spécialisée de Suisse Occidentale (HES-SO), pour m'avoir suivi et apporté un soutien irréprochable. Sa disponibilité et son sens critique ont été d'une aide précieuse.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS

AJAX	Asynchronous JavaScript and XML
BPMN	Business Process Model and Notation
CMS	Content Management System
CSS	Cascading Style Sheets
EAI	Enterprise Application Integration
HTTP	Hyper Text Transfer Protocol
MSA	Microservice Architecture
MVC	Model View Controller
MVT	Model View Template
NIST	National Institute of Standards and Technology
Nmap	Network mapper
ORM	Object Relational Mapping
OSSTMM	Open Source Security Testing Methodology Manual
OWASP	Open Web Application Security Project
RSA	Rivest Shamir Adlemann
SGC	Système de gestion de contenu
SOA	Services Oriented Architecture
TIDoS	Offensive Web Application Penetration Testing
URL	Uniform Resource Locator
WAF	Web Application Firewall
ZAP	Zed Attack Proxy

INTRODUCTION

Le RSA Security est l'un des leaders mondiaux dans le domaine de la cybersécurité et la gestion des risques numériques. L'entreprise a été fondée par les trois inventeurs du RSA, l'un des algorithmes de chiffrement les plus connus. (RSA Security, 2019). En 2011, la RSA a été la cible d'un pirate informatique qui est parvenu à subtiliser des informations confidentielles relatives à un produit[1] utilisé par 40 millions d'entreprises pour protéger leur propre infrastructure. L'attaque a été exécutée en plusieurs étapes et a commencé par l'envoi d'un mail d'hameçonnage[2] avec comme sujet « 2011 Recruitment Plan ». Le mail contenait un fichier Excel infecté par un logiciel malveillant qui utilisait une faille inconnue du logiciel Adobe's Flash. Grâce à une combinaison d'ingénierie sociale et d'un tour de force sur le logiciel d'Adobe, le cyberpirate a volé des informations qui lui permettaient d'attaquer d'autres entreprises comme Lockheed Martin[3]. (Richmond, 2011)

Les implications d'une cyberattaque peuvent être dévastatrices pour une entreprise. Mais il est possible d'éviter ce genre d'évènements à l'aide de prévention et en réalisant des audits de sécurité qui incluent des tests d'intrusions (pentest[4]). Le but de ce travail consiste à intégrer trois anciens travaux de Bachelor, afin de concevoir une application web qui permet d'exécuter des tests de pénétrations de types réseau, applicatif et social engineering. L'hypothèse qu'un framework open source et gratuit englobant des outils de pentests des trois catégories mentionnées existe, doit être vérifiée.

Dans cet ordre d'idée, un état de l'art sera réalisé. Le bilan de ces recherches sera analysé en vue de choisir les frameworks à intégrer. Ensuite, certains concepts fondamentaux et techniques concernant le framework Django Python seront expliqués, afin d'introduire le concept d'intégration appliqué durant la phase de développement. Chaque étape de la stratégie d'intégration sera expliquée en détail. Finalement, le résultat et les améliorations de ce travail d'intégration seront présentés.

[1] Le produit concerné est SecureID qui est une technologie qui permet une authentification à deux facteurs.

[2] Le terme hameçonnage est la traduction du terme anglais plus populaire « phising ».

[3] « Lockheed Martin est la première entreprise américaine et mondiale de défense et de sécurité » (Wikipedia, 2019d)

[4] Pentest est la contraction des mots anglais « penetration » et « test ».

1. État de l'art

Ce travail se situe dans le contexte de la sécurité informatique ou plus exactement l'intervention technique appelée test de pénétration ou pentest, en anglais (ORSYS, 2019). Ces techniques visent à identifier des vulnérabilités et les risques d'un environnement informatique cible. À l'aide de divers outils, un pentester[5] va simuler les attaques d'un potentiel hacker [6]. Ces tests sont essentiellement accomplis sur des systèmes d'information. Mais il est aussi possible de subtiliser des données en utilisant l'ingénierie sociale ou « social engineering »[7] (tutorialspoint.com, 2019).

En premier lieu, nous allons aborder quelques concepts théoriques. En second lieu, nous allons présenter les frameworks résultant de nos recherches, ainsi que le sujet des anciens travaux de Bachelor. Finalement, les outils de pentests utilisés dans les travaux précédents seront mentionnés.

1.1. Le hacking

Le hacking est un domaine pluridisciplinaire qui a pour but de détourner quelque chose de son usage habituel. En informatique, c'est l'application de diverses techniques d'attaques produite par un hacker qui aspire à subtiliser des données, modifier des droits d'utilisateurs, provoquer la surcharge d'un système ou encore falsifier l'identité d'une personne pour citer quelques exemples (Bancal et al., 2017).

1.2. Type de hacker

Il existe plusieurs types de hackers. Trois types sont largement connus du public.

Le White Hat ou Hacker éthique est un professionnel en sécurité qui va attaquer des organisations par lesquelles il a été mandaté, dans le but de tester la sécurité de ces systèmes. Le Black Hat Hacker, connu sous le nom de cracker, est une personne malintentionnée qui s'introduit illégalement dans des systèmes d'information, dans un but mercantile. Le Grey Hat Hacker est la combinaison des deux autres types. C'est un professionnel de la sécurité, qui s'introduit dans des systèmes illégalement, mais sans forcément créer un dommage. Par ailleurs, il pourrait intentionnellement garder pour lui une faille utilisable ultérieurement (Baloch, s. d.).

[5] « Personne chargée de tester la sécurité d'un système d'information » (Wiktionnaire, 2017).

[6] Un hacker est une personne qui cherche à s'introduire frauduleusement dans un système informatique (Larousse, 2019).

[7] Le social engineering est l'art de manipuler des gens pour obtenir des informations confidentielles (Druide informatique inc., 2015)

1.3. Méthodologie de test de pénétrations

Les méthodologies servent avant tout à s'organiser et à travailler en équipe. Elles définissent le déroulement d'un test de pénétration. Il en existe plusieurs, comme l'Open Source Security Testing Methodology (OSSTMM), le National Institute of Standards and Technology (NIST) ou encore l'Open Web Application Security Project (OWASP) (Baloch, s. d.). Les étapes varient selon ces modèles, mais par souci de simplicité ces phases sont ici synthétisées :

1. La reconnaissance : Collecte de renseignements à l'aide de différentes techniques et outils.
2. L'énumération : Identification et évaluation des vulnérabilités.
3. L'exploitation : Les vulnérabilités trouvées en amont sont exploitées.
4. L'escalade des privilèges : le pentester va chercher à s'implanter de manière durable.
5. La suppression des traces

(Ogma-sec, 2015)

1.4. Framework de pentest

Les solutions énumérées ci-dessous n'intègrent pas d'outils de type social engineering.

1.4.1. Faraday

Faraday est une plateforme collaborative de test de pénétration et de gestion de vulnérabilité développée par Infobyte SA. Cette société est basée à Buenos Aires en Argentine (Cyberpunk, 2018) (FaradaySEC, 2019).

Faraday est décrit sur le web comme un outil collaboratif, car il permet au moyen d'une console de suivre en temps réel les actions des autres utilisateurs. D'autre part, l'outil est open source et a été développé en Python.

C'est un framework[8] de gestion des risques de sécurité classés dans les outils de collecte d'informations par le site de Kali Linux. Il est disponible dans la dernière distribution Kali 2019.2 (Offensive Security, 2019b). L'application réalise des tests de pénétrations applicatifs et incorpore divers plug-ins, tels que Nmap[9], OpenVas, Metasploit et Qualys pour citer les plus populaires (Montilva, 2013/2019). Le résultat des outils est indexé et analysé par Faraday, afin de les présenter dans un workspace[10] (Cyberpunk, 2018). Le workspace et le tableau de bord sont illustrés ci-dessous (Figure 1).

[8] Le mot framework est un anglicisme qui signifie, cadre d'applications ou cadriciels (Druide informatique inc., 2018)

[9] Nmap est un scanner de ports utilisé pour la découverte du réseau et l'audit de sécurité (https://nmap.org/)

[10] Dans Faraday, le terme « workspace » désigne l'emplacement où l'on sauvegarde les informations collectées à partir des outils et commandes utilisés dans les tests de pénétration (https://github.com/infobyte/faraday/wiki/Workspaces).

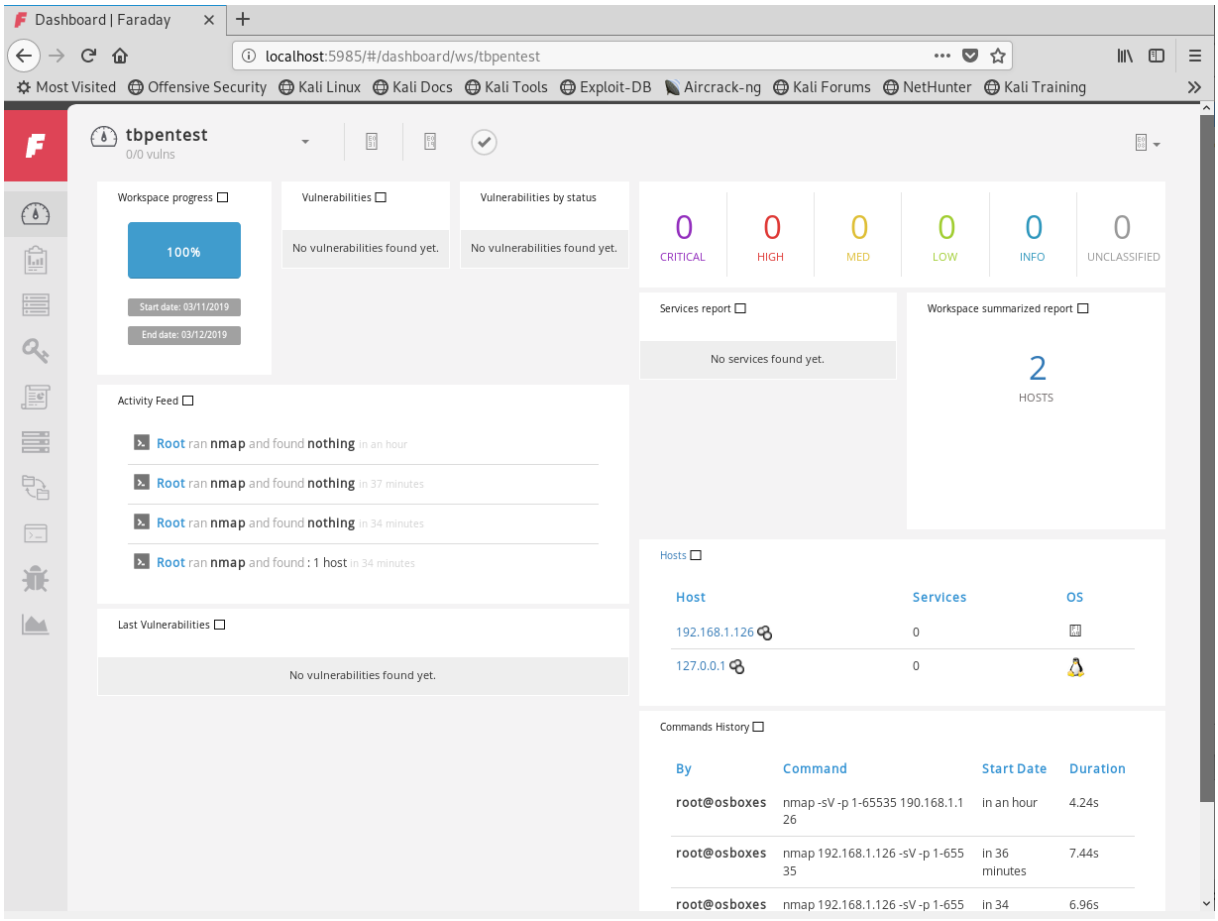


Figure 1 : Faraday - Dashboard (GUI)

Il existe trois versions de ce produit, la version « Community », « Professional » et « Corporate. La version « Community » est gratuite et intègre les fonctionnalités de base. Autrement dit, l'utilisation des plug-ins intégrés, l'accès à l'interface web, le statut des vulnérabilités identifiées sur une cible. Toutefois, certaines fonctionnalités sont bloquées. Par exemple, la création de rapport et les fonctionnalités de collaboration sont disponibles dans la version « Professional ». La version « Corporate » inclut l'analyse de données, l'importation de rapport ainsi que la comparaison de workspace.

Lien Github: https://github.com/infobyte/faraday

1.4.2. Pentest-tools

Pentest-tools est une application web développée par PentestTools SRL, une société basée en Roumanie.

Le site met à disposition 5 catégories d'outils : collecte d'informations d'un site web, test d'application web, test d'infrastructure réseau, scan de vulnérabilités, scan de requêtes Hyper Text Transfer Protocol (HTTP) et outils pratiques, tel que Whois[11] Lookup. Certaines des fonctionnalités mises en ligne font en réalité des appels à des outils gratuits et open sources tels que Nmap, OpenVas et OWASP ZAP, qui peuvent être utilisés en échange de crédits. Un total de 40 crédits est offert et renouvelé toutes les 24 heures.

Figure 2 : Pentest-Tools (SQL Injection)

L'exécution de chaque outil génère un rapport qui est présenté dans une fenêtre modale[12] que l'on peut réafficher plus tard. Il faut compter au minimum entre 10 à 50 crédits pour chaque utilisation.

[11] WHOIS est un service gratuit qui permet d'obtenir des informations sur le détenteur d'un domaine enregistré

[12] Dans une interface web, la fenêtre modale s'ouvre directement à l'intérieur de la fenêtre courante du navigateur.

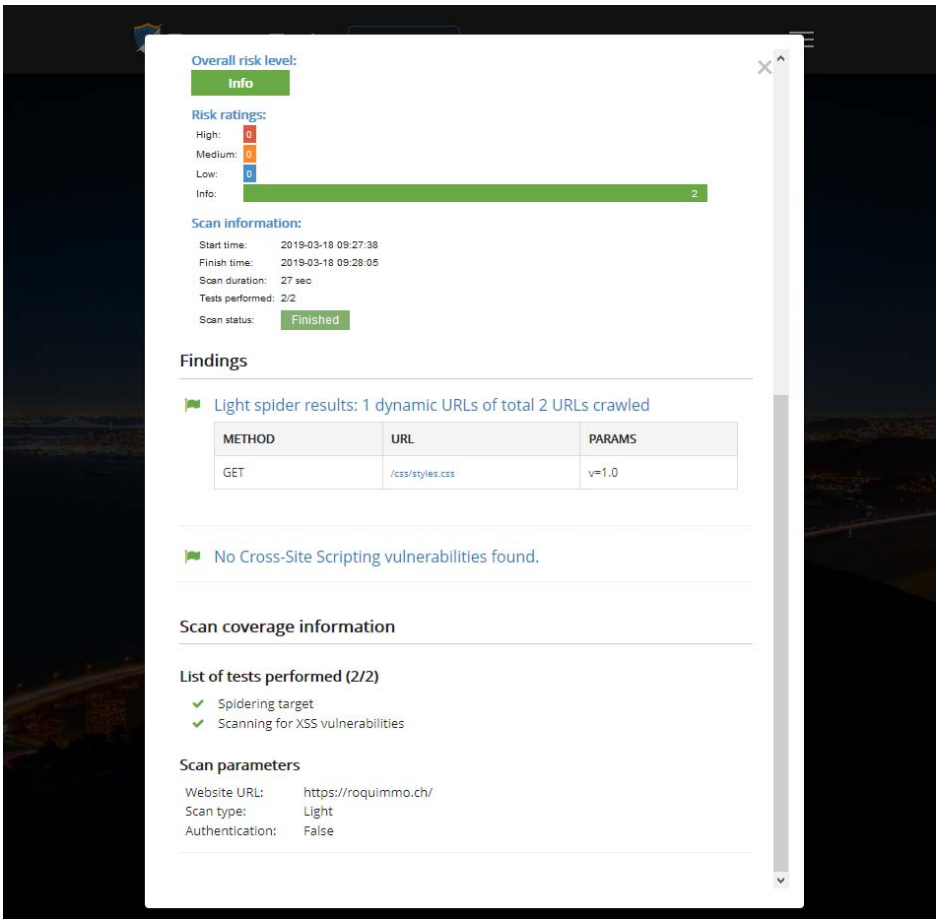


Figure 3 : Pentest-Tools (rapport)

En 2018, il existait encore une version gratuite appelée « Community » qui a été remplacée par les crédits offerts. La version « Pro Basic » contient 500 crédits valables durant 1 mois, en l'échange de 45 dollars américain. Elle comprend l'exportation des résultats de l'outil dans un rapport (voir Figure 3). Le plan « Pro Advanced » coûte $ 250 et inclut 3000 crédits valables pendant 1 année. Outre le nombre de crédits, des fonctionnalités supplémentaires sont ajoutées à cette version, comme la planification des scanners de vulnérabilités ou encore des rapports avancés. Pour finir, la version « Enterprise » a un nombre de crédits illimités pour une durée d'une année. Elle permet d'avoir un accès multi-utilisateur aux outils et la personnalisation des rapports[13]. Par exemple, en intégrant son propre logo. (PentestTools SRL, 2019)

Lien vers la plateforme : https://pentest-tools.com/home

[13] White label report (PenTestTools SRL, 2019)

1.4.3. PenTesters Framework

PenTesters framework a été développé en Python par David Kennedy, le fondateur de la compagnie TrustedSec basée aux États-Unis.

Le projet PenTesters Framework est open source et incorpore plusieurs modules qui peuvent être appelés en ligne de commande dans le terminal d'une distribution Linux. Ce sont principalement des modules pour faire des tests de pénétrations réseau et applicatifs.

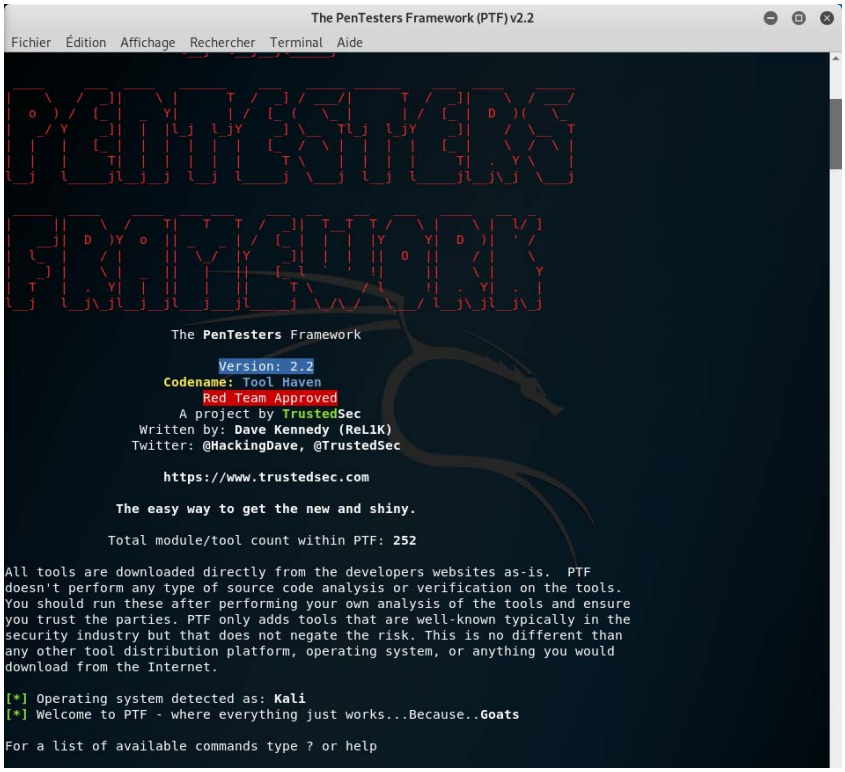


Figure 4 : Pentesters Framework - Accueil

Il est aussi possible de créer ses propres modules et de les intégrer. D'ailleurs l'auteur du projet encourage la communauté à intégrer de nouveaux modules (TrustedSec, 2019) .

Lien GithHub : https://github.com/trustedsec/ptf

1.4.4. Offensive Web Application Penetration Testing (TIDoS) Framework

TIDoS-Framework est un outil écrit en Python par un développeur anonyme portant le pseudonyme 0xInfection.

Le code est open source et disponible sur Github. Ce framework couvre une partie des étapes d'un test de pénétration applicati, c'est-à-dire la reconnaissance, l'énumération, l'exploitation et l'analyse des vulnérabilités. L'utilisation de TIDoS se fait en ligne de commande dans un terminal Linux.

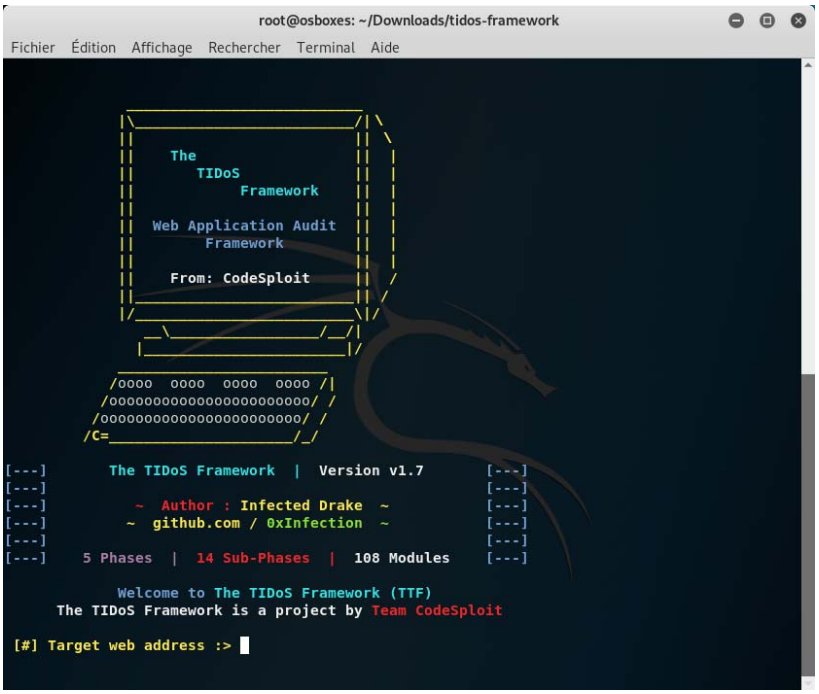


Figure 5 TIDoS Framework

La phase de reconnaissance contient plus de 50 modules, tels que Whois Lookup, recherche sur Google ou les réseaux sociaux. Le segment énumération contient lui 16 modules, qui permettent de scanner les ports d'un hôte, de reconnaître le système d'exploitation ou encore l'utilisation d'un crawler web, en français collecteur[14]. En ce qui concerne l'exploitation, c'est ShellShock [15] qui est utilisé. Mais cette partie est encore en phase de développement. (0xInfection, 2019)

Lien GitHub: https://github.com/0xInfection/TIDoS-Framework

[14] Un web crawler ou collecteur permet de collecter des ressources comme des pages web, images, vidéo, PDF, etc (Wikipedia, 2019e).

[15] Shellshock est une faille de sécurité découverte en 2014 qui permet d'exploiter des bugs dans l'interpréteur en ligne de commande des scripts de type Bash. Bash permet d'exécuter des commandes système dans Unix (Wikipedia, 2018c).

1.4.5. Exploit Pack

Exploit Pack est un framework programmé en Java et déployable sur les principaux systèmes d'exploitation (Mac, Windows, Linux). Le code source était mis à jour régulièrement et disponible sur Github jusqu'au 13 mars 2019 (Exploit Pack, 2019). Aucune information concernant cette suppression n'a été trouvée. Cependant, le logiciel est encore disponible sur le site officiel http://exploitpack.com/download.html. Par ailleurs, une copie du code source a été trouvée sur la plateforme GitLab à l'adresse https://gitlab.com/Scofield01/exploitpack.

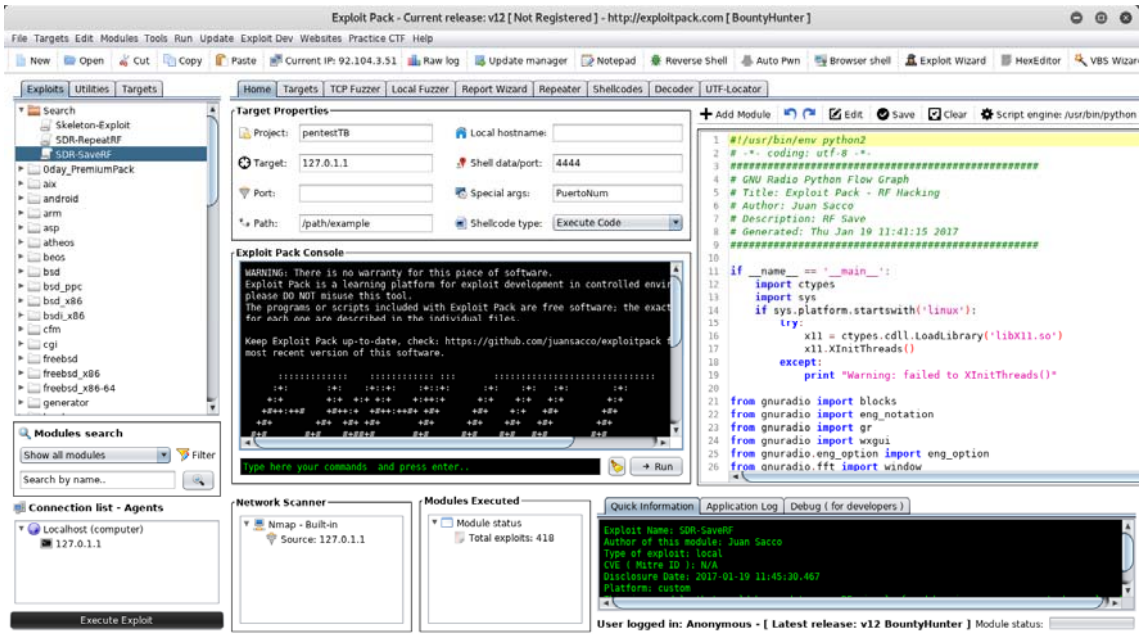


Figure 6 Exploit Pack dashboard

Exploit Pack est un outil contenant plus de 38'000 exploits[16] qui peut être utilisé lors d'une session de tests de pénétration. Il est possible de créer ses propres exploits en utilisant le langage Python dans un éditeur intégré au framework. « Un exploit est, dans le domaine de la sécurité informatique, un élément de programme permettant à un individu ou à un logiciel malveillant d'exploiter une faille de sécurité informatique dans un système informatique » (Wikipedia, 2019) et peut être télécharger depuis la base de données Exploit Database[17].

[16] Un exploit est, dans le domaine de la sécurité informatique, un élément de programme permettant à un individu ou à un logiciel malveillant d'exploiter une faille de sécurité informatique dans un système informatique (Wikipedia, 2019b)

[17] Exploit Database est une archive publique d'exploits. Voir le site https://www.exploit-db.com ou https://www.offensive-security.com/community-projects/the-exploit-database/.

Le framework utilise Nmap [18] pour identifier les ports et services d'un hôte distant. Il intègre également d'autres fonctionnalités comme le Fuzzing[19], l'attaque de site web, le contrôle à distance ainsi que la génération de rapports.

Trois versions d'Exploit Pack sont disponibles. La version « Community Edition » est très limitée en ce qui concerne les fonctionnalités mentionnées ci-dessus. Par ailleurs, elle intègre tout de même plus de 400 exploits auxquels l'utilisateur peut avoir recours. La licence « Professional Pack » est facturée mensuellement et contient plus de 39 000 exploits, dont les fonctionnalités de Fuzzing, scanner réseau, création de rapport, etc. Finalement, la différence principale entre la version « Professionnal Pack » et « Premium Pack » réside dans la durée de l'accès, qui est illimitée dans cette dernière. La version haut de gamme revient à 600 dollars et l'intermédiaire à 100 dollars. (Sacco, 2019)

1.4.6. PenBox

PenBox est un framework open source sous licence MIT[20]. Il a été écrit en python par deux développeurs, Fedy Wesleti et Mohamed Nour.

PenBox couvre différentes phases d'un test de pénétration : la reconnaissance, l'énumération, l'exploitation et l'élévation des privilèges. De plus, PenBox peut être uniquement installé sur une distribution Linux ou MacOS. (Wesleti, 2016/2019)

[18] Nmap ou Network Mapper est un outil qui permet de balayer un réseau afin de détecter les ports ouverts et identifier les services hébergés sur un système d'exploitation (Wikipedia, 2018b).

[19] Le fuzzing est une technique pour tester des logiciels en injectant des données aléatoires (Wikipedia, 2018a)

[20] La licence MIT une licence de logiciel libre et open source, non copyleft, permettant donc d'inclure des modifications sous d'autres licences, y compris non libres (Wikipedia, 2019).

Figure 7 Penbox menu

Les outils open source qu'utilisent PenBox ne sont pas installés dans le framework. Par ailleurs, lors de la sélection d'un outil, celui-ci est téléchargé et compilé afin d'être utilisable. Cela permet d'avoir systématiquement les outils à jour.

Lien GitHub : https://github.com/x3omdax/PenBox

1.4.7. DefectDojo

DefectDojo est une solution open source sous licence BSD-3-Claude développée en Django Python par trois ingénieurs en sécurité, Greg Anderson, Aaron Weaver et Matt Tesauro. C'est un Open Web Application Security Project (OWASP)[21], qui est une référence au niveau de la sécurité informatique. L'entreprise 10Security a été fondée pour offrir aux utilisateurs de DefectDojo un support commercial, et les aider à personnaliser le logiciel open source. (10Security, 2019)

[21] Open Web Application Security Project (OWASP) est une communauté en ligne travaillant sur la sécurité des applications Web. (« Open Web Application Security Project », 2018)

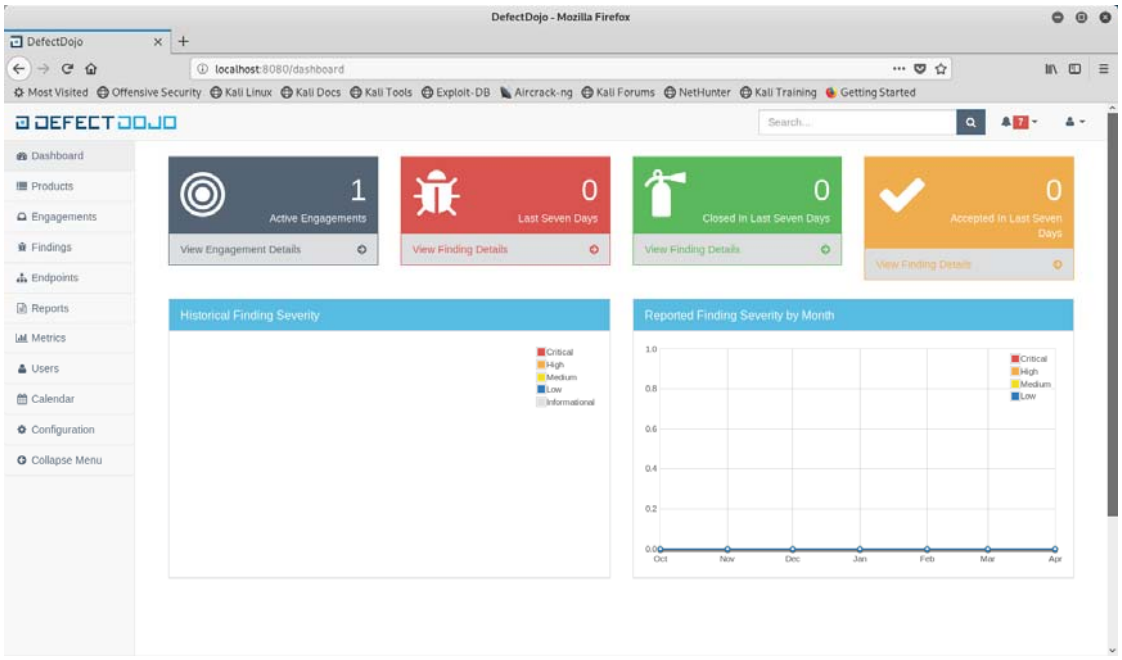


Figure 8 Defect Dojo Dashboard

L'application web a pour but de superviser les vulnérabilités d'un produit en important le résultat des tests réalisés avec des outils, tels que Nmap, Burp, Nessus, Nexpose, et cetera. Il est ainsi possible d'associer ces résultats au produit en cours d'analyse.

Dans un projet, on peut avoir plusieurs produits où les tests de pénétrations seront classés par environnement (développement, test, stable et production), date de réalisation, type de tests, ainsi que les vulnérabilités découvertes. En outre, le logiciel prend en charge les tests fonctionnels pour l'assurance qualité[22], les tests de sécurité et les tests d'interface de programmation applicative (API).

[22] Les tests fonctionnels du type Black Box sont utilisés dans le processus d'assurance qualité, afin d'évaluer la conformité d'une fonctionnalité selon les spécifications (« Functional testing », 2019).

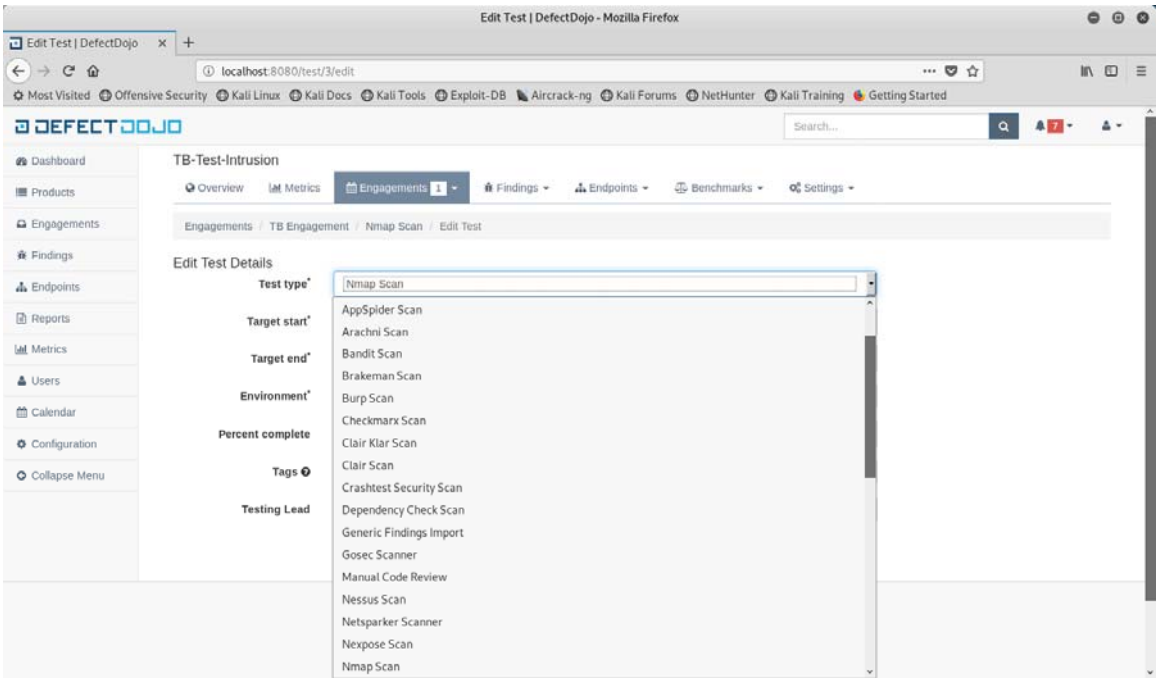


Figure 9 Defect Dojo engagements

En résumé, DefectDojo est un outil d'orchestration de sécurité qui de surcroit intègre une gestion d'utilisateurs permettant de donner un accès limité au framework à des utilisateurs externes. (Weaver, 2019)

Lien GitHub: https://github.com/DefectDojo/django-DefectDojo

1.4.8. SecureCodeBox

Secure Code Box est une chaîne d'outils modulaire open source sous licence Apache License 2. Cet outil a été créé et sponsorisé par iteratec GmbH une entreprise basée en Allemagne.

Le noyau de Secure Code Box fonctionne avec un moteur de workflow[23] en BPMN[24] appelé Camunda Modeler[25] qui fournit aussi l'interface utilisateur de l'outil. Les diagrammes qui décrivent les workflows des tests de pénétrations peuvent être modifiés par un développeur.

[23] Un moteur de workflow ou « process engine » en anglais est un logiciel qui permet d'exécuter des instances dans un workflow (Wikipedia, 2013). Un workflow est un flux de travail.

[24] BPMN est l'abréviation pour Business Process Model and Notation

[25] Camunda Modeler est une application open source et gratuite pour modéliser des workflows en BPMN. Voir le site de l'éditeur https://camunda.com/products/modeler/.

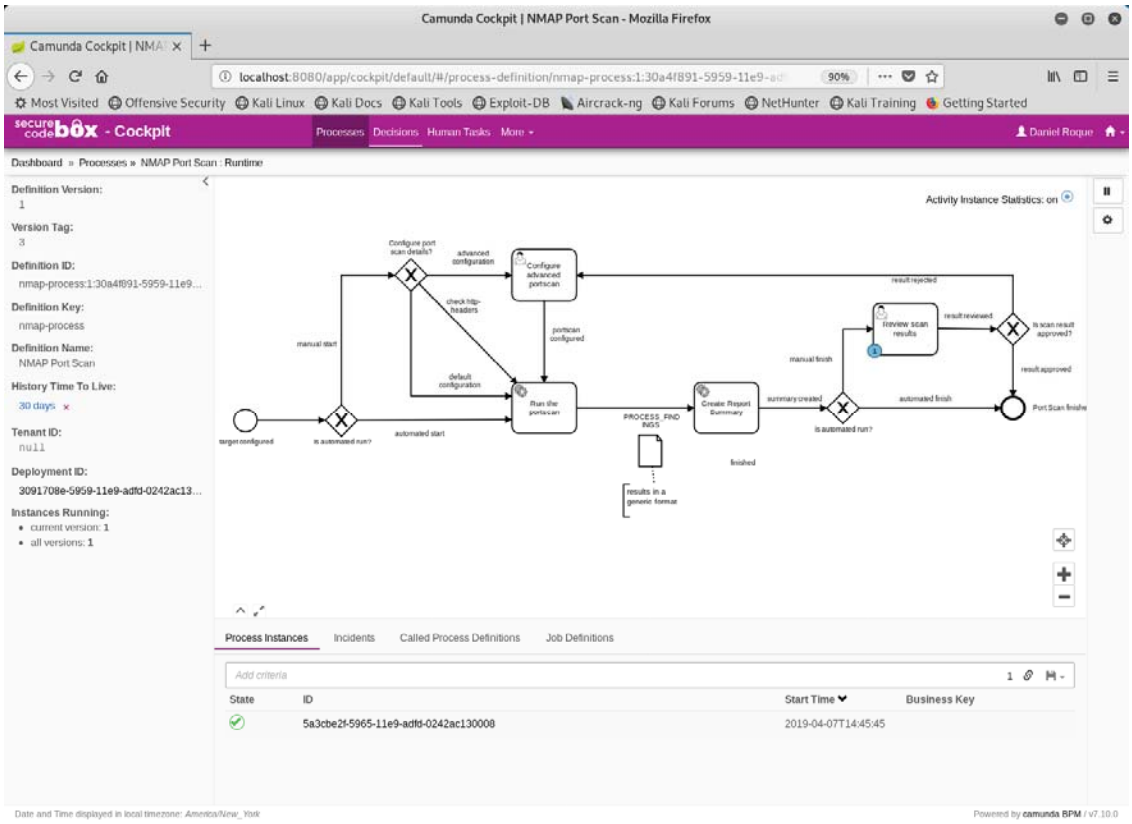


Figure 10 SecureCodeBox processus

L'architecture microservices[26] combinée avec Docker comme infrastructure, rend l'outil entièrement modulable, extensible et évolutif. Ce choix d'architecture facilite l'ajout ou la suppression des services utilisés au sein des flux de travail.

[26] L'architecture microservices est une approche pour concevoir des applications logicielles décomposées en plusieurs services indépendants, qui peuvent communiqués par le biais d'interfaces (API).

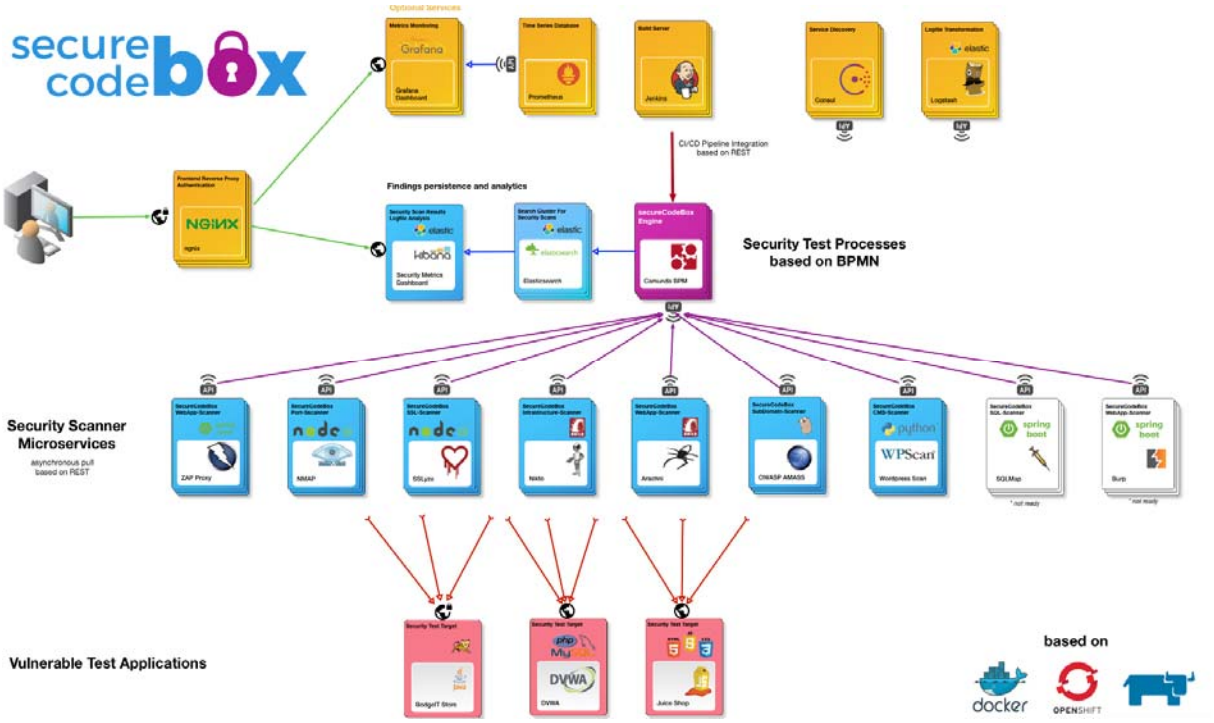


Figure 11 SecureCodeBox architecture

Secure Code Box comporte plusieurs scanners d'énumération et de vulnérabilités, tels que Nmap, Nikto, Arachni pour en citer quelques-uns utilisables après l'installation.

Lien GitHub : https://github.com/secureCodeBox/secureCodeBox

1.4.9. PatrOwl

PatrOwl est un framework d'orchestration de sécurité, open source et gratuit. Le code source écrit en Django Python est disponible sur GitHub et a été publié en octobre 2019 par Nicolas Mattiocco, un développeur français fondateur de la société GreenLock Advisory. GreenLock Advisory offre divers services de sécurité, ainsi qu'un support technique pour PatrOwl. Cette solution utilise une licence GPL (GNU Affero General Public License) v3.0. (Mattiocco, 2018/2019)

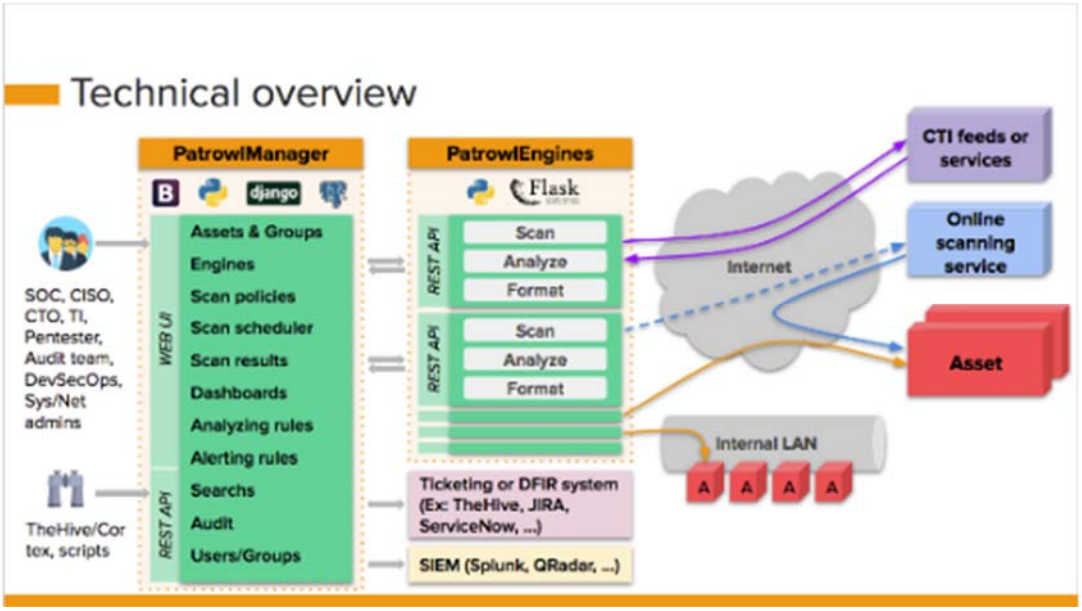


Figure 12 PatrOwl architecture

L'architecture du framework, illustrée dans à la figure ci-dessus (figure 14), est constituée du PatrowlManager et du PatrowlEngines. Le PatrowlEngines est le cœur du software et intègre les outils de tests de pénétrations, appelé « Engine ». L'enveloppe de chaque Engine, par exemple l'outil Nmap, permet l'exécution de l'outil et la mise en forme du résultat. Chaque Engine est lancé, depuis un container docker qui fait office d'infrastructure. Le PatrowlManager intègre l'interface web qui fait des appels aux API contenus dans le PatrowlEngines. (Mattiocco, 2018/2019)

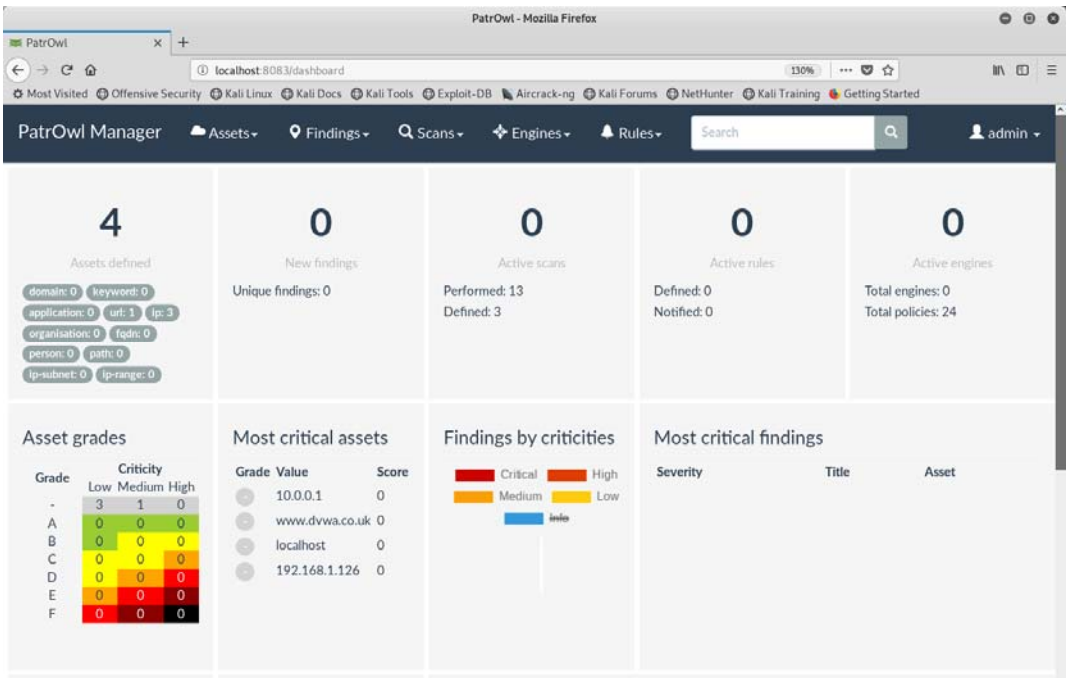


Figure 13 PatrOwl Dashboard

Dans le tableau de bord, on peut accéder aux ressources (Assets) à analyser, ainsi qu'à la liste des vulnérabilités découvertes. L'exécution des scanners, l'ajout des Engines et la gestion des utilisateurs se font sur l'interface utilisateur.

Ce système d'orchestration de sécurité intègre diverses fonctionnalités et outils. Il est aussi possible de coupler PatrOwl avec des systèmes comme Jira Software, pour la gestion de projets. Ce framework n'intègre pas d'outils pour réaliser des tests de pénétration de type social engineering, mais au vu de l'architecture, ils pourraient être ajoutés comme un « Engine ». (Mattiocco, 2019)

Lien GitHub : https://github.com/Patrowl

1.5. Rappel des anciens travaux de Bachelor

Dans cette partie nous allons brièvement présenter les travaux de Bachelor des années précédentes. Chacune des applications développées dans ces travaux est spécialisée dans un type de tests de pénétration. Les outils sélectionnés dans ces travaux sont par conséquent focalisés soit sur des tests de pénétration réseau, applicatifs ou de type social engineering.

1.5.1. Framework générique pour les tests d'intrusion (2016)

Le projet « Framework générique pour les tests d'intrusion » a été développé en PHP avec le framework Symfony en 2016. Le travail conçu par Siméon Bobylev porte sur des tests d'intrusions réseau, qui se basent sur l'appel de scanners de découverte réseau (nmap, amap), de scanners de vulnérabilités (Nessus, OpenVas) ainsi qu'une base de données contenant des exploits (www.cvedetails.com). La plupart des outils utilisés sont disponibles dans la distribution Kali Linux, qui est un prérequis à l'installation du projet .

Le framework permet d'identifier des ports ouverts sur une machine distante et déterminer les services qui sont actifs, afin de les exploiter à l'aide de faille existante. Les outils utilisés forment une chaîne où les résultats de chaque étape sont agrégés puis utilisés dans l'étape suivante. Finalement, un rapport peut être généré à partir des résultats des tests de pénétrations réalisés par l'utilisateur (Bobylev, 2016).

Par souci de simplicité, nous nommerons ce projet tout au long de ce travail « Framework Réseau ».

1.5.2. Framework générique et unifié pour les tests d'intrusion applicatifs (2017)

Ce travail de Bachelor a été écrit en Django/Python par Michel Lopez en 2017. C'est une application web qui utilise des outils de pénétration open source et gratuits. Le but étant de découvrir des vulnérabilités d'applications web. Kali Linux est un prérequis à l'installation de ce projet.

Au sein de l'application web, les tests de pénétration sont organisés par projet et peuvent être exécutés depuis un tableau de bord. Le processus d'exécution des outils est inspiré par la méthodologie définie par l'OWASP. Cela implique une étape de reconnaissance, d'énumération,

d'exploitation et finalement la génération d'un rapport contenant les résultats des tests réalisés. L'application web est sécurisée par un système d'authentification (Lopez, 2017).

Par souci de simplicité, nous nommerons ce projet tout au long de ce travail « Framework Applicatif ».

1.5.3. Framework générique et unifié de social (2017)

En 2017, Dany Marques a développé une application en Django/Python permettant de préparer et d'exécuter des tests de sécurité de type social engineering.

L'application web intègre deux outils, un formulaire pour rédiger des mails dans le but de conduire une campagne de phishing ainsi qu'un outil « d'Harvesting ». L'Harvesting est la technique utilisée pour cloner un site web, afin de subtiliser les informations confidentielles d'authentification d'un utilisateur. Les résultats des attaques menées sont ensuite résumés dans un rapport. Les tests de pénétration sont regroupés dans des projets liés à une entreprise. En outre, l'application est protégée par un système d'authentification (Marques, 2017).

Par souci de simplicité, nous nommerons ce projet tout au long de ce travail « Framework Social Engineering ».

1.6. Outils de tests d'intrusions

Cette partie de l'état de l'art tâche de récapituler les outils choisis dans les anciens travaux de Bachelor. Dans la mesure où le présent travail est axé sur de l'intégration, il ne sera pas nécessaire de sélectionner les outils, car ils ont été évalués et choisis au préalable. Pour avoir une appréciation complète et les raisons qui ont motivé le choix de ces outils, il faut se référer aux anciens travaux de Bachelor.

1.6.1. Nmap

Comme l'avaient déjà relevé Siméon Bobylev et Michel Lopez, Nmap[27] est l'outil de découverte réseau le plus populaire en ce qui concerne l'étape d'énumération dans le contexte des tests de pénétration. Gordon Lyon, aussi connu sous le nom de Fyodor, est l'auteur de ce logiciel libre et gratuit. Les Ethicals hackers et les administrateurs réseau apprécie l'usage de Nmap pour détecter les hôtes sur un réseau, les ports ouverts et les services subjacents disponibles ou encore le système d'exploitation installé sur une machine distante (Paul, 2019). Cet outil a été utilisé dans le travail de Bachelor Framework Applicatif et Framework Réseau.

1.6.2. Whatweb

Le scanner Whatweb a été développé par Andrew Horton et Brendan Coles à l'aide de plusieurs contributeurs. C'est un outil open source et gratuit. Il est utilisé dans les tests de pénétration de type applicatif. Il est utile pour récolter les technologies utilisées par un site web, telles que le Système de gestion de contenu (SGC ou CMS), les modules du framework, des erreurs SQL (Horton, s. d.) (Juned Ahmed Ansari, 2015). L'outil est installé sur la distribution Kali Linux et a été utilisé dans travail de Bachelor Framework Applicatif.

1.6.3. Amap

Amap est un scanner pour réaliser des tests pénétrations de type réseau. Il a été conçu par deux auteurs connus sous les pseudonymes van Hauser et DJ RevMoon. Amap arrive à identifier des services sur des ports spécifiques, sans que ce soit le port par défaut du service. Par souci de sécurité, un administrateur réseau pourrait cacher un service ou une application sous un autre port. Amap est l'outil idéal pour découvrir ce genre d'ingéniosité. (Offensive Security, 2019a). Cet outil a été utilisé dans le travail de Siméon Bobylev.

[27] Nmap est l'abréviation de network mapper.

1.6.4. Whafwoof

Cet outil, qui est utilisé à la phase de reconnaissance d'un pentest, permet d'identifier un firewall applicatif (WAF[28]). Cet outil a été intégré dans la solution du présent travail.

1.6.5. Whois

Whois un service qui permet d'obtenir des informations concernant la personne titulaire d'un nom de domaine.

1.6.6. Nikto

Nikto est un logiciel open source écrit en PERL par Chris Sullo. Cet outil est un scanner de serveur web, qui réalise de multiples tests permettant de détecter de mauvaises configurations, des fichiers à risque contenant des données sensibles ou des problèmes spécifiques à la version du serveur web. Les vulnérabilités trouvées par Nitko peuvent être exportées dans différents formats. Cet outil a été utilisé dans le travail Framework Réseau (Halton, Weaver, Ansari, Kotipalli, & Imran, 2017).

1.6.7. DIRBUSTER

DIRB ou DIRBUSTER est utile dans la phase de découverte pour les pentests applicatifs. C'est un scanner de contenu permettant de découvrir des fichiers ou objets web cachés. L'outil utilise une base de données contenant des attaques de serveur web connues, sur laquelle il se repose pour analyser les réponses retournées par le serveur web qu'il « interroge » (Ch & el, 2018).

1.6.8. Arachni

Arachni est un scanner développé en Ruby. Il permet d'évaluer les risques de vulnérabilités d'une application web à la phase d'énumération (Sarosys LLC, 2019). Cet outil a été intégré dans le Framework Réseau

1.6.9. ZAP

Zed Attack Proxy (ZAP) est un projet OWASP écrit en Java. C'est un outil développé et maintenu par une grande communauté qui permet d'identifier des vulnérabilités d'une application web. Il est utilisé à la phase d'énumération lors de l'analyse des vulnérabilités (OWASP, s. d.). L'outil a été utilisé dans le travail Framework Applicatif.

[28] WAF désigne Web Application Firewall

1.6.10. Nessus

Nessus est un scanner de vulnérabilité conçu par Renaud Deraison en 1998 et mis à disposition gratuitement sur la toile. En 2005, la licence de Nessus est devenue propriétaire et appartient aujourd'hui à l'entreprise Tenable Network Securiy. Cet outil permet de scanner un réseau pour ensuite réaliser des attaques actives pour identifier des vulnérabilités (Institut d'électronique et d'informatique Gaspard-Monge, 2019). Cet outil a été utilisé dans le travail Framework Réseau

1.6.11. OpenVas

OpenVas est actuellement développé et maintenu par Greenbone Networks, mais l'outil est sous licence open source GNU GPL. Cet outil est utilisé pour détecter des vulnérabilités. OpenVAS est un fork[29] open source et gratuit de Nessus, contrairement à Nessus qui est sous licence propriétaire. Cet outil a été utilisé à la phase d'énumération dans le travail Framework Réseau (Thebaud, 2019)

1.6.12. XSSER

« The cross Site Scripting Framework" connu sous XSSER est aussi un projet OWASP. C'est un outil open source qui permet de détecter et profiter d'une faille de Cross-Site Scripting[30] (XSS) (xsser.03c8.net, s. d.)

1.6.13. SQLMAP

Cet outil permet à la phase d'exploitation d'utiliser les failles trouver sur un site web pour y injecter des bouts de requêtes SQL pour compromettre la sécurité du site (sqlmap.org, 2019).

1.6.14. Social Engineer Toolkit

Cet outil est exclusivement conçu pour les attaques de type social engineering. Il a été conçu par TrustedSec et intègre diverses fonctionnalités, telles que la copie de site web, l'infection de fichier, l'envoie de mail en masse, et cetera (TrustedSec, 2019). Cet outil a été utilisé dans le Framework Social Engineering.

[29] « Un fork est un nouveau logiciel créé à partir du code source d'un logiciel existant » (Wikipedia, 2019c).

[30] « Le cross-site scripting (abrégé XSS) est un type de faille de sécurité des sites web permettant d'injecter du contenu dans une page, provoquant ainsi des actions sur les navigateurs web visitant la page. » (Wikipedia, 2019a)

2. Analyse et choix

L'état de l'art a permis de recueillir suffisamment d'informations pour débuter une phase d'analyse, dans le but d'orienter la suite de ce travail d'intégration. Il est maintenant possible d'évaluer chaque framework, en vue de préserver ceux qui sont les plus conformes aux contraintes énoncées, d'une part dans le cahier des charges, d'autre part dans les spécifications de ce travail.

Le but est de sélectionner les projets qui sont les plus adéquats à l'intégration d'une solution générique qui regroupe les trois types de tests de pénétrations. Le choix des frameworks sera en adéquation avec nos connaissances et le but de ce projet.

Tout d'abord, nous allons énumérer les contraintes et spécifications à la base de ce de travail et les fonctionnalités désirées. L'échelle des valeurs et la pondération de certains critères ont été reprises des anciens travaux, en raison de leur cohérence avec les contraintes du présent travail. Ensuite, chaque solution sera évaluée à l'aide d'une note pondérée. Finalement, les « meilleurs » frameworks seront choisis pour être intégrés.

2.1. Critères des contraintes et spécifications

Les critères considérés dans cette partie sont issus de l'état de l'art et du cahier des charges.

Critères	**Définitions**
Langage	Le framework est développé en Django Python.
Open source	Le code source est ouvert.
Gratuité	Le framework est gratuit.
Type de tests	Le framework inclut des tests de pénétration réseau, applicatif et social engineering.
Modularité	De nouveaux outils peuvent aisément être ajoutés.
Structure	L'ensemble des tests de pénétration sont encapsulés dans un projet.
Rapport	Le framework permet de générer un rapport avec les résultats des pentests.
Historisation	Les résultats sont historisés et sauvegardés.
Authentification	La gestion des utilisateurs est intégrée.

Intégration	De prime abord, le framework est utilisable dans le contexte de ce travail d'intégration. C'est-à-dire, qu'il respecte les contraintes énoncées ci-dessus et il est envisageable de l'intégrer.

Tableau 1 Tableau des contraintes et spécifications

2.2. Échelle des valeurs et définitions

Échelle des valeurs	Échelle verbale ou définitions
0	Le framework ne correspond pas au critère.
1	Le framework correspond partiellement au critère.
2	Le framework correspond entièrement au critère.

Tableau 2 Tableau de l'échelle des valeurs

2.3. Matrice décisionnelle

Framework	Langage		Open source		Gratuité		Type de tests		Modularité		Structure		Rapport		Historisation		Authentification		Intégration		Total pondéré
Pondération	3		3		3		1		2		2		2		2		1		2		
Faraday	2	**6**	1	**3**	1	**3**	1	**1**	1	**2**	1	**2**	0	**0**	2	**4**	2	**2**	0	**0**	20
Pentest-tools	0	**0**	0	**0**	1	**3**	1	**1**	0	**0**	1	**2**	1	**2**	1	**2**	2	**2**	0	**0**	11
PenTesters Framework	1	**3**	2	**6**	2	**6**	1	**1**	2	**4**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	13
TIDoS Framework	1	**3**	2	**6**	2	**6**	1	**1**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	9
Exploit Pack	0	**0**	1	**3**	1	**3**	1	**1**	1	**2**	1	**2**	0	**0**	1	**2**	0	**0**	0	**0**	9
PenBox	1	**3**	2	**6**	2	**6**	1	**1**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	9
DefectDojo	2	**6**	2	**6**	2	**6**	2	**2**	1	**2**	2	**4**	2	**4**	2	**4**	2	**2**	2	**4**	34
SecureCodeBox	1	**3**	2	**6**	2	**6**	1	**1**	2	**4**	1	**2**	0	**0**	2	**4**	2	**2**	0	**0**	23
PatrOwl	2	**6**	2	**6**	2	**6**	1	**1**	2	**4**	1	**2**	0	**0**	2	**4**	2	**2**	1	**2**	28
TB Bobylev	0	**0**	2	**6**	2	**6**	1	**1**	2	**4**	1	**2**	2	**4**	2	**4**	0	**0**	1	**2**	23
TB Lopez	2	**6**	2	**6**	2	**6**	1	**1**	2	**4**	2	**4**	2	**4**	2	**4**	2	**2**	2	**4**	35
TB Marques	2	**6**	2	**6**	2	**6**	1	**1**	2	**4**	2	**4**	2	**4**	2	**4**	2	**2**	2	**4**	35

Tableau 3 Matrice décisionnelle

2.4. Synthèse des résultats

La matrice décisionnelle a permis d'évaluer et classer les frameworks, conformément aux contraintes du projet. Il est donné de constater que les anciens travaux de bachelors répondent aux contraintes et spécifications attendues. Ces résultats sont cohérents, car ces frameworks étaient partiellement tributaires des mêmes exigences que ce travail.

On observe tout de même une exception, en ce qui concerne le travail de Bobylev. En effet, ce travail n'a pas été développé en Django Python. De plus, il ne possède pas de système d'authentification et il n'est pas possible de grouper les tests au sein d'un projet ou d'une compagnie. Dès lors, la difficulté d'intégration de ce projet semble plus élevée. Cependant, les fonctionnalités dans ce projet sont indispensables pour atteindre le but de ce travail, donc il faut le prendre en considération.

L'analyse des résultats de Defect Dojo et PatrOwl permet de conclure que ces deux frameworks sont à peu de chose près équivalents aux anciens travaux de bachelor. L'état de l'art a permis de reconnaître leur potentiel en qualité d'orchestrateur de processus et outil de sécurité, c'est-à-dire qu'ils permettent de connecter des outils, afin de les intégrer dans des processus automatisés dont les résultats peuvent par la suite être soumis à une décision humaine. (Rapid7, 2019)

PatrOwl serait un candidat intéressant, mais il est nécessaire de l'écarter des options envisageables pour ce travail d'intégration pour plusieurs raisons. Premièrement, lors de son installation, l'ajout d'outil pour réaliser des tests de pénétrations n'a pas abouti. Deuxièmement, la structure et la complexité de ce projet Django vont au-delà de nos compétences. Troisièmement, PatrOwl fonctionne dans une infrastructure Docker qui est pour nous une technologie peu connue.

Tout comme PatrOwl, nous pouvons à peu de choses près soutenir les mêmes arguments en ce qui concerne la solution Defect Dojo. En effet, ce framework a aussi été développé en Django et il se base également sur une infrastructure Docker. Somme toute, il n'intègre pas d'outils de tests de pénétration. Cependant, il est possible d'importer dans DefectDojo le résultat d'outils de pentests.

Finalement, certaines des applications analysées ne remplissent pas les attentes minimales pour faire l'objet d'une intégration. À partir de cette évaluation, nous pouvons déduire que les projets favorisant l'intégration d'un framework global et unifié sont le travail de Bachelor de Siméon Bobylev « Framework Réseau », de Michel Lopez « Framework Applicatif » et Dany Marques « Framework Social Engineering ».

3. Développement Django

Le concept d'intégration expliqué dans le chapitre suivant comporte des notions techniques et fait référence à certains concepts du framework Django. Il est donc plus commode d'avoir au préalable une connaissance élémentaire à ce sujet.

3.1. Introduction Django

Django est un framework open source et gratuit écrit en Python. Il est utilisé pour la création de site ou d'applications web telles qu'Instagram, Spotify ou encore Dropbox (Yarbrough, 2019). Ce framework a été créé en 2003, lorsque des développeurs du journal américain Lawrence Journal-World ont commencé à développer des applications en Python. L'origine de Django est apparentée à la culture de la communauté Django Software Foundation qui maintient le framework. De fait, les concepteurs du framework étaient entourés d'écrivains. On peut constater cette influence dans l'excellente documentation mise en ligne et la structure du framework (Fournier, 2015).

3.2. Architecture de Django

L'architecture de Django s'inspire de l'architecture MVC (Model/View/Controller) déployée par la plupart des frameworks web. Son architecture est composée de trois blocs appelés MVT (Modèles/Vues/Templates). Le premier bloc (Modèles) est un ORM (Object Relational Mapping) qui met à disposition une API pour manipuler et accéder à une base de données, telle que MySQL, SQLite ou encore PostgreSQL. Le deuxième bloc (Vues) permet de traiter les requêtes HTTP du client et les actions à exécuter dans les modèles. La troisième partie (Templates) est un langage de template permettant de générer des pages HTML, XML ou autre format texte. (Fournier, 2015)

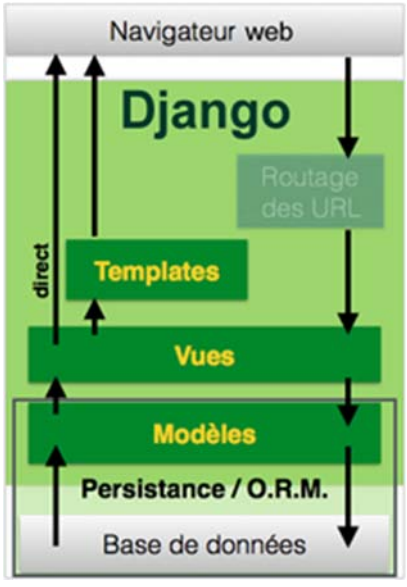


Figure 14 Model View Template (Fournier, 2015)

L'illustration ci-dessous résume les interactions entre ces blocs.

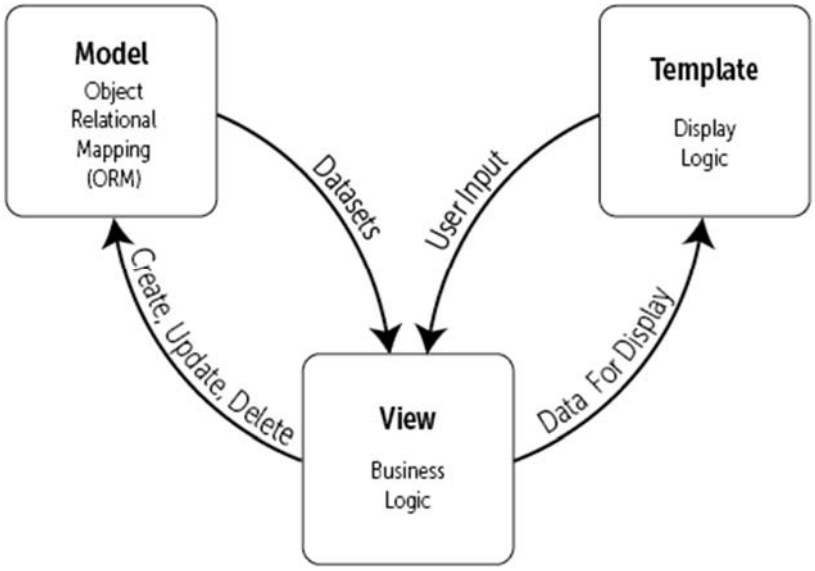


Figure 15 Diagramme MVT(The Django Book, 2019)

3.2.1. Structure d'un projet Django

L'explication de Franck Fournier dans son livre « Django – Industrialisez vos développements Python » explique correctement la structure d'un projet Django. Cependant, cette structure n'est pas univoque et peut varier d'un projet Django à un autre. Par exemple, il est possible dans la configuration de Django (settings.py) de définir un emplacement différent pour les « /templates » ou les fichiers statiques contenus dans le répertoire « /static ». Il faut considérer l'illustration ci-dessous, comme un modèle que l'on peut adapter. Pour plus de détails à ce sujet, il faut se référer à la documentation officielle à l'adresse https://docs.djangoproject.com/fr/2.2/intro/tutorial01/.

```
./projet            # Nom du projet
   /__init__.py     # (module python)
   /manage.py       # La commande Django de gestion du projet
   /projet                # Le répertoire du projet
         /settings.py # La configuration du projet
         /urls.py     # La configuration des URL du projet
         /wsgi.py     # Le script WSGI pour le déploiement
         /code_projet.py # Du code spécifique au projet
         /static      # Les fichiers statiques du projet
         /templates   # Les templates du projet
         /module/     # Un module Python du projet projet.module
         /__init__.py
   /db.sqlite3            # La base de données SQLite (si SQLite)
   /media     # L'emplacement où Django va mettre les téléchargements
   /static    # L'emplacement où Django va collecter les fichiers statiques
   /application1/          # Les applications du projet
   /application2/
      ...
   /applicationN/
```

Figure 16 Structure d'un projet Django (Fournier, 2015)

3.2.2. Application

Un projet Django est une somme de nombreuses applications, comme illustré à la figure 18. Une application Django est indépendante et entièrement réutilisable dans un autre projet. On peut la considérer comme un module du projet. Par exemple, un formulaire d'une application est réutilisable dans une autre application Django.

La structure d'une application est la suivante :

```
myApp
├── admin.py # Configuration de l'admin Django
├── apps.py # Registre de l'application contenant des méta données
├── __init__.py
├── migrations # Codes des migrations
│   └── __init__.py
├── models.py # Modèles de l'application
├── tests.py # Test unitaire
├── urls.py # Routage des requêtes HTTP aux vues spécifiqes à l'application
└── views.py # Vues de l'application
```

Figure 17 Structure d'une application Django

Le troisième fichier __init__.py, présenté à la figure 19, est une convention Python qui permet de déterminer que c'est un module. Une application peut contenir d'autres fichiers tels que (Spencer, 2017):

forms.py	Permets de stocker des formulaires de l'application
signals.py	Permets de stocker des « signals [31]»
api.py	utilisé pour faire appel à des API externes
	(Spencer, 2017)

[31] https://docs.djangoproject.com/en/1.11/topics/signals/

4. Concept d'intégration

Dans le contexte de l'architecture logicielle, l'intégration est considérée comme un système d'interaction ente plusieurs applications. C'est une base importante pour mener à bien l'évolution future d'une architecture (Rais, 2016). Il existe plusieurs styles d'architecture, comme le EAI (Enterprise Application Integration) ou le SOA (Services Oriented Architecture) qui permettaient de résoudre en partie deux aspects de l'évolution rapide des Systèmes d'Information (SI) : l'hétérogénéité et le changement (Thomas BAILET, 2012). Aujourd'hui il y a un style d'architecture qui émerge et s'étend dans les entreprises, c'est l'architecture microservices (MSA). En bref, dans une MSA une application est divisée en petite partie où chaque partie est indépendante, tant au niveau du développement que du déploiement. Le cycle de vie d'un microservice est donc indépendant de son écosystème (Schwartz, 2017).

4.1. Introduction

À partir de ce préambule, nous pouvons extrapoler et retenir une chose importante pour concevoir et intégrer les frameworks sélectionnés au moment de notre analyse. Les applications[32] Django à intégrer doivent être dans l'idéal indépendantes. Dans l'article « 0–100 in Django: Starting an app the right way » du site https://hackernoon.com, Jeremy Spencer explique comment déterminer s'il faut créer une application ou pas. Il répond ainsi à cette question : "If you can't explain what your app does in one sentence, it should be more than one app" (Spencer, 2017).

Maintenant, nous avons une base conceptuelle pour avancer et pouvons mettre en œuvre notre stratégie d'intégration pour les trois travaux de Bachelor. Tout d'abord, les fonctionnalités de chaque projet seront énumérées. Ensuite, la méthodologie de consolidation sera présentée où ces étapes seront détaillées et expliquées avec des exemples d'implémentation, ainsi que les difficultés rencontrées.

[32] Une application Django est une application web qui contient des fonctionnalités qui peuvent être réutilisées dans différents projets. Or, « un projet est un ensemble de réglages et d'applications pour un site Web particulier. » (Django Software Foundation, 2019a)

4.2. Fonctionnalités

Les trois frameworks sélectionnés contiennent chacune des fonctionnalités intéressantes pour concevoir un framework global intégrant des outils de tests de pénétrations. Le schéma ci-dessous illustre les fonctionnalités de chaque framework et le but recherché dans ce travail d'intégration :

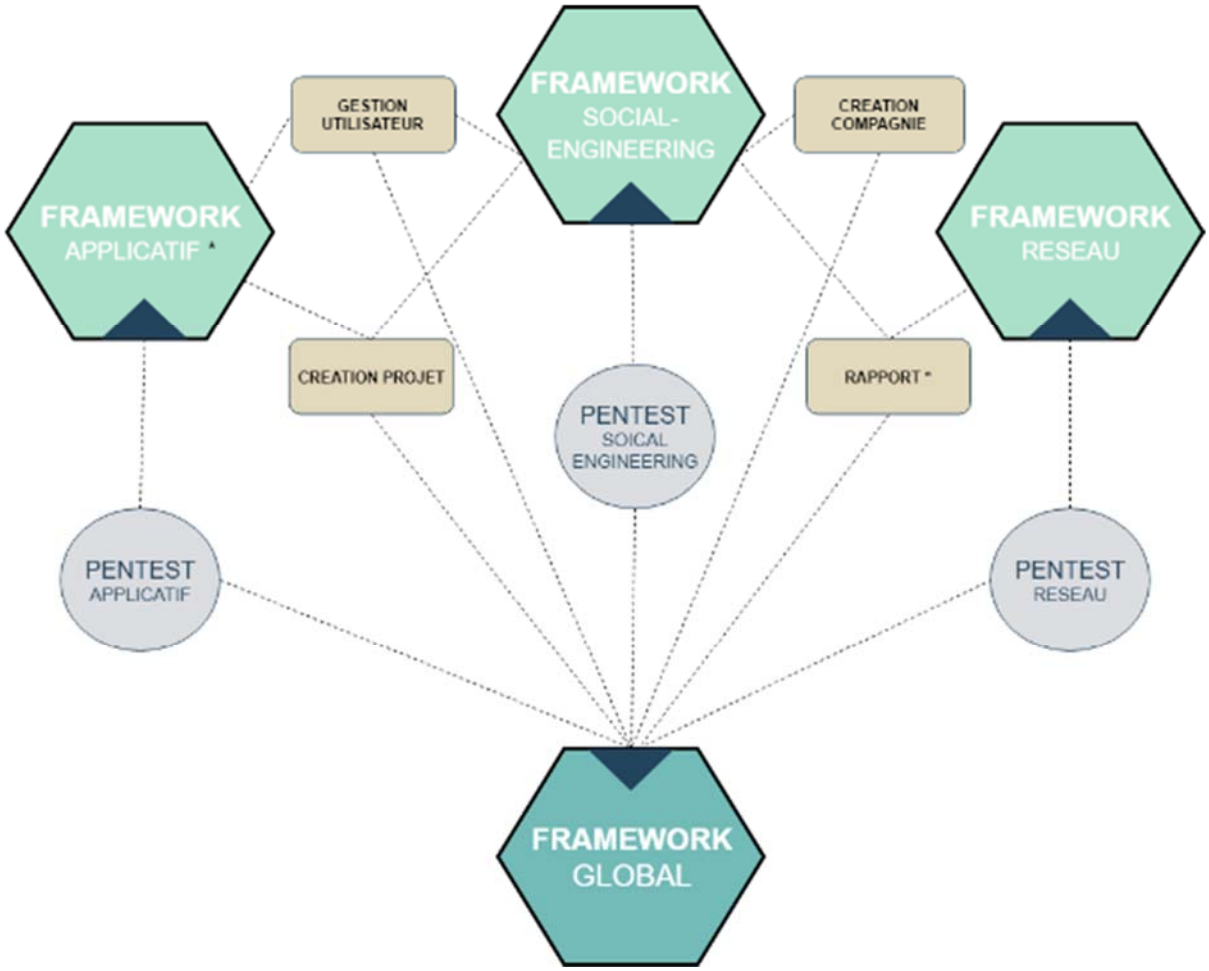


Figure 18 Fonctionnalités et spécifications des frameworks

Nous pouvons sur la base de cette esquisse détailler les spécifications attendues :

- Outils de pentests de type réseau, applicatif et social-engineering
- Base de données unifiée
- Service de mail unifié
- Gestion des utilisateurs unifiée
- Génération de rapport unifié
- Création de compagnies
- Création de projets contenants des pentests historisés

4.3. Méthodologie de consolidation

Les recherches ont permis de constater qu'il n'existait pas de méthodologie officielle pour fusionner plusieurs projets Django. Compte tenu de la modularité des applications d'un projet Django, nous pouvons utiliser une approche générale de consolidation.

Dans cette approche, nous allons fusionner les fichiers de configurations, ainsi que les dépendances dans le projet que l'on souhaite consolider. Après cette étape, nous allons éliminer graduellement les applications du projet à migrer dans le projet consolidé en évitant les redondances, tant au niveau du code source que de la base de données. Par conséquent les duplicatas seront éliminés. La logique est semblable pour les autres détails d'implémentation, comme les modèles, les vues et les templates. (Hillard, 2019)

4.4. Étapes de consolidation

Durant la phase de l'état de l'art, nous avons installé et testé chaque projet. Il faut préciser que ces projets étaient incompatibles avec la version de Kali Linux 2019.1. Il y a deux raisons principales à cela. La première raison d'incompatibilité est liée à la mise à jour des outils de pénétration dans la version actuelle de Kali Linux (2019.2). La deuxième découle des environnements virtuels utilisés lors du développement. En effet, ces environnements virtuels n'ont pas été figés ou n'existaient pas. Par conséquent, lors de l'installation des anciens projets la version des outils et des librairies n'était pas compatible avec le code source.

En ce qui concernant l'environnement virtuel, le gestionnaire de paquet Python PIP[33] (Package Installer for Python) permet de figer les dépendances d'un projet et de les enregistrer dans un fichier texte, nommé par convention « requirements.txt » (Python Software Foundation, 2019). Ceci dans le but de déployer un projet Python dans n'importe quel autre environnement, car l'environnement virtuel est isolé du système d'exploitation. C'est d'ailleurs son objectif principal.

4.4.1. Mise à niveau du projet « Framework Application »

Dans cette étape, nous avons débuté par la mise à jour de Django 1.10 à la version Django 1.11. Cette opération a résolu certaines erreurs, mais le projet ne démarrerait tout de même pas. Après diverses modifications, la version du projet était stable et pouvait être lancée. Cependant, la plupart des outils ne fonctionnaient pas, notamment WhatWeb, Whois, Dirbuster, Zap proxy et SQLMap.

Par la suite, nous avons procédé à un nettoyage du code. Par nettoyage, nous entendons la suppression du code redondant ou pas utilisé. Dans les illustrations à l'*annexe I*, on peut apercevoir

[33] Vous trouverez plus d'informations concernant le gestionnaire de paquet pour Python sur le site https://pypi.org/project/pip/ ou dans l'article rédigé sur le site https://realpython.com/what-is-pip/.

les répétions de certaines portions codes non utilisés qui ont été commentés à l'aide du symbole dièse (#).

4.4.2. Création du fichier de dépendances du projet « Framework Applicatif »

À cette étape, il était possible de créer un fichier contenant les dépendances du projet, afin de pouvoir le réinstaller dans un autre environnement de développement. Le fichier des dépendances est illustré à l'*annexe II*.

4.4.3. Mise à niveau du projet « Framework Social Engineering »

Le code source de ce projet n'a pas été développé dans un environnement virtuel. Par conséquent, il n'y avait pas de fichier contenant une liste de dépendances pour réinstaller le code source. Il a donc fallu procéder par étape pour obtenir un état stable.

Après diverses adaptations, le projet pouvait être lancé, mais certaines fonctionnalités ne se comportaient pas comme attendu. Par exemple, il était impossible de cloner un site web ou de tracer les emails envoyés dans une campagne de phishing. De plus, la structure du code et la logique d'implémentation étaient différentes du projet « Framework Applicatif ».

Finalement, nous avons décidé de reprendre l'idée de la gestion des projets d'intrusions par compagnie et d'intégrer l'application « mail » et l'application « credential_harvester », illustrée à l'a*nnexe III*. La base de données MySQL a aussi dû être migré en PostgreSQL.

4.4.4. Création d'un nouveau projet Django

À ce stade, nous avons initialisé un nouveau projet Django et débuté l'intégration du code du Framework Applicatif. Le choix de ce projet comme base d'architecture est le résultat de l'étude des codes sources des différents frameworks à intégrer. En effet, le système d'ajout d'outils de pentests mis en place dans le Framework Applicatif est aisément extensible.

En outre, cette démarche d'intégration a mis en lumière les points à améliorer, comme le nommage des méthodes et des URL. Ces modifications sont relatives aux conventions définies dans le PEP8 et seront détaillées dans la suite de ce rapport.

4.4.5. Intégration de l'application « company »

L'idée d'avoir une compagnie pour laquelle un auditeur réalise des tests de pénétration vient du Framework Social Engineering. Cette idée a ensuite évolué pour qu'une compagnie contienne des projets de trois types (réseau, applicatif, et social-engineering) et que ces projets puissent contenir plusieurs pentests.

4.4.6. Consolidation de l'application « mail »

Dans cette phase, l'application « mail » a été intégrée au nouveau projet. Déplacer une application dans Django implique plusieurs mises au point, comme l'installation de nouvelles dépendances, la modification des modèles, la modification des vues et des templates, ainsi que du fichier de réglages (settings.py) du projet.

Dans un pentest, la fonctionnalité d'envoi de « mail », dans le contexte du social engineering, est utilisée à la phase d'exploitation. Après son intégration, l'outil pouvait être piloté depuis un tableau de bord. Une illustration de l'intégration de l'outil se trouve à l'a*nnexe IV*.

Par ailleurs, l'application « mail » utilisait le serveur mail avec Postfix. Dans le Framework Applicatif, c'est le backend de Django qui était utilisé. Afin d'harmoniser l'utilisation du service de messagerie, l'envoi de mail avec Postfix a été implémenté dans les autres composants utilisant l'envoi de mail. Par exemple, l'envoi des rapports ou l'enregistrement d'un nouvel utilisateur.

Pour tirer avantage de cette fonctionnalité, nous avons intégré la configuration des serveurs de messagerie utilisée par Postfix, directement dans la compagnie pour laquelle les tests de pénétrations sont réalisés. Autrement dit, pour chaque compagnie nous pouvons utiliser plusieurs serveurs de messagerie associés à cette compagnie. Cela pourrait être utile lors de la mise en place d'une campagne de phishing[34].

4.4.7. Consolidation de l'application « credential_harvestring »

Le processus d'intégration de cette application est similaire à l'application « Mail » compte tenu des particularités d'implémentation.

4.4.8. Réutilisation de l'outil « Nmap » pour l'intégration du « Framework Réseau »

Le code source du projet « Framework Réseau » a été développé avec le framework Symfony. Ce point constitue une barrière importante à l'intégration des outils présents dans ce travail.

Premièrement, le framework Django et Symfony n'est pas écrit dans le même langage. Deuxièmement, l'algorithme mis en œuvre pour utiliser les outils de pénétrations n'a pas pu être réécrit.

[34] Le terme phishing désigne une technique qui consiste à envoyer des courriels à des utilisateurs en falsifiant l'expéditeur, afin de subtiliser des renseignements personnels ou privés. (Tipton & Krause, 2006)

Par manque de temps, ces considérations nous ont menés à intégrer uniquement l'outil Nmap. En effet, l'adaptation de la totalité des outils exigerait un temps d'analyse et d'implémentation trop important dans le cadre de ce travail.

4.4.9. Consolidation de la gestion des utilisateurs

Django Python intègre un système basique d'authentification. Mais il est possible de l'étendre avec des librairies telles que « django-registration-redux ». Cette libraire a été reprise et intégré à l'ensemble des applications dans le projet.

4.4.10. Consolidation de l'application rapport

Les anciens travaux de Bachelor utilisaient librairie « xhtml2pdf » pour générer les rapports. Une modification a été effectuée sur les modèles HTML pour générer des rapports.

4.4.11. Modification de la base de données

Les modifications n'ont pas été réalisées à la fin de notre procédure d'intégration. La base de données a évolué tout au long de cette procédure. Cette tâche a été nécessaire pour éviter les redondances, mais aussi pour ajouter de nouvelles fonctionnalités.

Les redondances des champs se trouvaient notamment dans les tables illustrées ci-dessous.

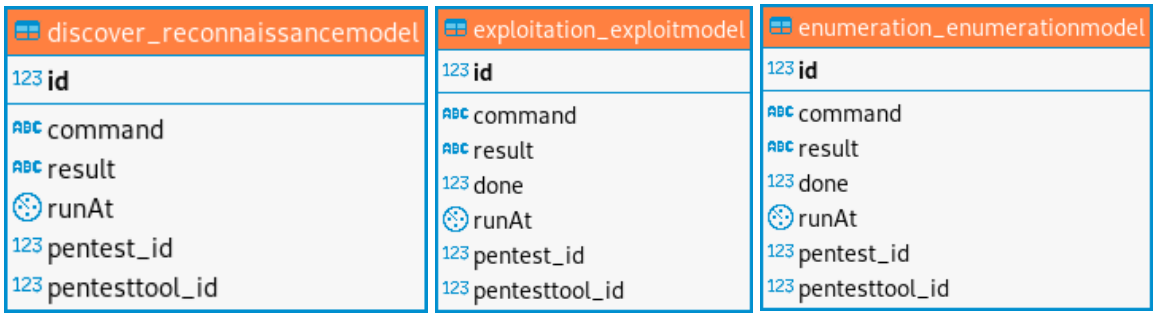


Figure 19 Extrait des tables de la base de données

4.4.12. Application des conventions PEP8

En général, des conventions et styles de codage permettent d'uniformiser les pratiques de développement. Ces conventions sont utiles pour maintenir le code dans un standard. Elles sont utilisées dans le but de rendre le code lisible et compréhensible par son auteur et les développeurs qui reprendront le code source d'un projet (Le Goff, 2019).

Dans ce travail, les éléments suivants ont été modifiés pour être conformes aux conventions PEP8 :

- Convention de nommage

- Déclaration des imports
- Suppression des imports non utilisés
- Indentation et espacements (4 espaces par section d'indentation)
- Longueur maximum des lignes (79 caractères)
- Harmonisation du style de développement
  (van Rossum & Warsaw, 2013)

Les conventions de nommage ont été appliquées aux variables, fonctions, méthodes, URL et nom des templates. Par exemple, les variables sont nommées à l'aide d'un soulignement. Vous trouverez quelques exemples à l'a*nnexe V*.

4.5. Synthèse de l'approche de consolidation

La procédure d'intégration a permis d'une part de consolider les fonctionnalités des anciens projets et d'autre part d'intégrer les outils de pentest applicatifs, social engineering et partiellement les outils de pentests réseaux. La base de données a été restructurée tout en supprimant les duplicatas. Le code source a été « nettoyé » et harmonisé selon les conventions PEP8.

Nous pouvons à peu de chose près affirmer que cette méthode d'intégration à fonctionné. Il faut tout de même dénoter que les premières tâches de consolidation, qui consistait à stabiliser les anciens projets a demandé beaucoup de temps. De plus, le concept de départ qui voulait des applications Django complètement indépendantes a été réalisé en partie, en raison des contraintes liées à la structure même des projets à intégrer. L'ambition de mettre à jour la version du framework Django et de Python n'a malheureusement pas abouti.

D'autre part, cette phase d'intégration à donné le jour à de nouvelles fonctionnalités et des améliorations.

5. Projet et applications Django

Le projet Django et les applications qui le composent utilisent plusieurs outils, services et librairies. Dans cette section, l'infrastructure et les technologies utilisées seront présentées, ainsi que les détails d'implémentation du framework global et unifié.

5.1. Infrastructure et librairies

Figure 20 Infrastructure

L'application web mise en place dans ce travail est dépendante de différents services et composants pour fonctionner. Le framework Django a été installé sur un système d'exploitation Kali Linux 2019.2 qui offre une suite d'outils de tests de pénétration. D'autres services tels que PostgreSQL, RabbitMQ, Apache, Celery et Postfix ont été ajoutés. Le développement de l'application web a été réalisé dans un environnement virtuel fourni par l'outil Virtualenv. Dans l'environnement virtuel, diverses librairies, telles que Bootstrap, JQuery et XHTML2PDF ont été installées à l'aide du gestionnaire de paquet PIP. Les autres dépendances installées dans l'environnement virtuel sont listées dans un fichier de dépendances (requirements.txt) dans le code source.

5.1.1. Kali Linux

Kali Linux est une distribution Linux basée sur Debian, qui est très utilisée par les professionnels de la sécurité. Ce système d'exploitation met à disposition plus de 600 outils utilisés pour des tests de pénétrations et d'autres tâches en sécurité. Cette distribution est maintenue et mise à jour par Offensive Security, une entreprise spécialisée dans la sécurité (Sri Manikanta, 2019).

5.1.2. PostgreSQL

PostgreSQL est une base de données relationnelle open sourde sous licence BSD. (The PostgreSQL Global Development Group, 2019)

5.1.3. RabbitMQ

RabbitMQ est un message broker open source qui permet de distribuer des messages. Il permet de gérer la file d'attente de messages entre des processus, des applications et des serveurs. Par exemple, un message peut inclure des informations concernant l'enchaînement d'un processus (Johansson, 2015). Dans ce projet, Celery utilise RabbitMQ pour gérer les tâches asynchrones à exécuter.

5.1.4. Celery

Celery est un logiciel open source qui permet de gérer les tâches asynchrones exécutées par un utilisateur. Autrement dit, Celery prend en charge de manière asynchrone les opérations exécutées du côté client (utilisateur). Ce qui permet à l'utilisateur de continuer à utiliser l'application web sans attendre la réponse de son action. (Ask Solem, 2019)

5.1.5. Apache

Apache est un serveur web open source maintenu par « The Apache Software Foundation ». Dans le contexte de ce projet, il est utilisé pour héberger les sites clonés par un des outils pentests pour faire du social-engineering.

5.1.6. Postfix

Postfix est un logiciel libre qui fournit un service de messagerie électronique. Dans ce projet il est utilisé pour envoyer les mails créés dans une campagne de phishing ou encore pour envoyer les rapports générés dans un pentest.

5.1.7. Virtualenv

Virtualenv permet d'isoler Python dans un environnement virtuel. Les motivations de son utilisation et de son fonctionnement ont déjà été résumées au chapitre 5. La documentation, que l'on trouve à l'adresse https://virtualenv.pypa.io/en/latest/, contient plus de détails concernant son utilisation.

5.1.8. JQuery

JQuery est une librairie JavaScript largement utilisée par les développeurs web. La philosophie de JQuery est d'écrire moins de code pour faire plus. Cette librairie permet de manipuler un document HTML, du code CSS (Cascading Style Sheets) et intègre l'architecture AJAX (Asynchronous JavaScript and XML).

5.1.9. Bootstrap

Bootstrap est l'un des frameworks CSS le plus utilisés pour développer des pages web et des applications mobiles « responsive[35] » (W3schools, 2019).

5.1.10. XHTML2PDF

Cette librairie permet de convertir du HTML en PDF. Dans ce projet, elle est utilisée lors de la génération des rapports.

[35] Un design « responsive » signifie que le contenu (HTML/CSS) d'un site ou d'une application mobile s'adapte à la taille de l'appareil utilisé.

5.2. Structure du dépôt des fichiers

Le répertoire « security-pentest » contient tout le dépôt des fichiers nécessaires au fonctionnement de l'application web.

```
[security-pentest]/
|---[installation-files]/
|---[static]/
|---[tools]/
|---[venv]/
```

Figure 21 Structure du dépôt

Le répertoire « installation-files » dans le dossier racine contient tous les scripts d'installation et les fichiers de configurations des services utilisés. Une explication plus détaillée de ces fichiers est fournie dans le « Guide d'installation » dans les documents en annexe.

```
[security-pentest]/
|---[installation-files]/
    |--- [apache]/
         |--- 000-default.conf
         |--- apache2.conf
    |--- [database]/
         |--- credential_harvester_settingsharvester_data.sql
         |--- mail_configsmtp_data.sql
         |--- pentest_pentesttool_data.sql
         |--- pentest_soustype_data.sql
    |--- createDb.sh
    |--- installDependencies.sh
    |--- main.cf
    |--- replaceConfigFiles.sh
    |--- sasl_passwd
    |--- startArachniServer.sh
    |--- startFrameworkLocal.sh
    |--- startServices.sh
```

Figure 22 Fichiers d'installations

Le répertoire « static » contient les librairies et autres fichiers statiques [36]utilisés dans Django.

[36] Les fichiers statiques sont « des fichiers supplémentaires tels que des images, du JavaScript ou du CSS »(Django Software Foundation, 2019b).

```
...
|---[static]/
    |--- [admin]/
    |--- [bootstrap]/
    |--- [ckeditor]/
    |--- [css]/
    |--- [debug_toolbar]/
    |--- [docs]/
    |--- [font-awesome]/
    |--- [fonts]/
    |--- [img]/
    |--- [js]/
    |--- [php]/
...
```

Figure 23 Fichiers statiques

Le répertoire « tools » contient l'outil de tests de pénétration « Arachni » qui n'est pas installée par défaut dans la distribution Kali Linux.

```
...
|---[tools]/
    |--- [arachni-1.5.1-0.5.12]/
...
```

Figure 24 Outils externes à Kali Linux

5.3. Code source et environnement virtuel

Dans ce travail, l'environnement virtuel et Django ont été installés dans le répertoire « venv ».

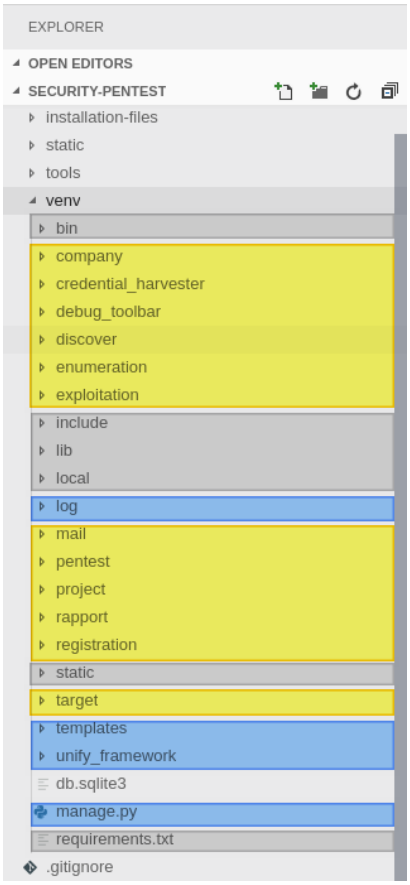


Figure 25 Répertoires de l'environnement virtuel et du code source

Dans l'illustration ci-dessus, le répertoire « venv » contient trois groupes de fichiers surlignés par une couleur. Les répertoires grisés (bin, include, lib, local, static) appartiennent à l'environnement virtuel. Les répertoires surlignés en bleu (log, templates, unify_framework, manage.py) sont les répertoires utilisés par le projet Django. Les répertoires surlignés en jaune (company, credential_harvester, debug_toolbar, etc.) sont les applications installées dans le projet.

5.4. Projet Django

Cette partie complète la mise en contexte du développement Django présentée au chapitre 4, afin de faire un lien au présent travail.

Lors de la création d'un nouveau projet Django avec le gestionnaire de paquets PIP, différents fichiers sont générés. Ici, le projet Django a été nommé « unify_framework ».

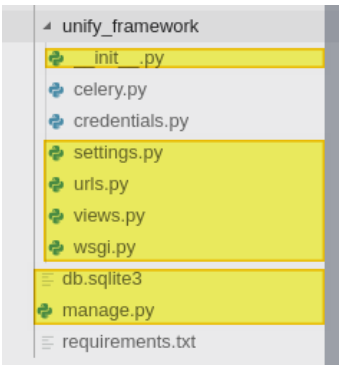


Figure 26 Structure du projet Django

Les fichiers surlignés en jaune sont générés par défaut lors de l'installation du framework Django. Les autres fichiers (celery.py, credentials.py, requirements.txt) ont été ajoutés ultérieurement.

__init__.py	Script python pour reconnaître le paquet dans le framework.
settings.py	Fichier de configuration du projet.
urls.py	Fichier de configuration utilisé pour la distribution des URL.
views.py	Vues (MVT) contenant la « logique » de l'application web.
wsgi.py	Permet de déployer l'application web sur un serveur web (NGINX, Apache, etc.)
manage.py	L'utilitaire manage.py permet de lancer des commandes fournies par Django. Par exemple, pour créer une nouvelle application ou lancer le serveur de développement.
celery.py	**Ce script permet de configurer Celery et de le combiner au service** RabbitMQ.
credentials.py	Ce script python stocke les constantes utilisées dans le fichier de configuration (settins.py) pour l'authentification à certains services utilisés. Il sera aussi utilisé lors de l'installation de ce projet.

Les constantes du fichier « credentials.py » sont récupérées dans le fichier « settings.py » (voir *annexe X*).

Le répertoire « log » contient le fichier « debug.log » où sont enregistrés les logs générés par le système de journalisation mis en place dans le projet.

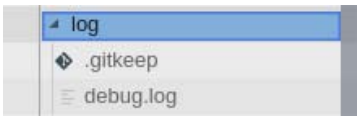


Figure 27 Répertoire log

Le répertoire « templates » contient tous les gabarits classifiés par application. Les gabarits sont des templates qui sont de simples fichiers textes qui peuvent contenir du HTML, du CSS, du JavaScript ou même du CSV. C'est un des trois blocs de l'architecture MVT.

Les templates peuvent hériter ou inclure d'autres templates. Plus d'informations au sujet de l'héritage des templates sont disponibles dans la documentation officielle à l'adresse https://docs.djangoproject.com/fr/2.2/ref/templates/language/.

Figure 28 Répertoire templates

5.5. Applications Django

La structure d'une application Django est expliquée au chapitre 4.

Les applications Django sont déclarées dans le fichier de configuration (settings.py), comme illustré ci-dessous à partir de la ligne 67 à 76.

```
54  # Application definition
55  INSTALLED_APPS = [
56      'django.contrib.sites',
57      # Import django-registration-redux library
58      'registration',
59      'django.contrib.admin',
60      'django.contrib.auth',
61      'django.contrib.contenttypes',
62      'django.contrib.sessions',
63      'django.contrib.messages',
64      'django.contrib.staticfiles',
65      'debug_toolbar',
66      # Import apps
67      'company',
68      'credential_harvester',
69      'discover',
70      'enumeration',
71      'exploitation',
72      'mail',
73      'pentest',
74      'project',
75      'rapport',
76      'target',
77      # Third part apps
78      'bootstrap',
79      'crispy_forms',
80      'fontawesome',
81      'ckeditor',
82      'django_celery_results',
83      'zapv2',
84      'jquery',
85      'jquery_ui',
86      'django_sb_admin',
87  ]
```

Figure 29 settings.py applications installées

5.5.1. Application « company »

L'application « company » est le point d'entrée pour commencer un « audit de sécurité [37]» pour une entreprise. Depuis une compagnie, il est possible de créer des projets, d'ajouter des cibles (Target) et de configurer le serveur SMTP de la compagnie. Si aucun serveur de messagerie n'est associé à l'entreprise, c'est le serveur défini par défaut qui sera utilisé.

Dans le menu de navigation, on peut accéder aux projets, aux pentests et aux cibles d'attaques (Targets) associés à l'entreprise (voir figure 32).

[37] « Un audit de sécurité est plus large qu'un test d'intrusion, lors d'un audit de sécurité, nous allons vérifier la sécurité organisationnelle, le PRA/PCA, DLP (Data Loss Prevention), la conformité par rapport aux exigences d'une norme (exemple : PCI DSS)… » (Ogma-sec, 2015)

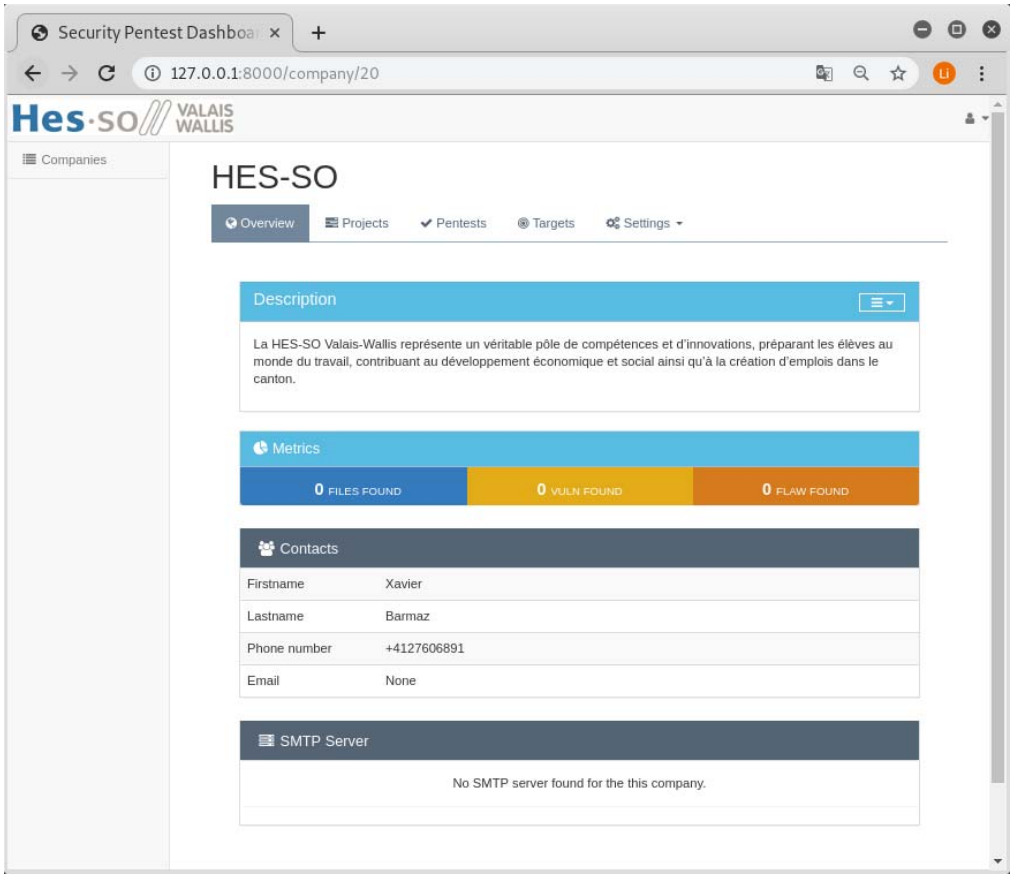


Figure 30 Overview company

5.5.2. Application « project »

Il existe trois types de projets : réseau, applicatif et social-engineering. Selon le type de projet sélectionné, il sera possible de créer plusieurs pentests qui utilisent des outils de pénétrations relatifs à la catégorie du projet (voir figure 34).

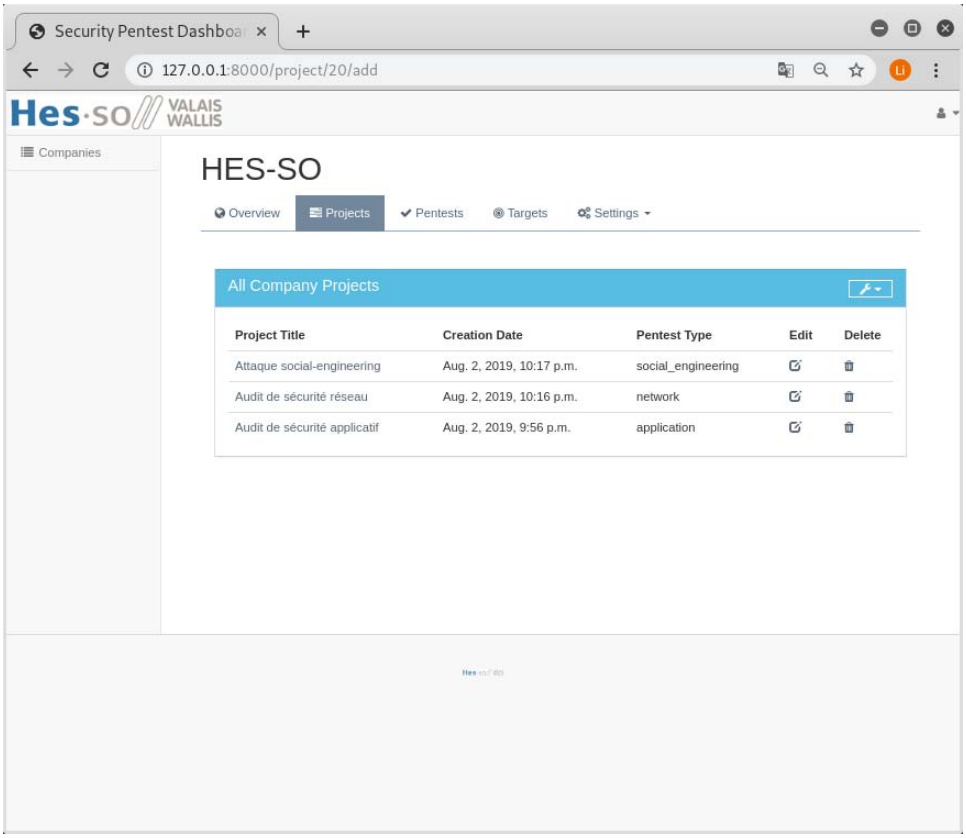


Figure 31 All company projects

5.5.3. Application « pentest »

Un pentest est associé à un projet. La création d'un pentest implique le choix d'une cible (Target) que l'on souhaite tester. La cible doit être définie au préalable pour l'associer à un pentest. Cette règle s'applique à tous les pentests à l'exception de ceux du type social-engineering.

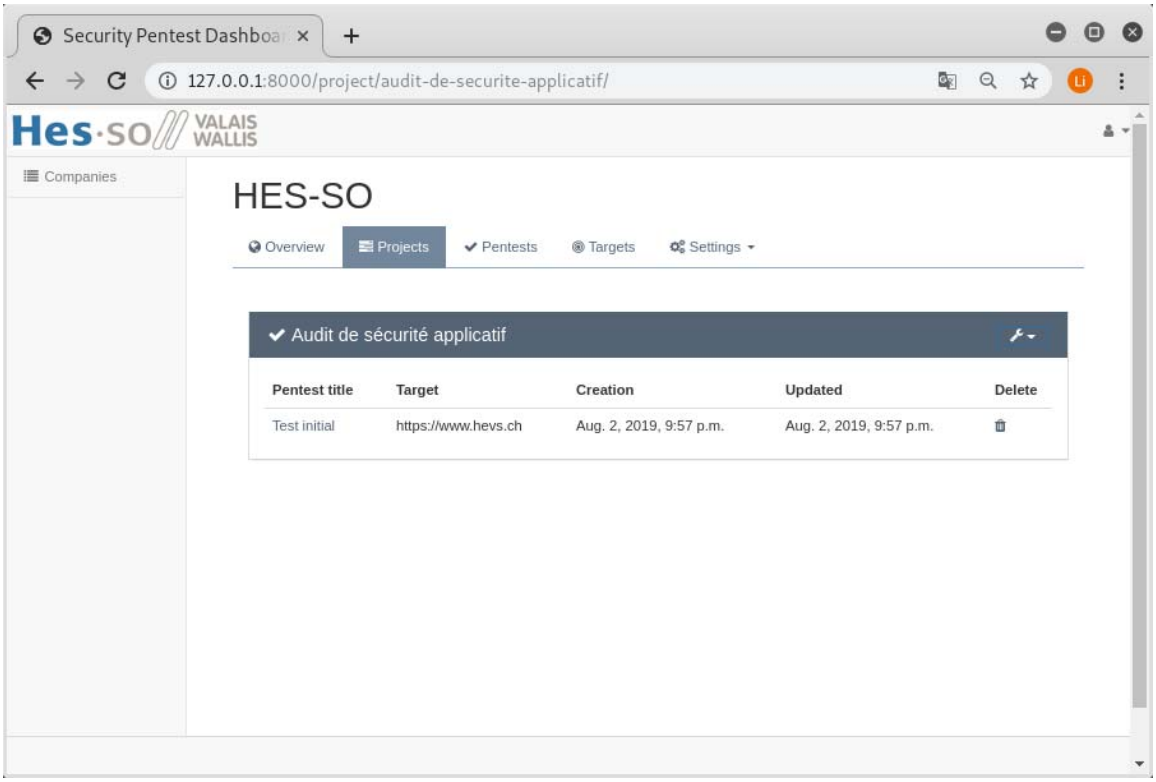


Figure 32 Pentest applicatif

Dans l'illustration suivante on aperçoit un extrait de la vue (views.py) de l'application pentest. On constate que selon le type de projet (surligné en jaune) un tableau de bord spécifique au type de pentest sera renvoyé. Il est de même pour les différentes phases que l'on retrouve dans le menu latéral gauche du tableau de bord.

```python
# Create your views here.
@login_required
def dashboard(request, id=None):
    pentest = get_object_or_404(Pentest, id=id)
    project = get_object_or_404(Project, id=pentest.project_id)
    note_form = PentestNoteForm(request.POST or None, instance=pentest)
    recom_form = PentestRecommendationsForm(request.POST or None,
                                            instance=pentest)
    report = Report.objects.filter(pentest=pentest)
    if project.pentestType == 'application':
        target = get_object_or_404(Target, pentest=pentest)
        app_reco_tools = PentestTool.objects.filter(
            tooltype="reco_app").count()
        app_reco_use = PentestPhases.objects.filter(pentest=pentest).exclude(
            result__isnull=True).exclude(result__exact='').count()
        app_enum_tools = PentestTool.objects.filter(
            tooltype="enum_app").count()
        app_enum_use = PentestPhases.objects.filter(pentest=pentest).exclude(
            result__isnull=True).exclude(result__exact='').count()
        if (app_reco_tools > 0):
            app_reco_perc = app_reco_use * 100 / app_reco_tools
        else:
            app_reco_perc = 0
        if (app_enum_tools > 0):
            app_enum_perc = app_enum_use * 100 / app_enum_tools
        else:
            app_enum_perc = 0

        report = Report.objects.filter(pentest=pentest)
        context = {
            "target": target,
            "pentest": pentest,
            "noteform": note_form,
            "recomform": recom_form,
            "nbrecontool": app_reco_perc,
            "nbenumtool": app_enum_perc,
            "report": report,
        }
        return render(request, "dashboard/dashboard_app.html", context)

    elif project.pentestType == 'social_engineering':
        soc_reco_tools = PentestTool.objects.filter(
            tooltype="reco_soc").count()
        soc_reco_use = PentestPhases.objects.filter(
```

Figure 33 Application pentest - views.py

Pour chaque phase d'un pentest (reconnaissance, énumération ou exploitation) les outils correspondants sont affichés. Le menu de navigation latéral permet de naviguer dans ces trois phases.

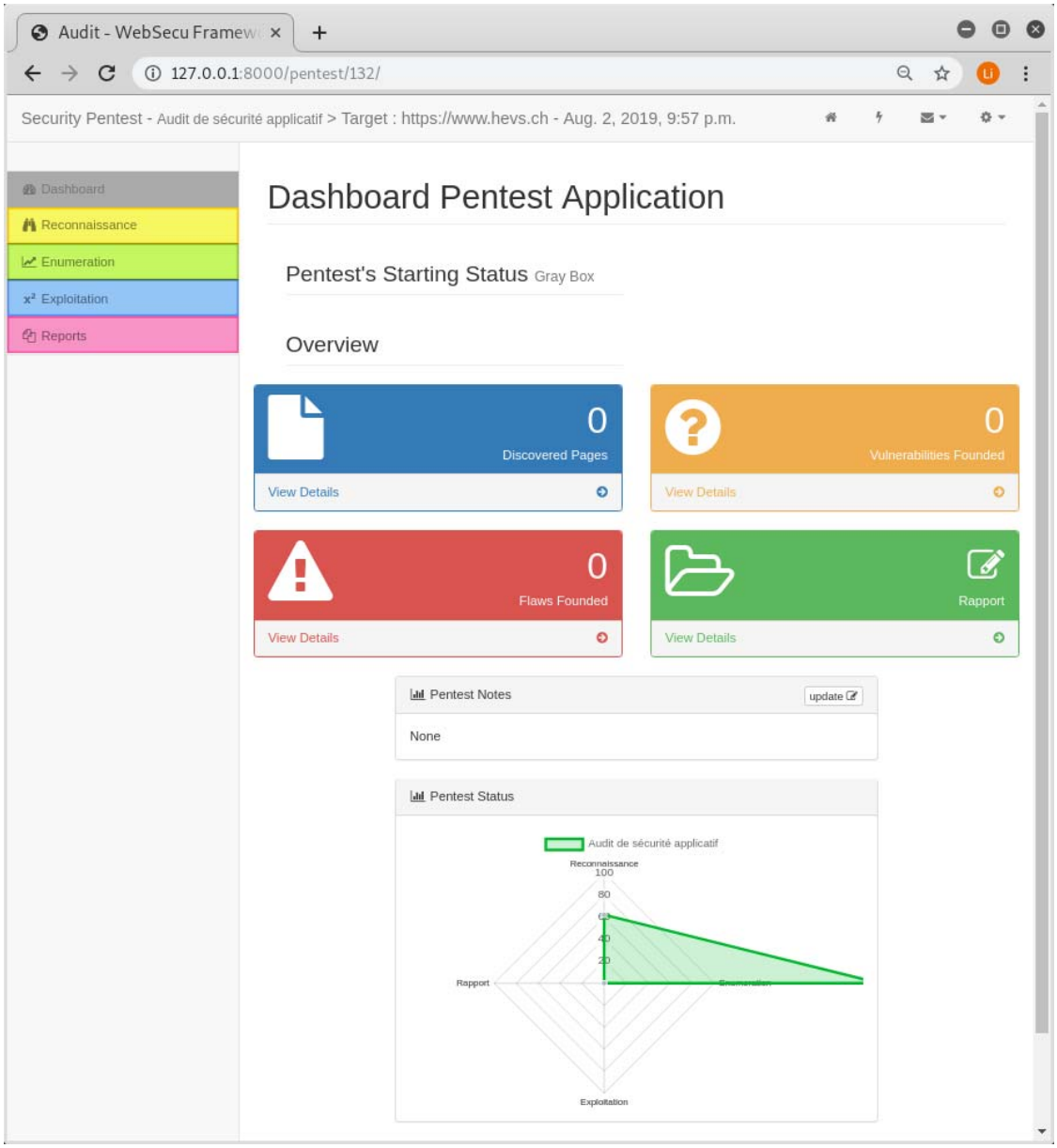


Figure 34 Application pentest - Tableau de bort

Les onglets du menu de navigation correspondent chacun à une application dans ce projet. Nous allons expliquer leur fonctionnement dans les prochains sous-chapitres.

5.5.4. Applications selon la phase du pentest

Les applications « discover », « ennumeration » et « exploitation » permettent d'exécuter des tâches en arrière-plan en utilisant Celery et RabbitMQ. Ces tâches (tasks.py) asynchrones sont exécutées dans des sous-processus qui permettent de lancer les outils de pénétration.

Dans l'image ci-dessous on peut voir que les outils *Whatweb*, *NMAP*, *WAFW00F* ont fini leur tâche. L'outil Nikto n'a pas encore été exécuté et *NMAP information Gathering* et *Dirbuster* sont en cours d'exécution.

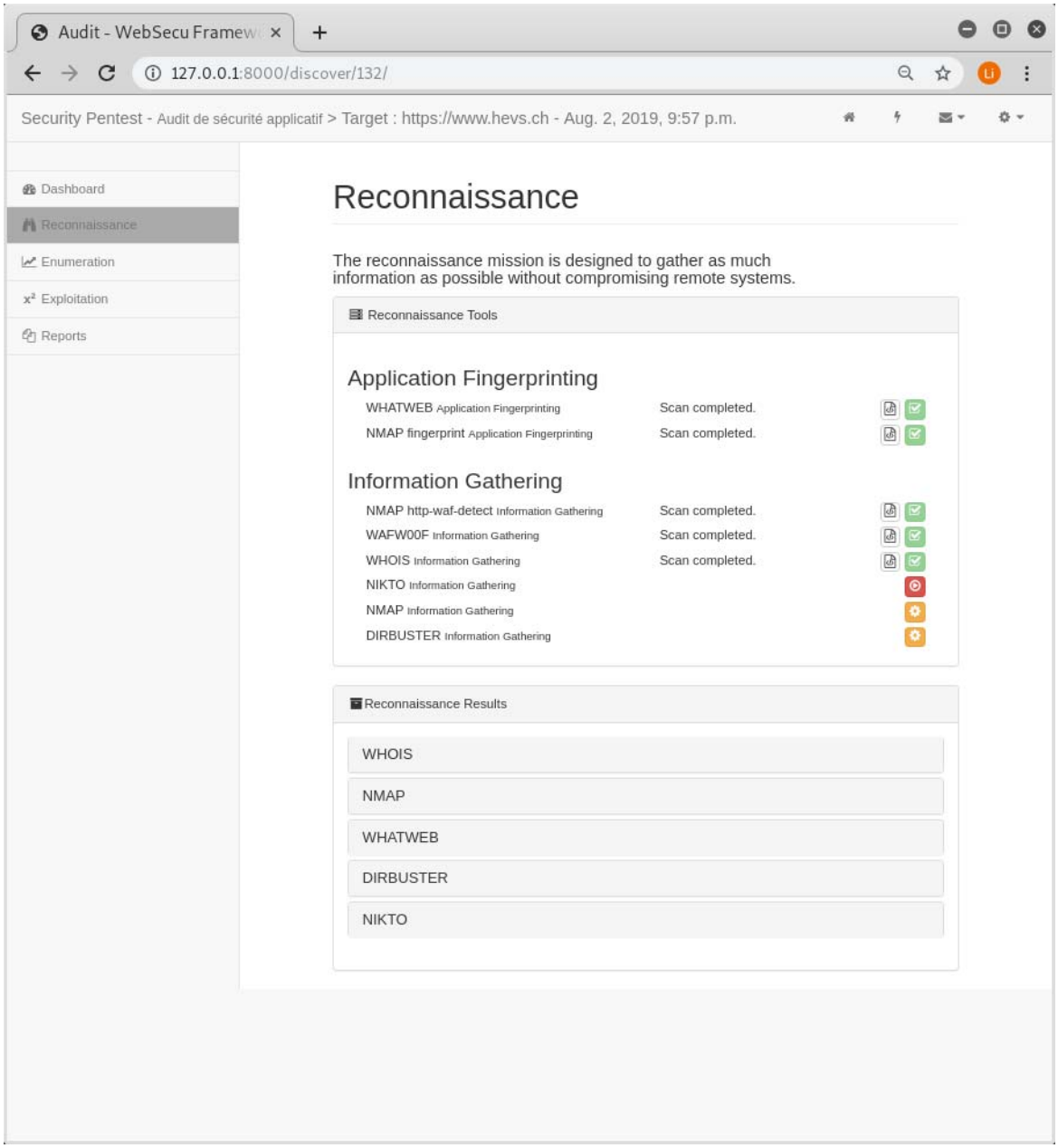


Figure 35 Application discover - tableau de bord

Les outils affichés sont récupérés depuis les tables « pentest_pentestool » et « pentest_soustype » dans la base de données.

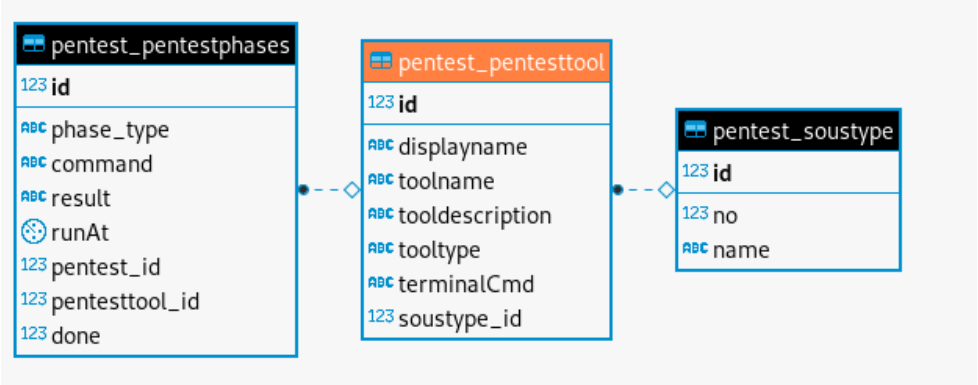


Figure 36 Tables pentest

Ces tables contiennent des outils de différents types, qui seront affichés dans le tableau de bord du même type. Autrement dit, les outils de type social engineering ne seront pas affichés dans le tableau de bord des pentests applicatifs. Ci-dessous on voit un extrait de la table où de nouveaux outils peuvent être enregistrés.

pentest_pentesttool | Enter a SQL expression to filter results (use Ctrl+Space)

	id	displayname	toolname	tooldescription	tooltype	terminalCmd	soustype_id
1	1	NIKTO	nikto	Nikto is an Open Source (GPL) web server scanner which p	reco_app	nikto -h "exemple.com"	2
2	2	WHATWEB	whatweb	WhatWeb recognises web technologies including content	reco_app	whatweb -v "exemple.cc	1
3	3	OWASP ZAP	zap	The OWASP Zed Attack Proxy (ZAP) is one of the world's i	enum_app	zap.urlopen(target)	3
4	4	ARACHNI	arachni	Arachni is a feature-full, modular, high-performance Ruby	enum_app	arachni.ArachniClient()	3
5	5	SQLMAP	sqlmap	Sqlmap is an open source penetration testing tool that aut	expl_app	sqlmap -u "http://exemp	4
6	6	XSSER	XSSer	Cross Site "Scripter" (aka XSSer) is an automatic -framew	expl_app	xsser -u '"http://exemple	6
7	7	HARVESTING	social kit tool	Harvesting is a technique, as its name suggests, to harvest	expl_soc	1;2;3;2;%IP;%website	14
8	8	MAIL	mail	We these tool you can prepare social engineering attacks I	expl_soc		15
9	9	DIRBUSTER	dirbuster	DirBuster is a multi threaded java application designed to	reco_app	dirb "exemple.com"	2
10	10	NMAP fingerprint	NMAP fingerprin	The fingerprints of known operating systems that Nmap re	reco_app	[NULL]	1
11	11	NMAP http-waf-det	NMAP http-waf-	WAF (Web application firewalls) plays an important role in	reco_app	[NULL]	2
12	12	WAFW00F	WAFW00F	WAFW00F is a Python tool to help you fingerprint and iden	reco_app	[NULL]	2
13	13	WHOIS	WHOIS	WHOIS (pronounced as the phrase "who is") is a query and	reco_app	[NULL]	2
14	14	NMAP Network	NMAP Network	NMAP can find host in a network	reco_net	[NULL]	16
15	15	NMAP	NMAP	Nmap, short for Network Mapper, is a free, open-source t	reco_app	[NULL]	2

Figure 37 Table pentest_pentesttool

Les outils sont aussi filtrés selon la phase du pentest correspondante. Par exemple, dans la vue (views.py) de l'application « discover » uniquement les outils de cette phase sont envoyés au template. Voir l'illustration du code à l'*annexe VI*.

Pour lancer les outils, un script JQuery récupère l'identifiant de l'outil dans une balise HTML et fait une requête à une méthode dans la vue correspondante. A l'aide de l'identifiant, il sera possible d'exécuter une tâche Celery. Cette tâche va lancer un sous-processus dans Kali Linux qui exécute l'outil en arrière-plan. Chaque exécution est enregistrée dans la table « PentestPhases » (voir le

diagramme de classe au chapitre 6.10). Cela permet d'historiser l'utilisation des outils et de récupérer le résultat généré. Le code à l'*annexe VII* et l'*annexe VIII* illustre ce fonctionnement.

Le résultat est ensuite récupéré à l'aide d'un script JQuery. Le script appelle la méthode « get_recon_response() » dans la vue pour renvoyer les résultats des outils au template. Le code source de cette méthode est illustrée à l'*annexe IX.*

Le résultat est finalement renvoyé au template et affiché à l'utilisateur, comme illustré ci-dessous. Ceci est valable notamment pour les applications « discover » et « enumeration ».

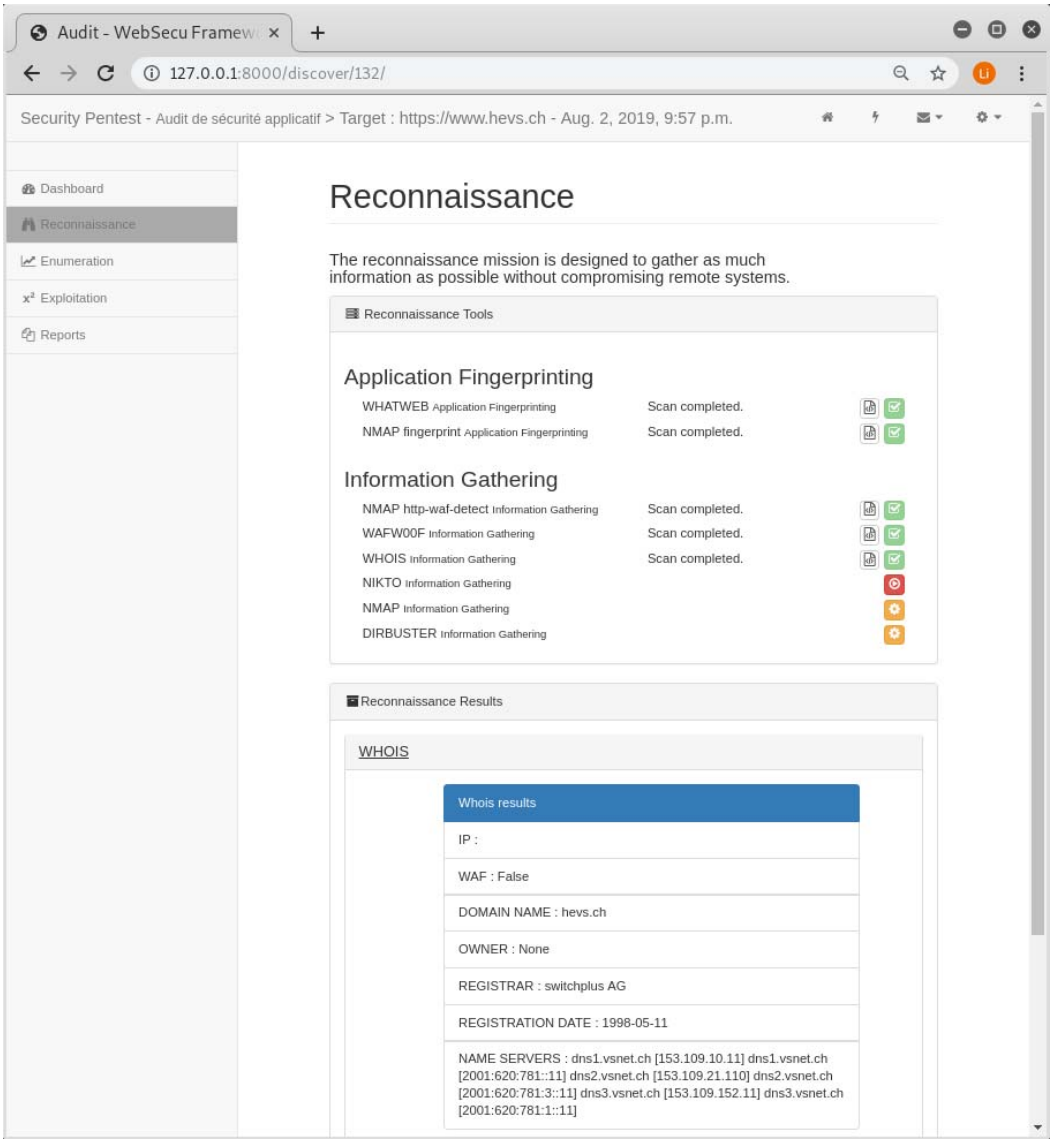


Figure 38 Application discover - template discover_app.html

En ce qui concerne, les tests de pénétration social engineering, il y avait uniquement deux outils à intégrer. L'outil « mail » et « credential_harvesting » qui sont des applications Django.

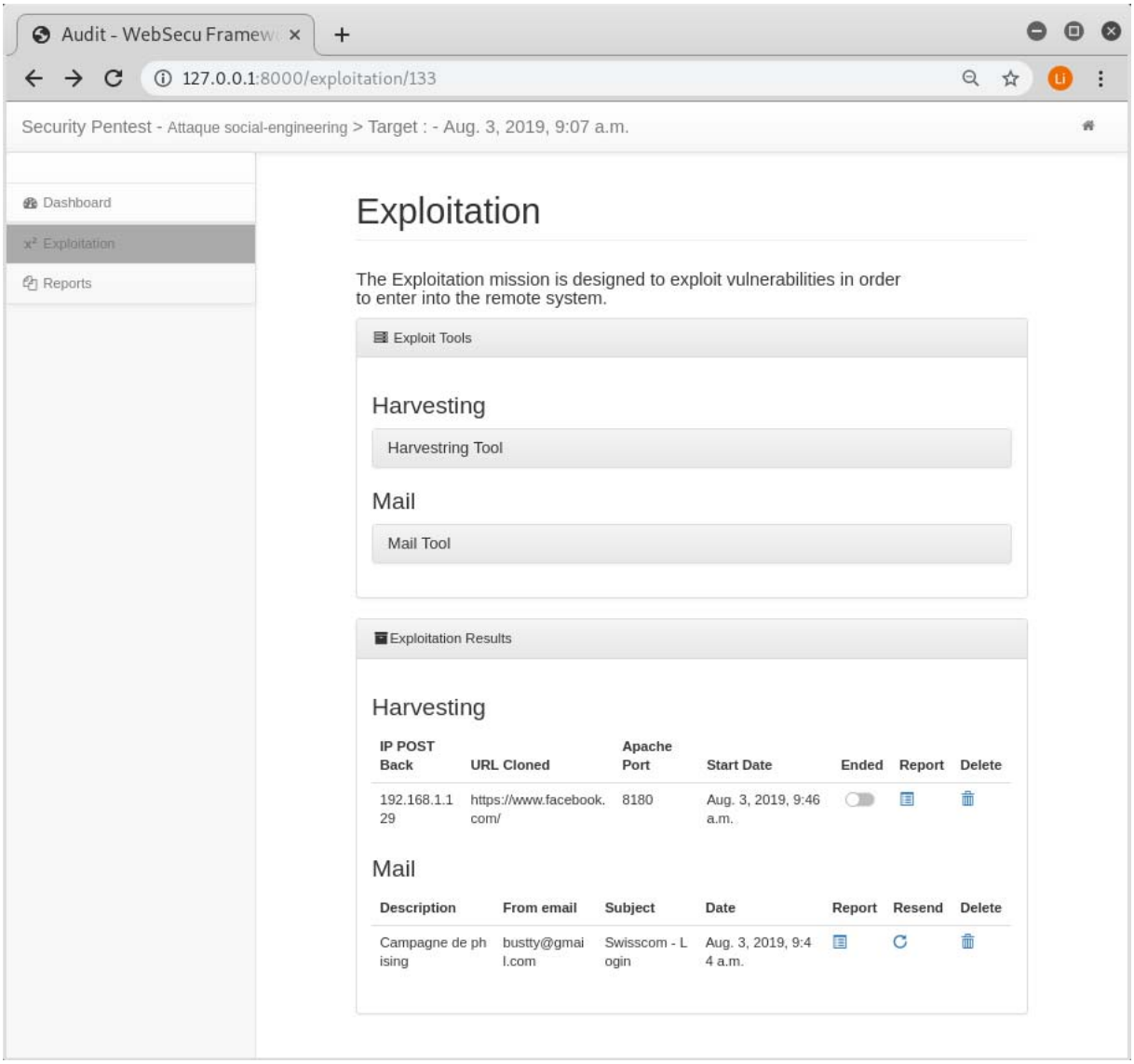


Figure 39 Tableau de bord social-engineering

Finalement, tous les résultats pour chaque catégorie de pentests sont synthétisés dans un rapport à l'aide de l'application « rapport ». On accède à cette fonctionnalité depuis le menu de navigation latéral.

5.5.5. Application « mail »

L'application mail contient une fonctionnalité pour créer des campagnes de phishing dans des projets de type social engineering. Cet outil permet d'envoyer des mails infectés, de tracer l'ouverture d'un mail et d'insérer une URL contenant un lien caché. Par exemple, pour récolter des identifiants d'un utilisateur en usurpant la page d'authentification de la plateforme sur laquelle il posséderait un compte.

5.5.6. Application « credential_harvester »

Cette application est dédiée aux attaques appelées « Credential Harvester », qui consiste à voler les identifiants d'un utilisateur.

5.5.7. Application « target »

Cette application permet de gérer les cibles d'attaques d'une entreprise qui seront utilisées dans les pentests de type réseau et applicatifs.

L'ajout de nouvelles « targets » se fait via un formulaire qui va récupérer le protocole, le « fully qualified domain name » (FQDN[38]), l'hôte, le port et le chemin de l'adresse IP ou l'URL.

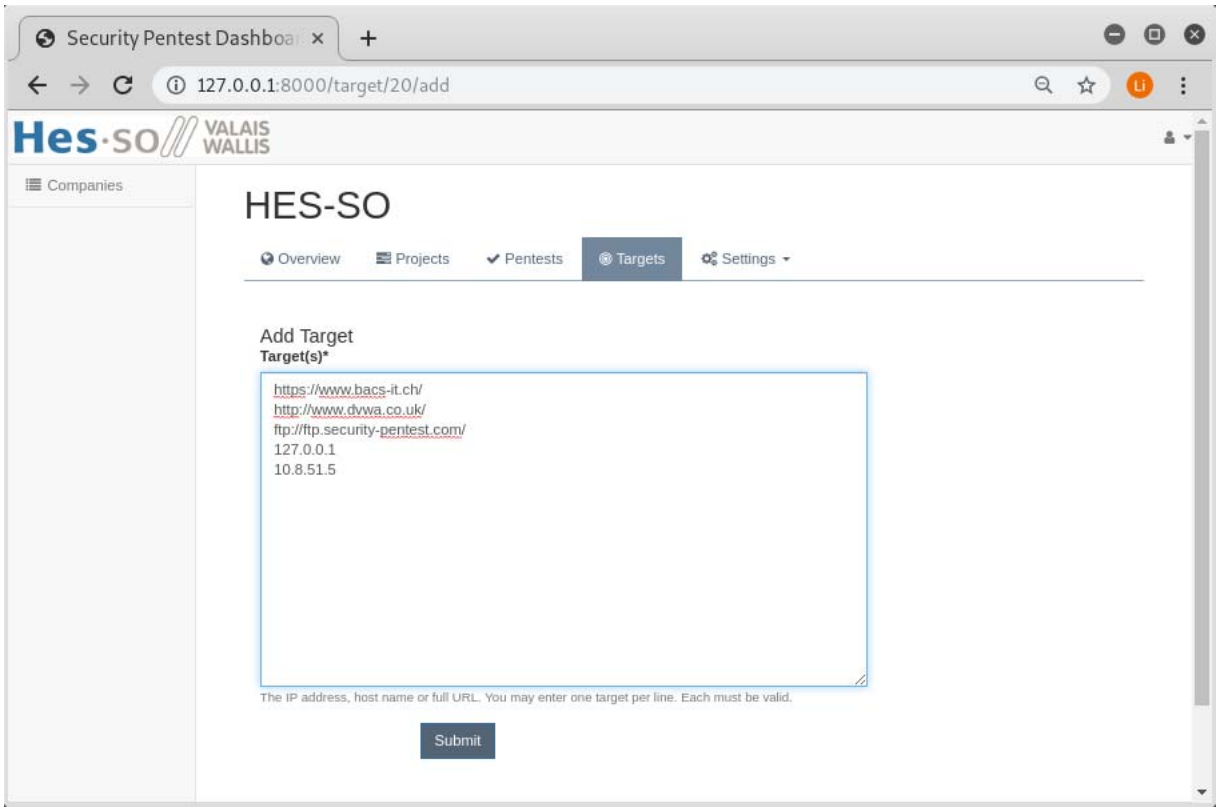


Figure 40 Company targets

[38] https://en.wikipedia.org/wiki/Fully_qualified_domain_name

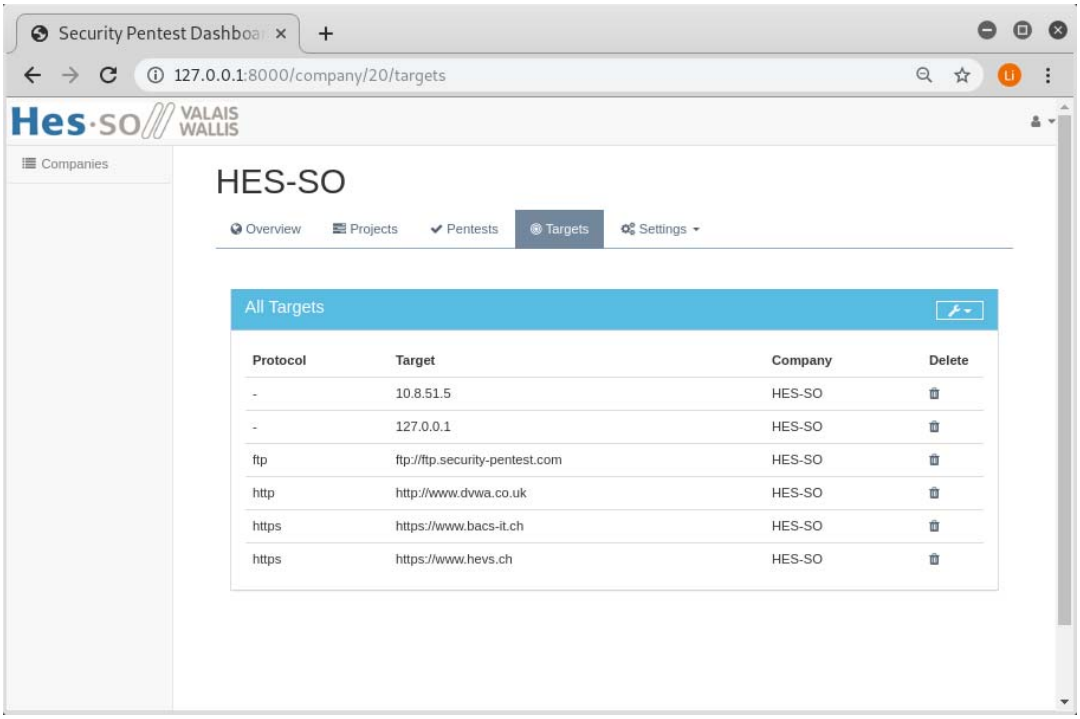


Figure 41 Liste des Targets d'une compagnie

Ces « targets » sont ensuite utilisés, lors de la création d'un pentest.

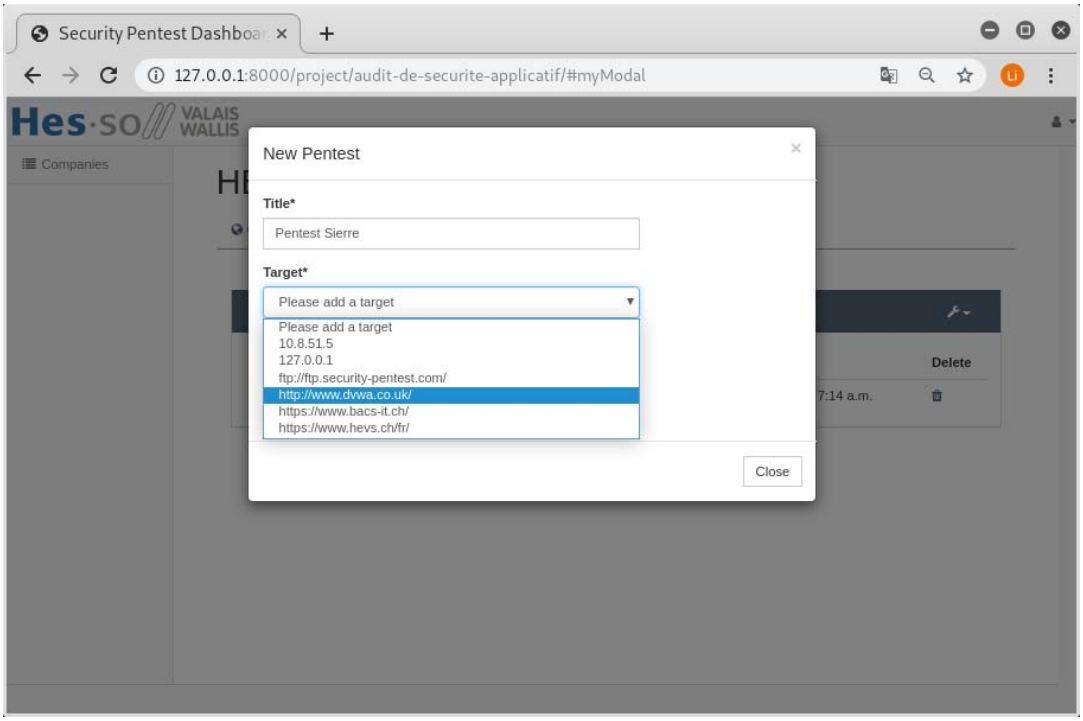


Figure 42 Création d'un pentest

5.5.8. Application « rapport »

L'application rapport récupère tous les résultats des outils réalisés dans un pentest et génère un fichier PDF. Le rapport peut ensuite être envoyé par mail.

5.5.9. Application registration

Le système d'authentification est géré par l'application « registration » qui est une extension du système de gestion des utilisateurs mis en place par Django. Cette application a été installée avec le gestionnaire de paquet PIP. Plus d'informations sont disponibles à l'adresse https://django-registration-redux.readthedocs.io/en/latest/.

5.6. Fonctionnement de Django

Pour comprendre le processus de traitement des requêtes par Django, nous allons dans ce chapitre expliquer sommairement l'interaction d'un utilisateur avec l'application web à l'aide d'un scénario.

5.6.1. Scénario

Un utilisateur est connecté à l'application web et veut accéder au formulaire pour créer une nouvelle compagnie.

5.6.2. Processus

L'utilisateur est sur la page d'accueil et s'apprête à créer une nouvelle compagnie.

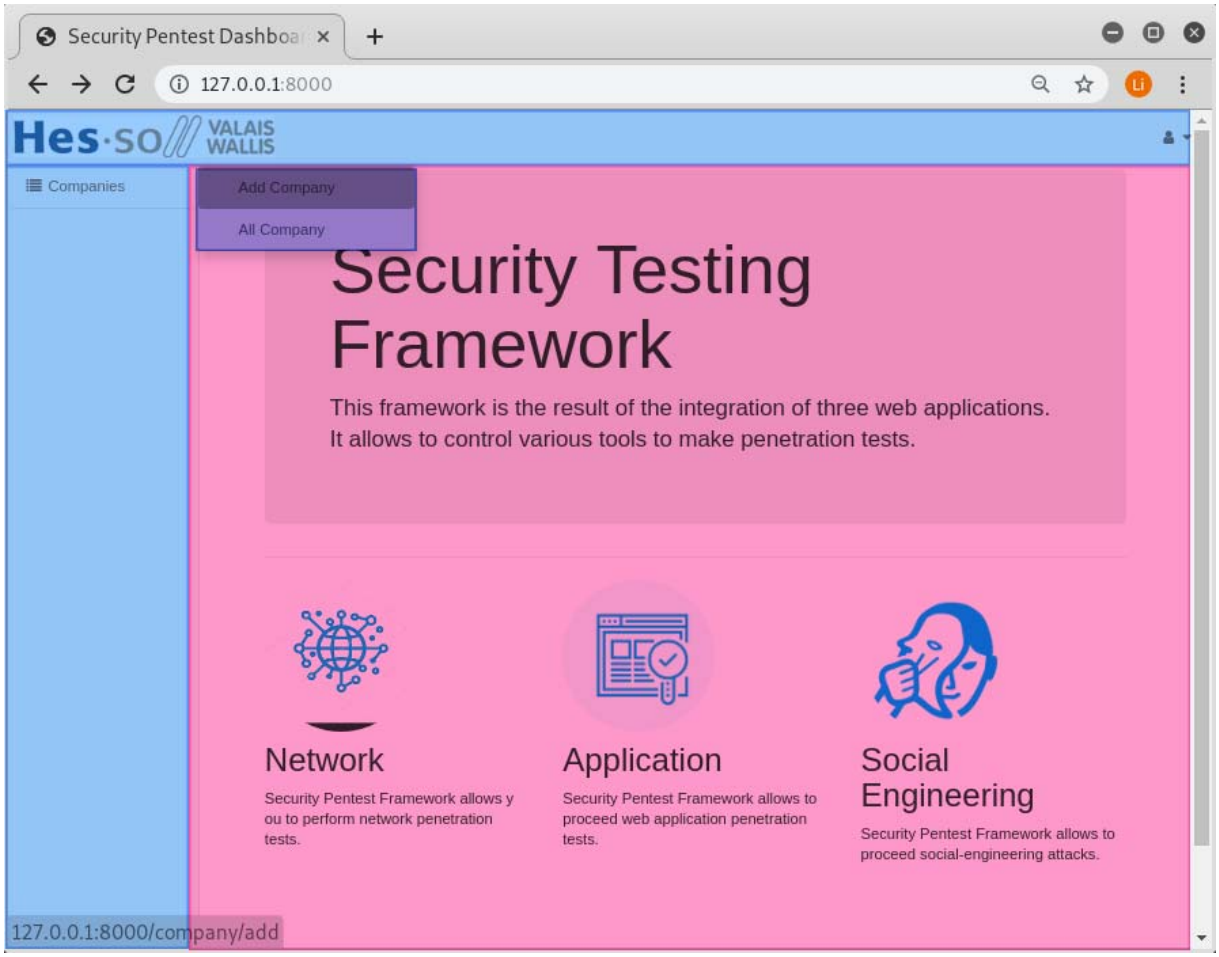


Figure 43 Fonctionnement - Ajout d'une nouvelle compagnie

Ce que l'utilisateur voit dans son navigateur est le résultat de l'affichage du gabarit « home.html » surligné en rose dans l'image ci-dessus. Ce template se trouve dans le répertoire */security-pentest/venv/templates/common/home.html.* Ce template hérite du gabarit « base.html » surligné en bleu, qui se trouve dans le répertoire */security-pentest/venv/templates/base.html*.

La figure ci-dessous illustre un extrait du gabarit « home.html » qui étend le gabarit « base.html ».

```
{% extends "base.html" %}
{% load staticfiles %}

{% block content %}
<div class="jumbotron">
   <h1 class="cover-heading">Security Testing Framework</h1>
   <p class="lead">
   This framework is the result of the integration of three web applicat
   </p>
</div>

```

Figure 44 Template : home.html

Lorsque l'utilisateur clique sur « Add Company » dans le menu latéral gauche, une requête HTTP est envoyée à Django pour accéder à la page demandée. Django va charger le module déclaré dans la constante ROOT_URLCONF du fichier de configuration. Dans ce cas, le module chargé est « urls.py » qui se trouve dans le répertoire « unify_framework » à la base de notre projet. L'URL recherchée est /company/add.

```
ROOT_URLCONF = 'unify_framework.urls'
```

Figure 45 settings.py - ROOT_URLCONF

La variable « urlpatterns » dans le fichier « url.py » est parcourue jusqu'à ce que Django trouve un motif d'URL correspondant. Dans ce cas, c'est l'URL surlignée en jaune ci-dessous qui correspond à la requête.

```
urlpatterns = [
    url(r'^$', views.home, name='home'),
    url(r'^admin/', admin.site.urls),
    url(r'^aboutus/', views.aboutus, name='aboutus'),
    url(r'^accounts/', include('registration.backends.default.urls')),
    url(r'^ckeditor/', include('ckeditor_uploader.urls')),
    url(r'^company/', include(company_urls)),
    url(r'^credential-harvester/', include(harvester_urls)),
    url(r'^discover/', include(reconnaissance_urls)),
    url(r'^enumeration/', include(enumeration_urls)),
    url(r'^exploitation/', include(exploitation_urls)),
    url(r'^mail/', include(mail_urls)),
    url(r'^project/', include(project_urls)),
    url(r'^pentest/', include(pentest_urls)),
    url(r'^report/', include(rapport_urls)),
    url(r'^target/', include(target_urls)),
]
```

Figure 46 unify_framework - urls.py

Lorsque l'URL est trouvée, une fonction de la vue est appelée. Dans ce cas, c'est la fonction « views.new_company » qui est appelée dans la vue de l'application « company ».

```
4  urlpatterns = [
5      url(r'^(?P<id>\d+)$', views.view_company, name='view_company'),
6      url(r'^companies$', views.get_all_companies, name='companies'),
7      url(r'^add', views.new_company, name='new_company'),
8      url(r'^(?P<id>\d+)/delete$', views.delete_company, name='delete_company'),
9      url(r'^(?P<id>\d+)/projects', views.view_company_projects,
10         name='view_company_projects'),
11     url(r'^(?P<id>\d+)/targets', views.view_company_targets,
12         name='view_company_targets'),
13     url(r'^(?P<id>\d+)/smtpconfig', views.view_smtp_configs,
14         name='view_smtp_configs'),
15     url(r'^(?P<id>\d+)/update', views.update_company, name='update_company'),
16 ]
```

Figure 47 company - urls.py

La fonction « new_company » reçoit la requête et renvoie le template (surligné ci-dessous en vert) avec son contexte (surligné en violet) qui contient le formulaire.

```
104 @login_required
105 def new_company(request):
106     form = CompanyForm(request.POST)
107
108     # Catch POST request from form
109     if request.method == 'POST':
110         form = CompanyForm(request.POST)
111
112         if form.is_valid():
113             form.save()
114             return redirect('companies')
115
116     return render(request, 'company/new_company.html', {'form': form})
```

Figure 48 company - views.py

Finalement le formulaire pour la création d'une nouvelle compagnie est affiché.

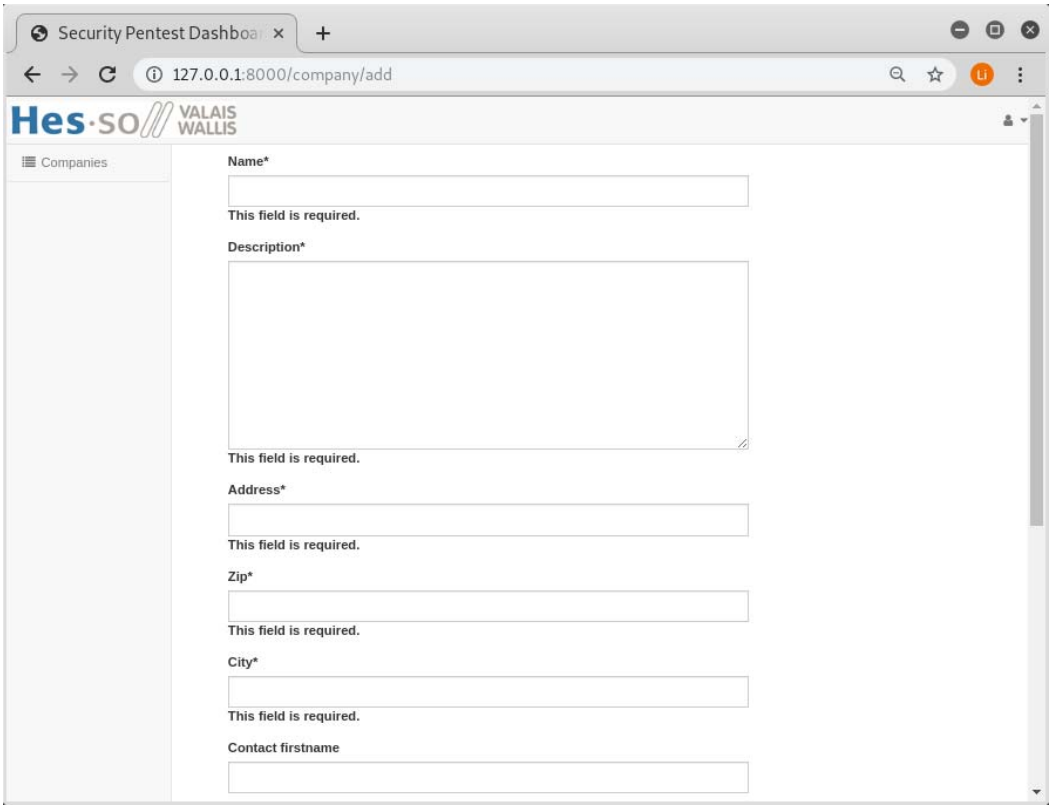


Figure 49 Template - new_company.html

L'algorithme du processus de traitement est le même pour toutes les requêtes[39].

5.7. Système de journalisation

Django intègre nativement un système de journalisation[40], qu'il faut configurer dans le projet. En développement, ce système est très utile pour déboguer les erreurs, mais en production il est indispensable. Par exemple, pour envoyer des informations par mail à l'administrateur du site à des fins de maintenance (Baumgartner, s. d.).

[39] Pour plus d'informations, visitez https://docs.djangoproject.com/fr/2.2/topics/http/urls/.

[40] En anglais « logging »

Le logging est configuré dans le fichier settings.py.

```
156  # Logger
157  logging.config.dictConfig({
158      'version': 1,
159      'disable_existing_loggers': False,
160      'formatters': {
161          'console': {
162              'format': '%(asctime)s %(name)-12s %(levelname)-8s %(message)s'
163          },
164          'file': {
165              'format': '%(asctime)s %(name)-12s %(levelname)-8s %(message)s'
166          }
167      },
168      'handlers': {
169          'console': {
170              'class': 'logging.StreamHandler',
171              'formatter': 'console'
172          },
173          'file': {
174              'level': 'DEBUG',
175              'class': 'logging.FileHandler',
176              'formatter': 'file',
177              'filename': '/var/www/security-pentest/venv/log/debug.log'
178          },
179
180      },
181      'loggers': {
182          '': {
183              'handlers': ['console', 'file'],
184              'level': 'DEBUG',
185              'propagate': True,
186          },
187      },
188  })
189
```

Figure 50 Configuration de la journalisation - settings.py

Le logger peut ensuite être appelé dans les composants souhaités[41].

```
1   import json
2   import logging
3
4   from django.shortcuts import render, get_object_or_404, redirect
5   from pentest.models import (PentestPhases, Pentest,
6                               PentestTool, SousType)
7   from models import (Whatweb, Technology,
8                       Reconurl, Nikto, Host,
9                       Port, Server)
10  from project.models import Project
11  from target.models import Target
12  from django.core import serializers
13  from django.http import HttpResponse
14  from tasks import (nmap_reco_server_scan_task, nmap_reco_waf_identif_task,
15                     nmap_reco_http_waf_detect_task, wafwoof_detect_task,
16                     whois_reco_task, whatweb_reco_task, dirb_reco_task,
17                     nikto_reco_task)
18  from django.contrib.auth.decorators import login_required
19
20  logger = logging.getLogger(__name__)
21
```

Figure 51 Importer le système de journalisation

[41] Plus d'informations sont disponibles dans la documentation officielle à l'adresse https://docs.djangoproject.com/fr/2.2/topics/logging/

5.8. Debug Toolbar

La barre d'outils de débogage est une librairie considérablement utile lorsqu'on développe un projet Django. Avec cet outil il est possible voir ce qui est utilisé par Django en arrière-plan. Par exemple, le template affiché, les requêtes SQL exécutées, etc.

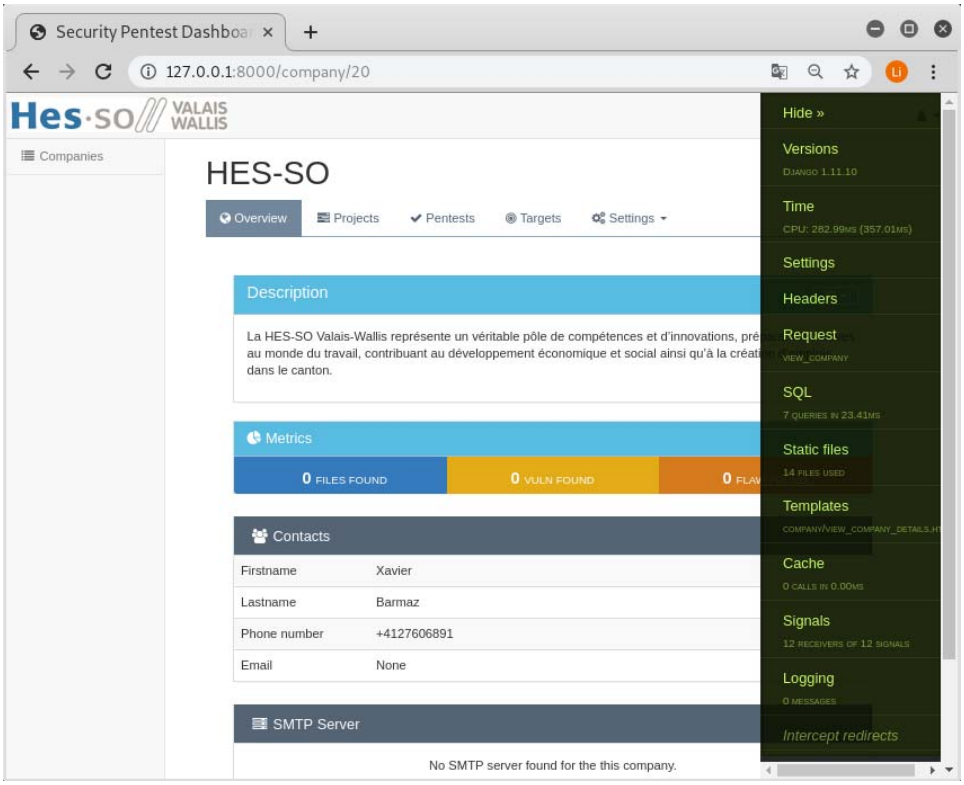


Figure 52 Barre d'outils de débogage

Pour utiliser cet outil, il faut configurer Django en mode débogage et activer le callback[42] pour afficher cette barre latérale.[43]

```
49  # SECURITY WARNING: don't run with debug turned on in production!
50  DEBUG = True
51
52  # ACTIVATE DEBUG TOOLBAR
53  DEBUG_TOOLBAR_CONFIG = {
54      'SHOW_TOOLBAR_CALLBACK': lambda r: True,  # disables it = FALSE
55  }
56
```

Figure 53 Activation de l'outil de débogage

[42] Un callback est une fonction accessible par une autre fonction qui est appelée lorsque la première est finie (8bitjunkie, 2018)

[43] La documentation se trouve à l'adresse https://django-debug-toolbar.readthedocs.io/en/latest/.

5.9. Interface d'administration

Le framework Django intègre une interface d'administration qui permet d'interagir avec les modèles des différentes applications. La classe du modèle doit être enregistrée dans le fichier « admin.py » (Django Software Foundation, 2019). Les droits administrateur (superuser) sont nécessaires pour accéder à cette interface[44].

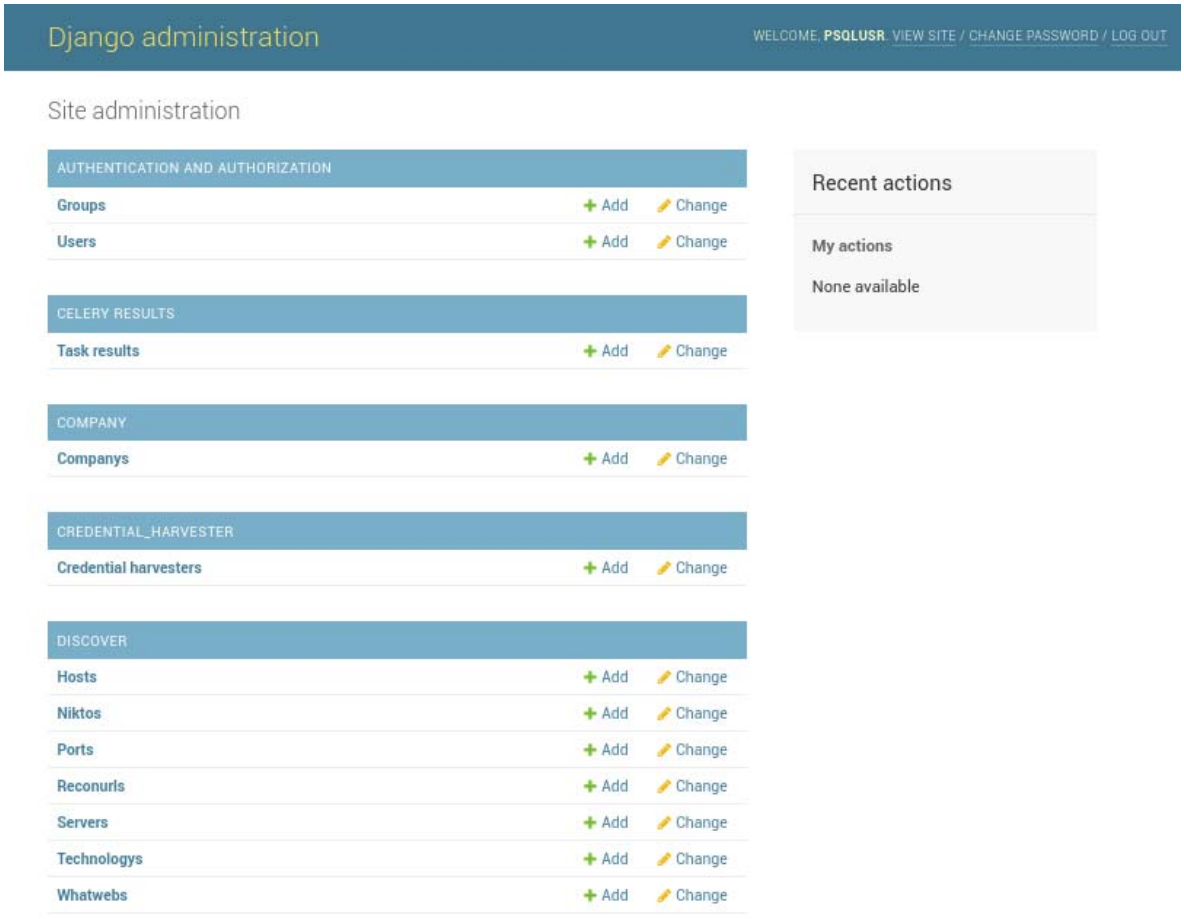
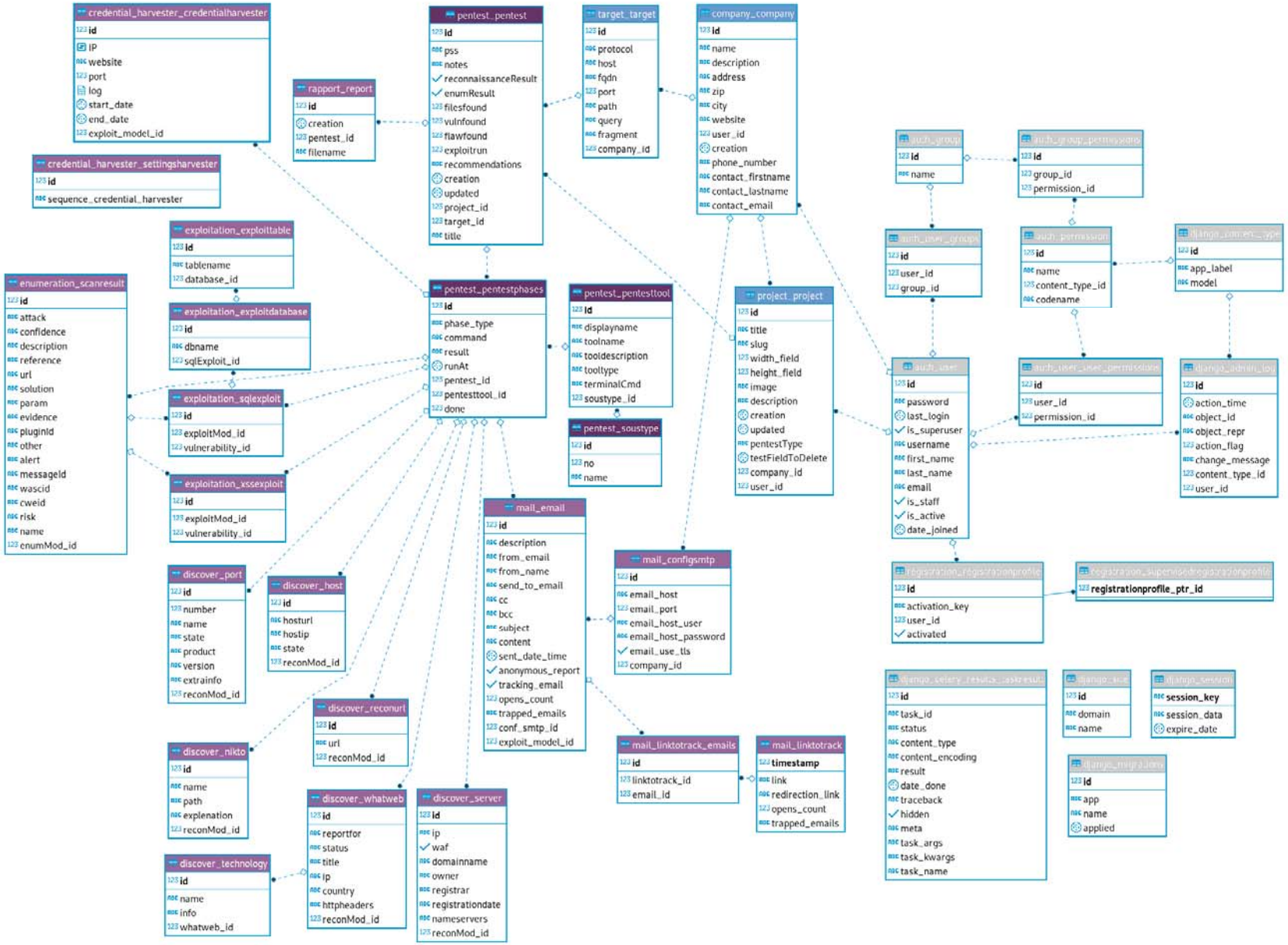


Figure 54 Site d'administration

[44] Des informations complémentaires concernant le site d'administration sont accessibles à l'adresse de la documentation de Django https://docs.djangoproject.com/fr/2.2/ref/contrib/admin/

5.10. Diagramme de classe



5.11. Améliorations

La « mise à niveau » des anciens projets dans la phase d'intégration était une tâche importante et n'a pas permis d'implémenter les améliorations énumérées ci-dessous. En outre, certaines de ses améliorations ont été commencées, mais n'ont pas pu être abouties.

- Mise à jour de la version de Django et de Python : cette tâche n'a malheureusement pas été priorisée dans les étapes d'intégration.
- Déployer l'application Django sur un serveur web de production : cette tâche n'a pas abouti lors du déploiement.
- Repenser l'architecture du projet pour rendre l'utilisation des outils plus flexible. Par exemple, une application Django par outil, afin de l'utiliser comme une API.
- Intégration des outils du Framework Réseau : cette tâche n'est pas uniquement une traduction du PHP en Python, mais un travail d'analyse.
- Gestion des autorisations d'accès par compagnie et projets : par exemple, on pourrait imaginer qu'un client puisse récupérer tous les rapports générés dans un projet, etc.

6. Visuel de l'application web

6.1. Login

L'accès à l'application web force l'utilisateur à s'authentifier pour accéder aux autres pages.

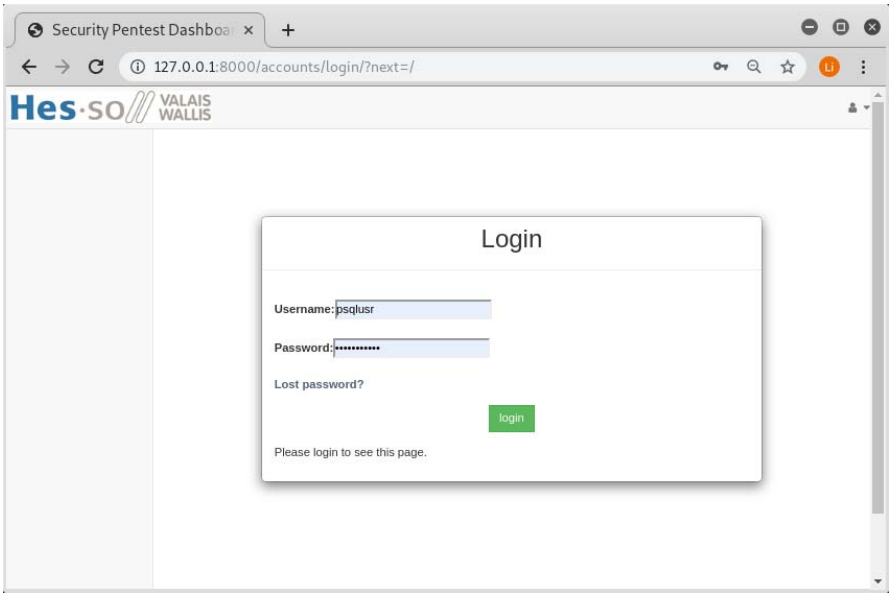


Figure 55 Page de login

6.2. Page d'accueil

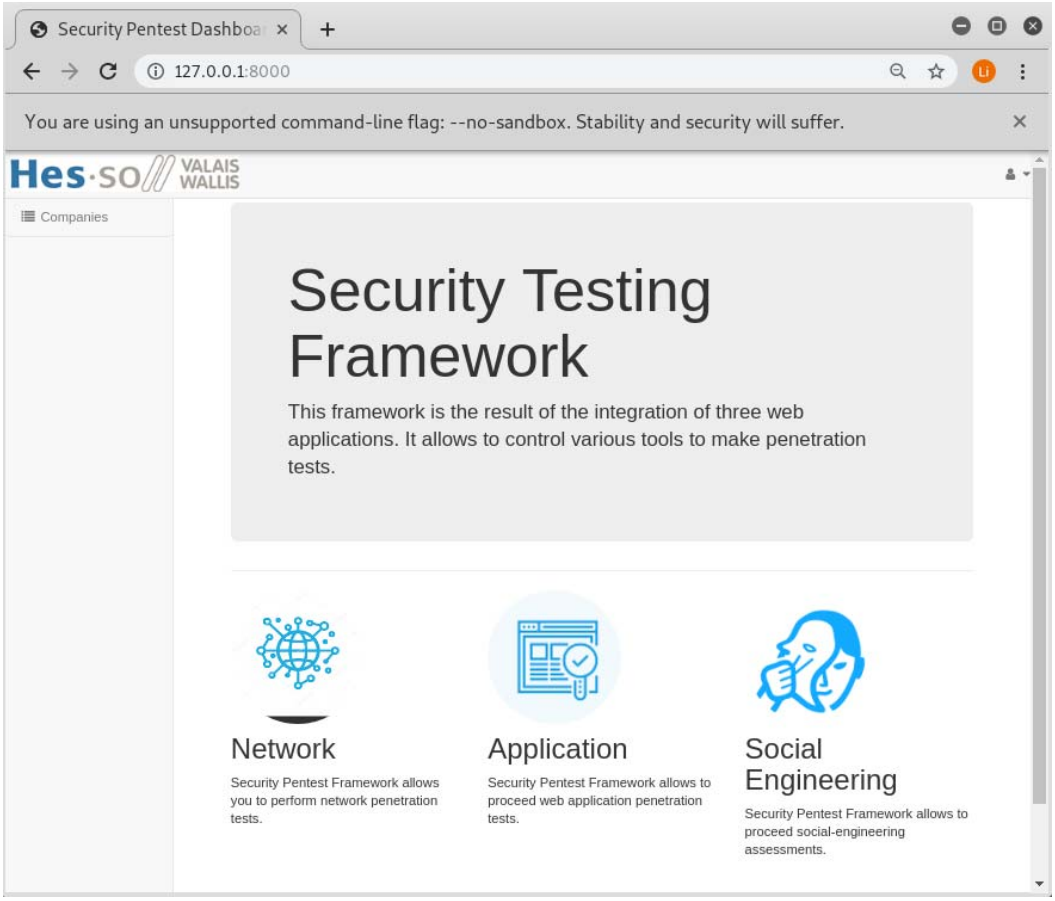


Figure 56 Page d'accueil

6.3. Liste des compagnies

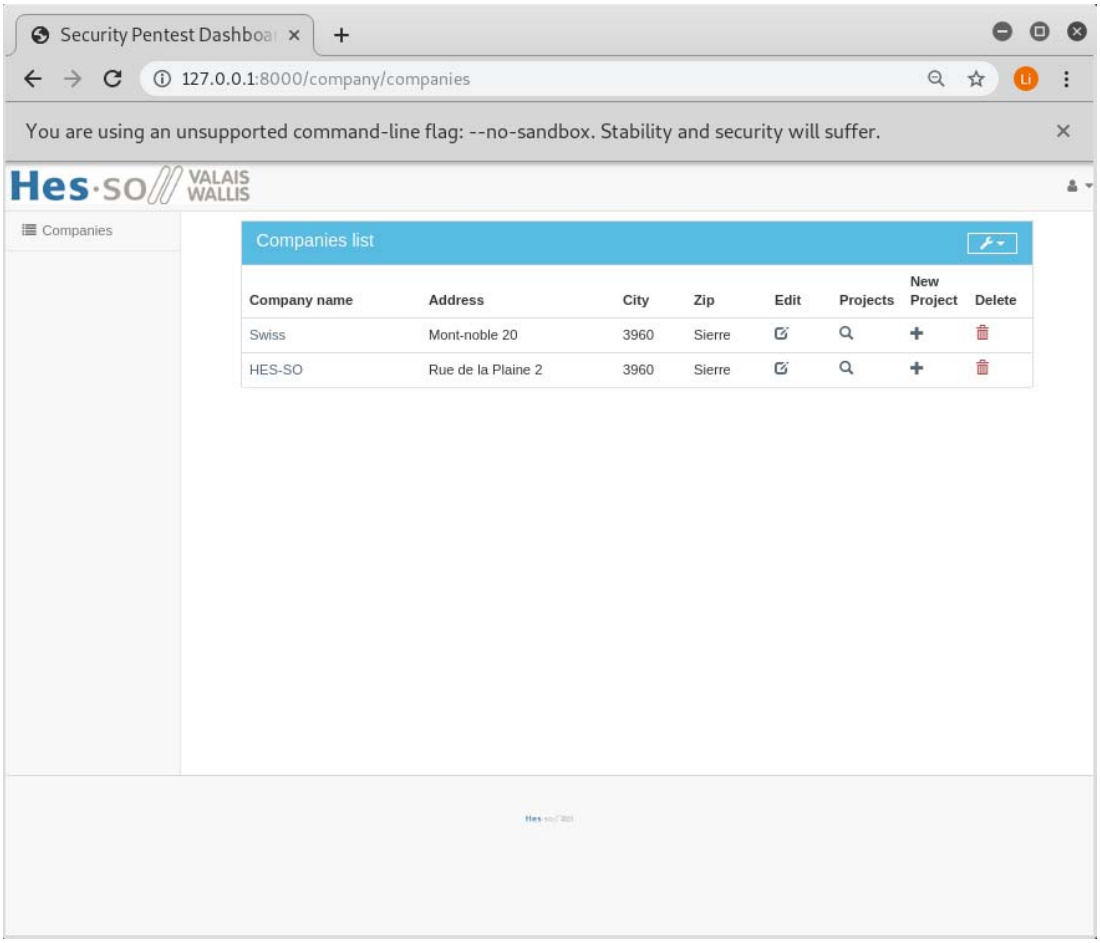


Figure 57 Pages de la liste des compagnies

6.4. Détails compagnie

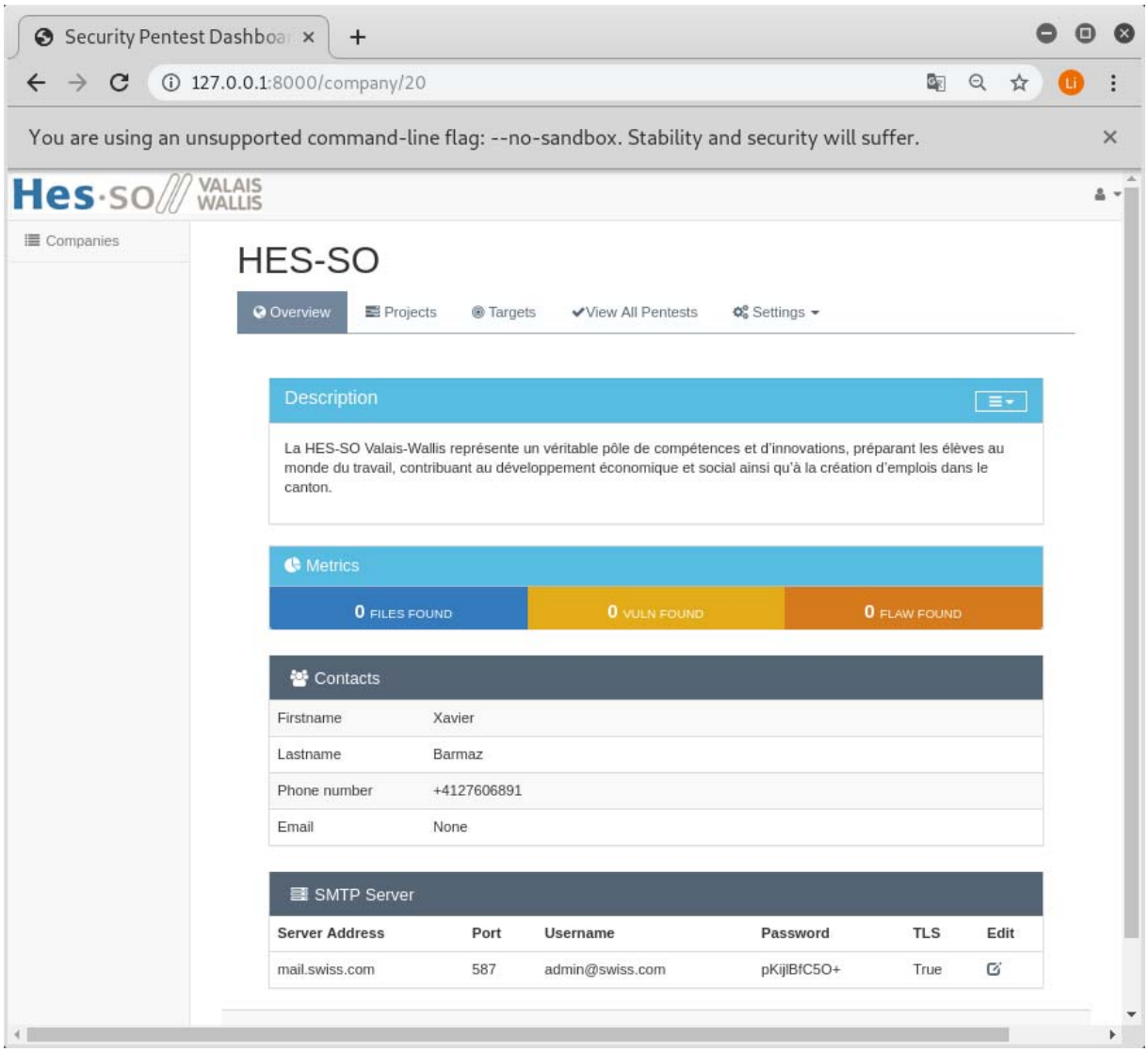


Figure 58 Page de détails de la compagnie

6.5. Projets d'une compagnie

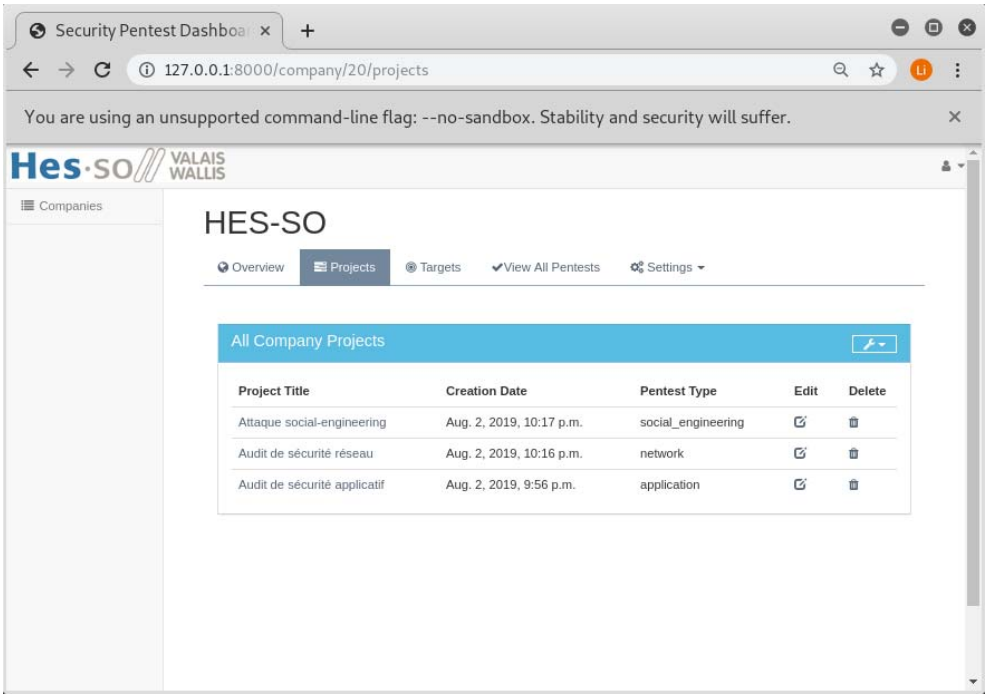


Figure 59 Page des projets d'une compagnie

6.6. Targets d'une compagnie

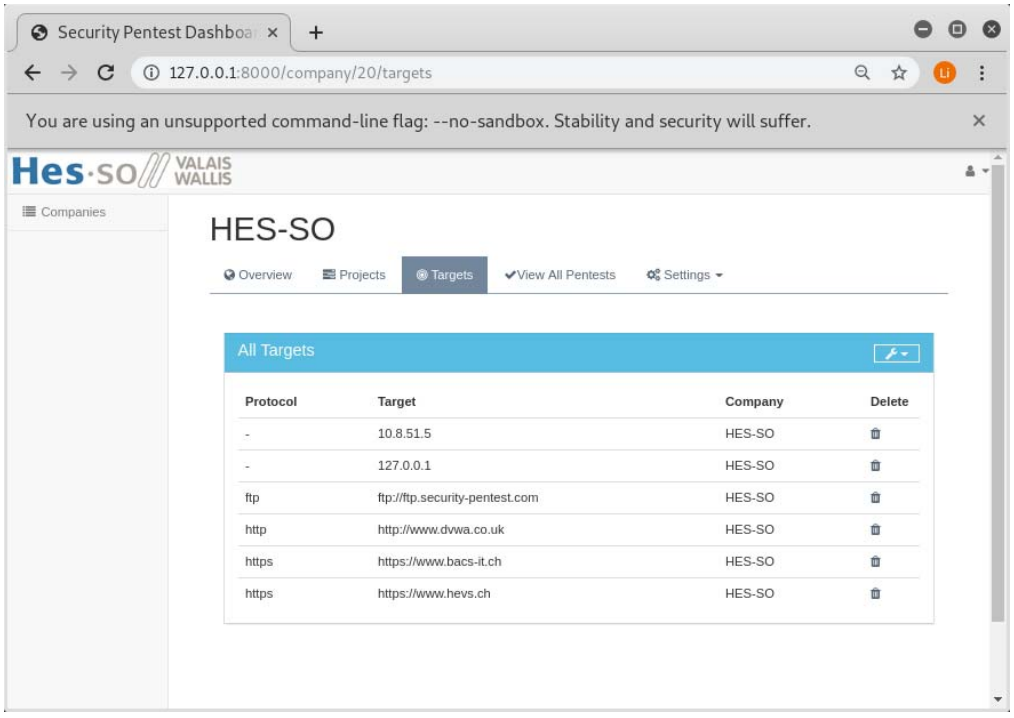


Figure 60 Parge des targets de la compagnie

6.7. Pentests de tous les projets

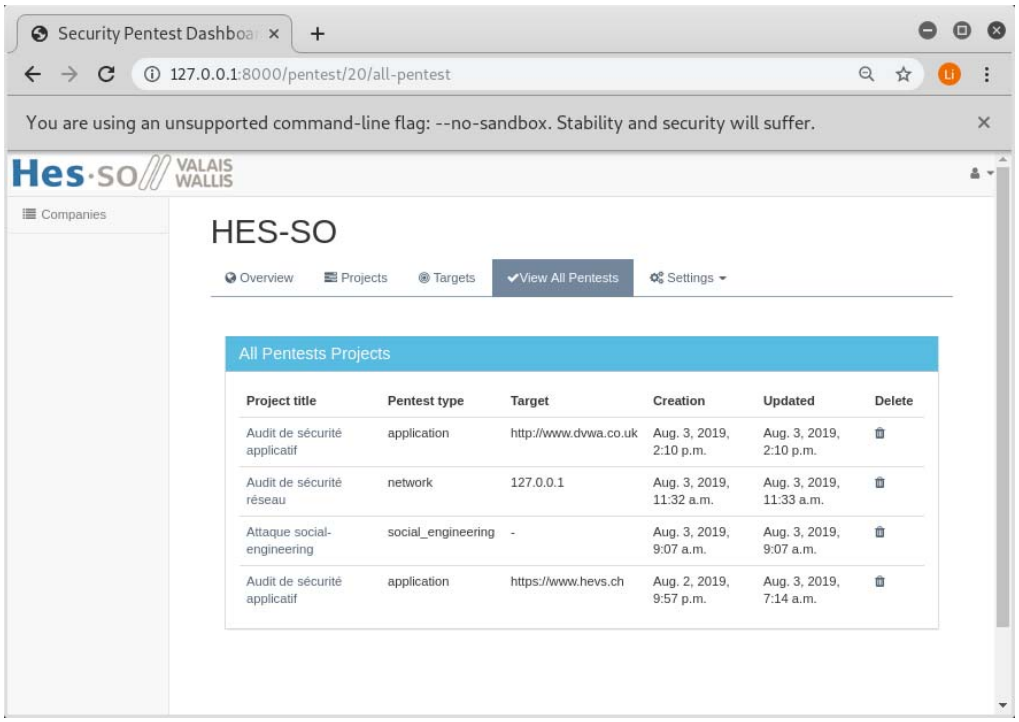


Figure 61 Page de tous les pentests d'une compagnie

6.8. Serveur de messagerie

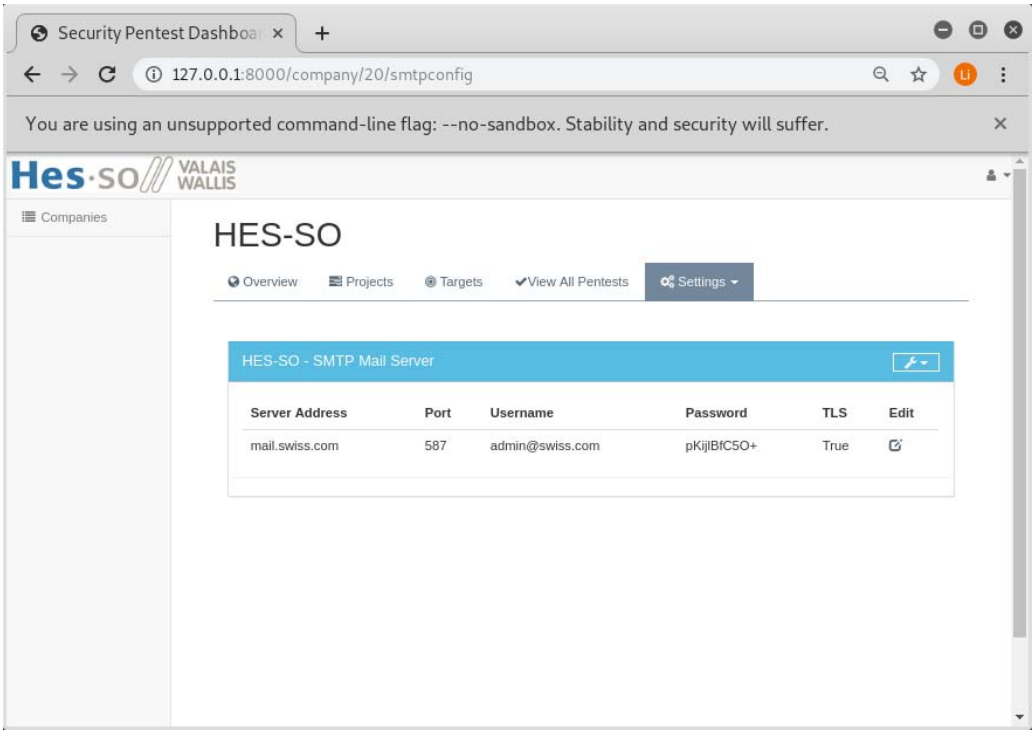


Figure 62 Page des serveurs de messagerie d'une compagnie

6.9. Pentests d'un projet

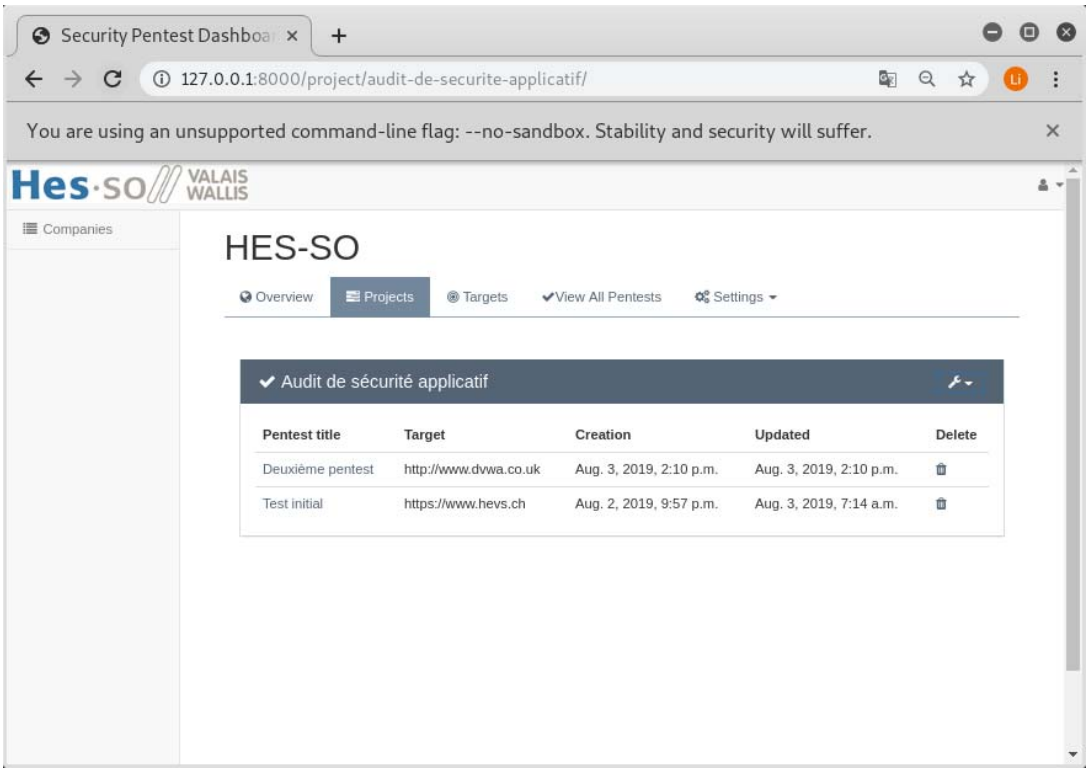


Figure 63 Page des pentests d'un projet

6.10. Dashboard des pentests

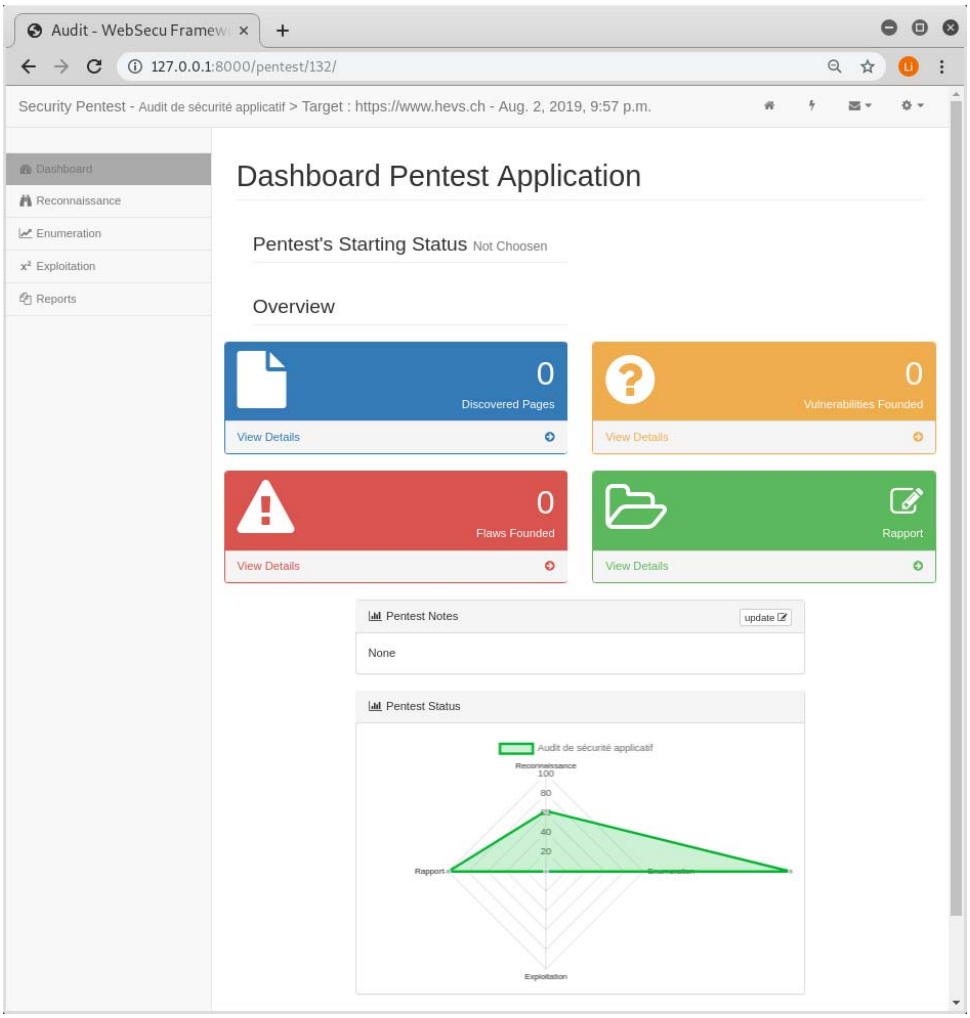


Figure 64 Page du dashboard d'un pentest type applicatif

6.11. Outils de pentests à l'étape reconnaissance

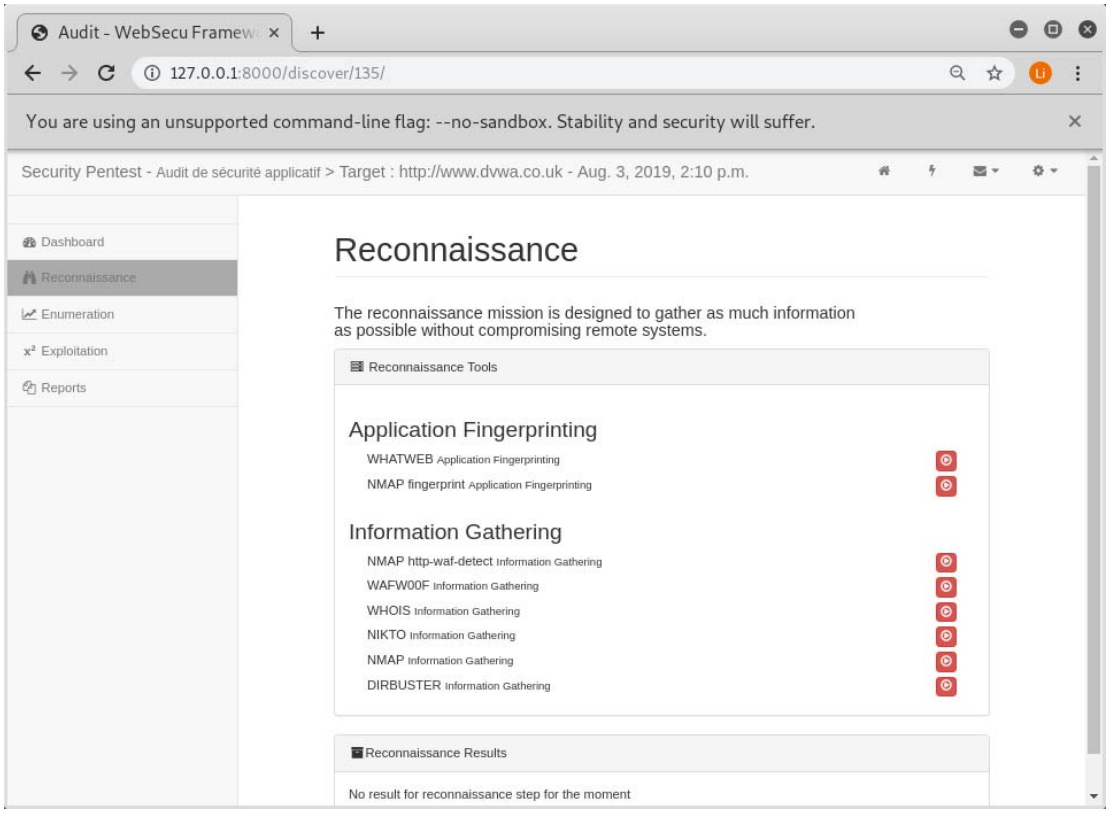


Figure 65 Page des outils de pentests applicatif

6.12. Rapport

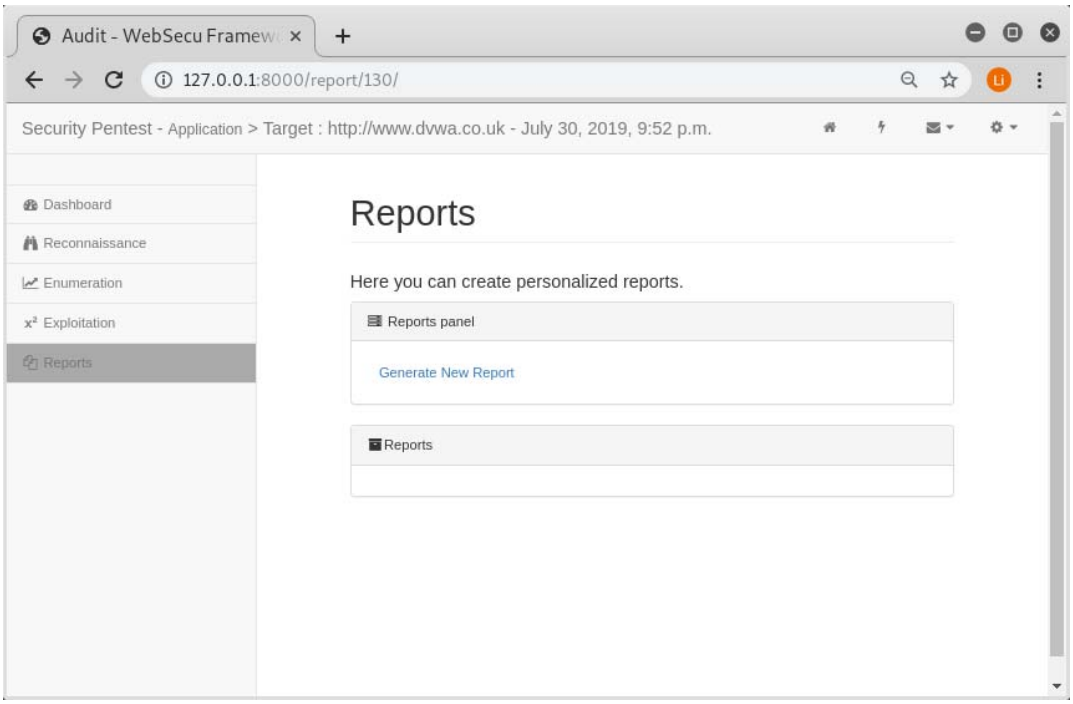


Figure 66 Page des rapports

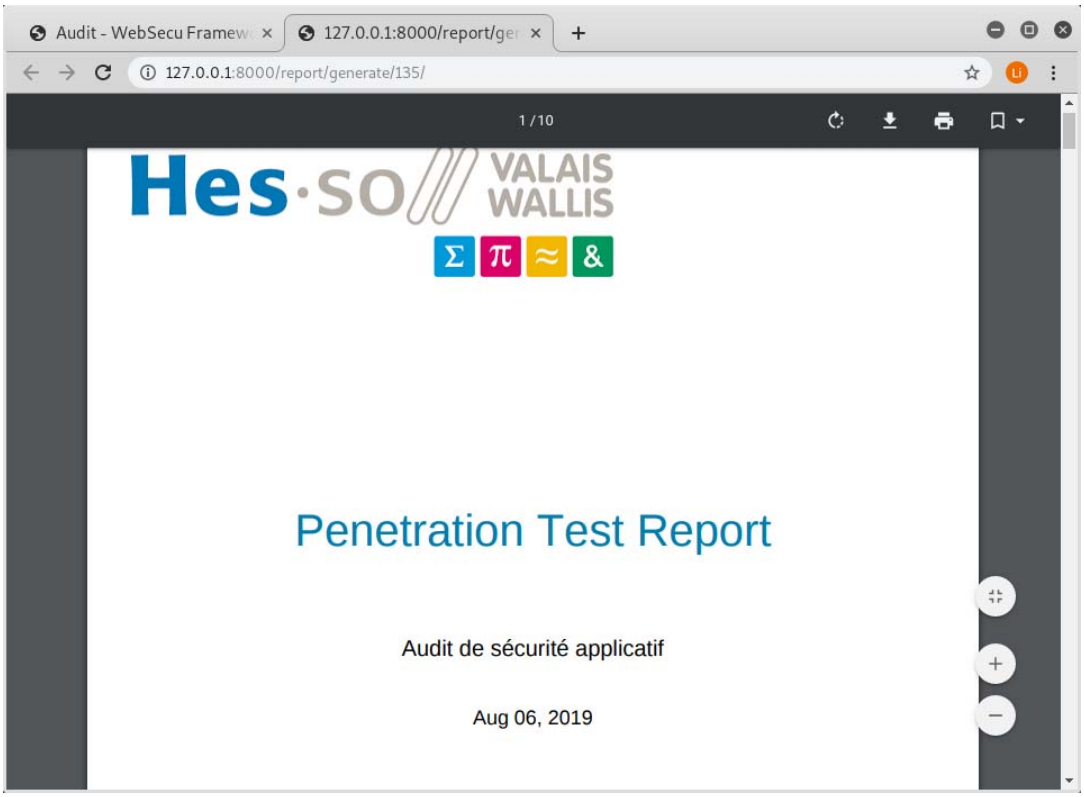


Figure 67 Rapport généré en PDF

6.13. Dashboard Pentest Réseau

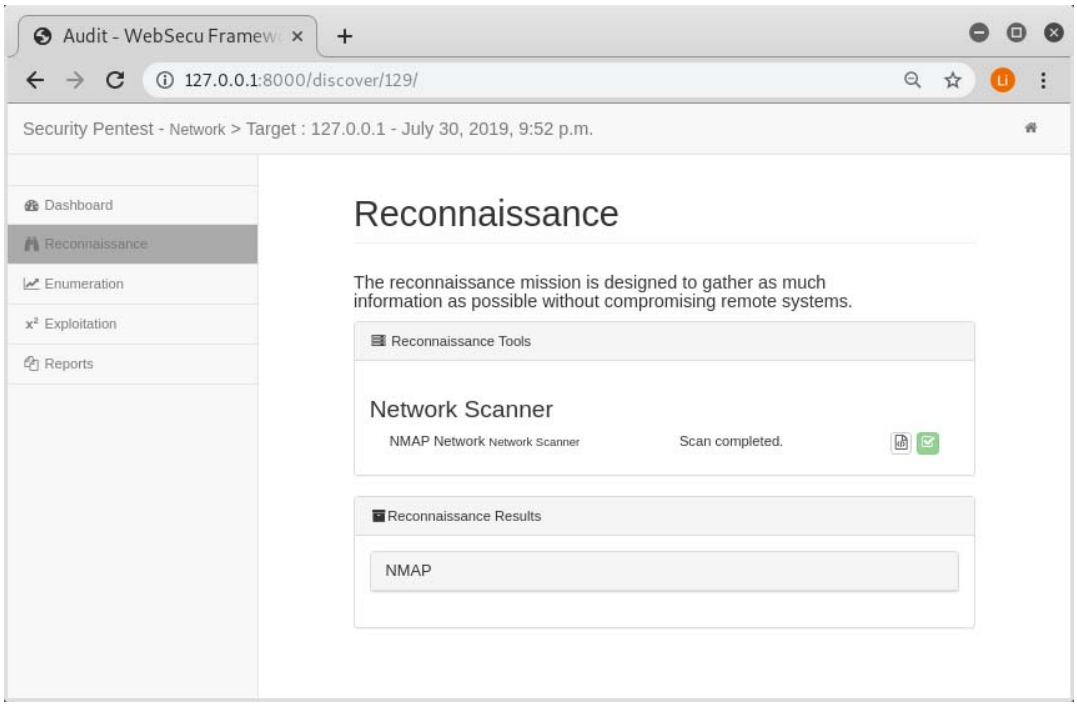


Figure 68 Pentest réseau phase reconnaissance

6.14. Dashboard Pentest Social-Engineering

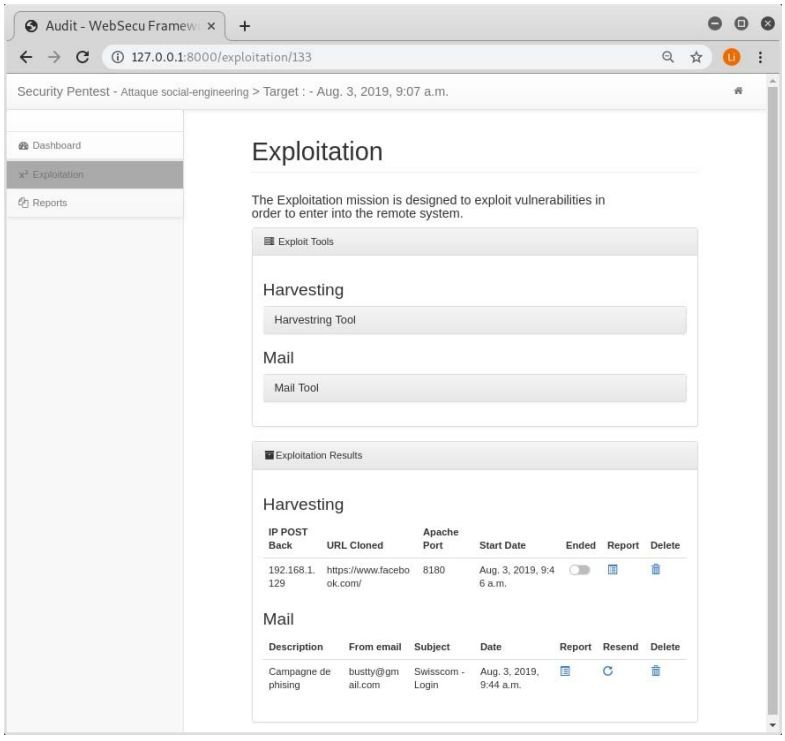


Figure 69 Dashboard social-engineering

6.15. Outil Credentials Harvester

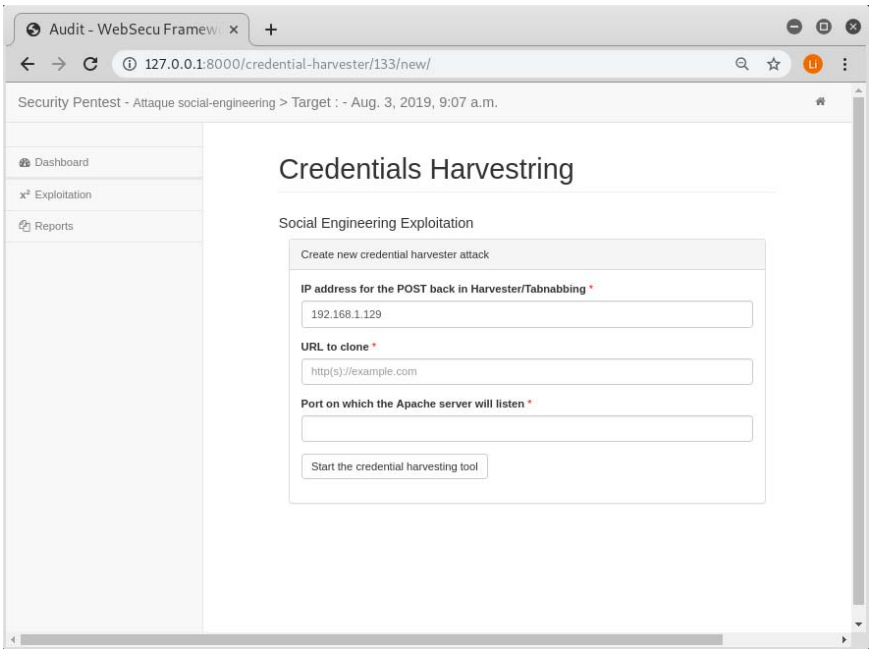


Figure 70 Page de l'outil Credential Harvester

6.16. Rapport Credentials Harvester

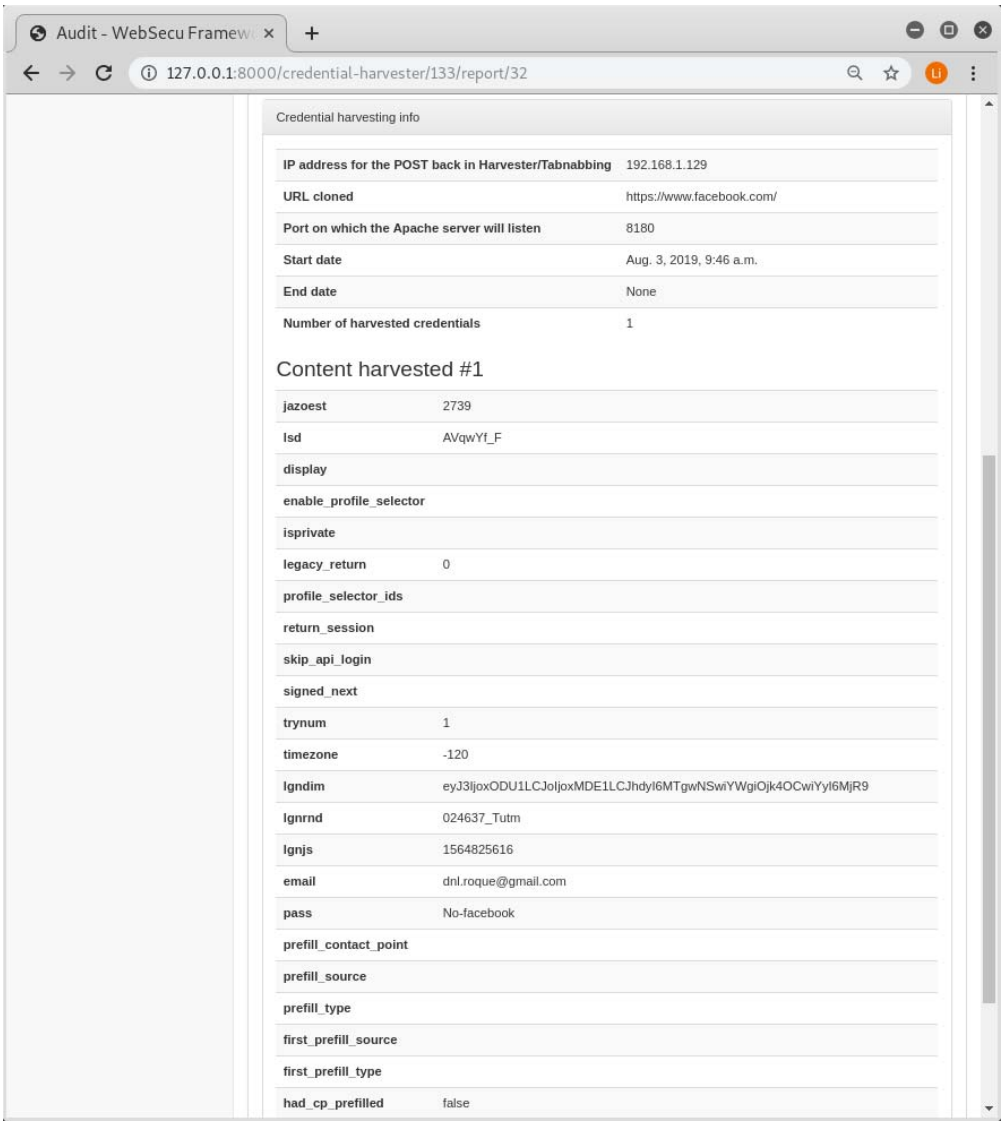


Figure 71 Page du rapport de l'attaque credentials harvester

6.17. Outil Mail

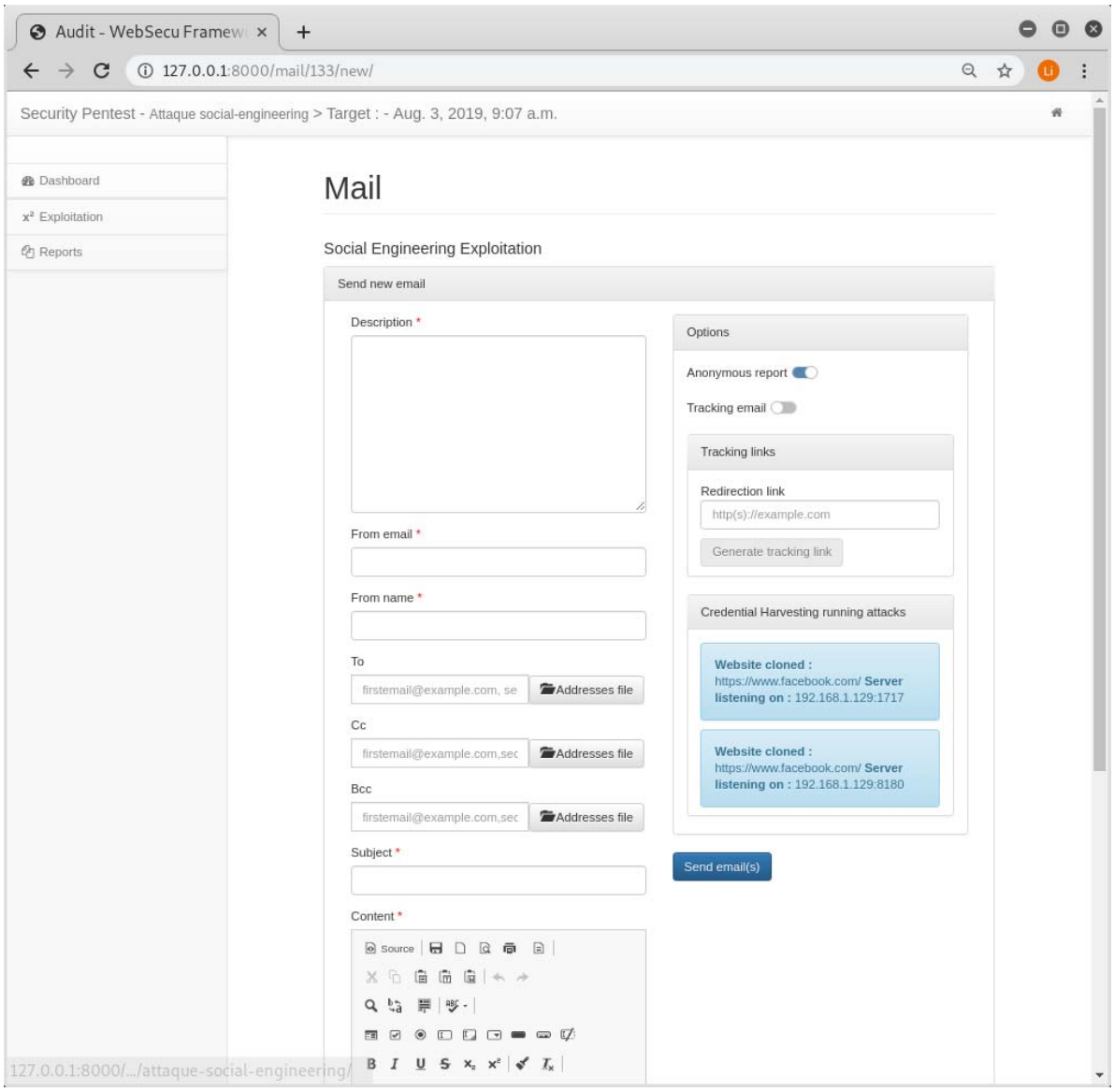


Figure 72 Page de l'outil Mail

7. Conclusion

Le résultat des recherches au début de ce travail a permis de mettre en perspective les frameworks open source et gratuits disponibles sur Internet et de constater qu'aucun ne permet de réaliser des pentests de type réseau, applicatif et social engineering. L'analyse nous a amenés à choisir les projets adéquats à une intégration pour répondre à ce besoin. Dans le concept d'intégration, les spécifications du projet et les étapes à réaliser ont été définies. De cette façon, une application web unifiée permettant de réaliser différents types de tests d'intrusions a pu voir le jour. De plus, le code source a été amélioré et suit les conventions du langage Python. Divers composants ont également été adaptés ou créés pour intégrer de nouvelles fonctionnalités. Finalement, la base de données a été consolidée.

L'automatisation des outils est une bonne chose, mais l'implémentation actuelle ne permet pas de personnaliser leur utilisation. En outre, il serait intéressant de repenser l'architecture de ce travail afin d'avoir une application Django par outil, qui permettrait d'utiliser toutes les options/commandes de l'outil. La génération du rapport pourrait aussi être plus flexible.

8. Références

8bitjunkie. (2018, mai 21). language agnostic—What is a callback function? Consulté 5 août 2019, à l'adresse Stack Overflow website: https://stackoverflow.com/questions/824234/what-is-a-callback-function

10Security. (2019). DefectDojo—Commercial Support. Consulté 6 avril 2019, à l'adresse DefectDojo – Commercial Support website: https://10security.com/services-by-technology/defectdojo-commercial-support/

Ask Solem. (2019). Tasks—Celery 4.3.0 documentation. Consulté 2 août 2019, à l'adresse https://docs.celeryproject.org/en/latest/userguide/tasks.html

Baloch, R. (s. d.). *Ethical Hacking and Penetration Testing Guide*. Auerbach Publications.

Bancal, D., Dumas, D., Puche, D., Ebel, F., Vicogne, F., Fortunato, G., … Lasson, S. (2017). *Ethical Hacking : Apprendre l'attaque pour mieux se défendre (5e édition)* (Editions ENI).

Baumgartner, P. (s. d.). Django Logging, The Right Way. Consulté 3 août 2019, à l'adresse Lincoln Loop website: https://lincolnloop.com/blog/django-logging-right-way/

Bobylev, S. (2016). *Pentest—Framework générique pour les tests d'intrusion*.

Ch, R., & el. (2018, octobre 15). Comprehensive Guide on Dirb Tool. Consulté 6 juillet 2019, à l'adresse Hacking Articles website: https://www.hackingarticles.in/comprehensive-guide-on-dirb-tool/

Cyberpunk. (2018, août 3). Collaborative Pentest & Vulnerability Management Platform—[Faraday]. Consulté 7 avril 2019, à l'adresse CYBERPUNK website: https://www.cyberpunk.rs/collaborative-pentest-vulnerability-management-platform-faraday

Django Software Foundation. (2019a). Écriture de votre première application Django, 1ère partie | Documentation de Django | Django. Consulté 19 juillet 2019, à l'adresse https://docs.djangoproject.com/fr/2.2/intro/tutorial01/

Django Software Foundation. (2019b). Gestion des fichiers statiques (par ex. Images, JavaScript, CSS) | Documentation de Django | Django. Consulté 5 août 2019, à l'adresse https://docs.djangoproject.com/fr/2.2/howto/static-files/

Django Software Foundation. (2019c). Le site d'administration de Django | Documentation de Django | Django. Consulté 3 août 2019, à l'adresse https://docs.djangoproject.com/fr/2.2/ref/contrib/admin/

Druide informatique inc. (2015). *Antidote 9 - Correcteur Et Dictionnaires Pour Le Francais Ou L'anglais Druide Plate-forme : Windows 8, Windows 7, Windows Vista, MAC OS X*. Educa Books.

FaradaySEC. (2019). FaradaySEC | Penetration Testing Environment. Consulté 18 juillet 2019, à l'adresse Faraday website: https://www.faradaysec.com

Fournier, F. (2015). *Django : Industrialisez vos développements Python*. Paris: Editions ENI.

Functional testing. (2019). In WIkipedia (Éd.), *Wikipedia*. Consulté à l'adresse https://en.wikipedia.org/w/index.php?title=Functional_testing&oldid=888100048

Halton, W., Weaver, B., Ansari, J. A., Kotipalli, S. R., & Imran, M. A. (2017). *Penetration Testing : A Survival Guide*. Packt Publishing Ltd.

Hillard, D. (2019, juillet 9). Consolidating Multiple Django Projects—Easy as Python. Consulté 19 juillet 2019, à l'adresse Medium website: https://easyaspython.com/how-to-consolidate-multiple-django-projects-b7c9bb940a59

Horton, A. (s. d.). WhatWeb. Consulté 6 juillet 2019, à l'adresse MorningStar Security website: https://www.morningstarsecurity.com/research/whatweb

Institut d'électronique et d'informatique Gaspard-Monge. (2019). Nessus—Un scanner de vulnérabilité. Consulté 6 juillet 2019, à l'adresse Nessus—Scan website: http://igm.univ-mlv.fr/~dr/XPOSE2009/Nessus/nessus_scan.html

Johansson, L. (2015, mai 18). Part 1 : RabbitMQ for beginners - What is RabbitMQ? - CloudAMQP. Consulté 2 août 2019, à l'adresse https://www.cloudamqp.com/blog/2015-05-18-part1-rabbitmq-for-beginners-what-is-rabbitmq.html

Juned Ahmed Ansari. (2015). *Web Penetration Testing with Kali Linux*. Packt Publishing Ltd.

Larousse, É. (2019). Définitions : Hacker - Dictionnaire de français Larousse. Consulté 4 août 2019, à l'adresse https://www.larousse.fr/dictionnaires/francais/hacker/38812

Le Goff, V. (2019, juin 26). De bonnes pratiques. Consulté 24 juillet 2019, à l'adresse OpenClassrooms website: https://openclassrooms.com/fr/courses/235344-apprenez-a-programmer-en-python/235263-de-bonnes-pratiques

Licence MIT. (2018). In Wikipedia (Éd.), *Wikipédia*. Consulté à l'adresse https://fr.wikipedia.org/w/index.php?title=Licence_MIT&oldid=149216437

Lopez, M. (2017). *Framework générique et unifié pour les tests d'intrusion applicatifs*.

Marques, D. (2017). *Framework générique et unifié de social engineering*.

Mattiocco, N. (2019). PatrOwl. Consulté 13 avril 2019, à l'adresse PatrOwl website: https://www.patrowl.io

Mattiocco, N. (2019). *PatrOwl : Open Source, Smart and Scalable Security Operations Orchestration Platform - Patrowl/PatrowlManager* [HTML]. Consulté à l'adresse https://github.com/Patrowl/PatrowlManager (Original work published 2018)

Montilva, J. (2019). *Collaborative Penetration Test and Vulnerability Management Platform : Infobyte/faraday* [Python]. Consulté à l'adresse https://github.com/infobyte/faraday (Original work published 2013)

Offensive Security. (2019a). Amap. Consulté 6 juillet 2019, à l'adresse Amap Package Description website: https://tools.kali.org/information-gathering/amap

Offensive Security. (2019b). Kali Linux Tools Listing. Consulté 7 avril 2019, à l'adresse https://tools.kali.org/tools-listing

Ogma-sec. (2015, octobre 12). Test d'intrusion - Pentest : Présentation et méthodologies. Consulté 4 août 2019, à l'adresse Ogma-Sec website: https://ogma-sec.fr/test-dintrusion-pentest-presentation-methodologies/

Open Web Application Security Project. (2018). In Wikipedia (Éd.), *Wikipédia*. Consulté à l'adresse https://fr.wikipedia.org/w/index.php?title=Open_Web_Application_Security_Project&oldid=154112229

ORSYS. (2019). Formation : Technologies numériques, management, développement personnel, gestion de projets, RH, finance, marketing, commercial, droit, achat, ... Consulté 4 août 2019, à l'adresse https://www.orsys.com/

OWASP. (s. d.). OWASP Zed Attack Proxy Project—OWASP. Consulté 6 juillet 2019, à l'adresse https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

Paul. (2019, janvier 28). A Complete Guide to Nmap | Nmap Tutorial. Consulté 4 août 2019, à l'adresse Edureka website: https://www.edureka.co/blog/nmap-tutorial/

Python Software Foundation. (2019). 12. Environnements virtuels et paquets—Documentation Python 3.7.4. Consulté 19 juillet 2019, à l'adresse https://docs.python.org/fr/3/tutorial/venv.html

Rais, A. A. (2016). Interface-based software integration. *Journal of Systems Integration*, 79-88. https://doi.org/10.20470/jsi.v7i3.261

Rapid7. (2019). Security Orchestration and Automation for Security Operations. Consulté 18 juillet 2019, à l'adresse Rapid7 website: https://www.rapid7.com/products/komand/

Richmond, R. (2011, avril 2). The RSA Hack : How They Did It. Consulté 3 août 2019, à l'adresse Bits Blog website: https://bits.blogs.nytimes.com/2011/04/02/the-rsa-hack-how-they-did-it/

RSA Security. (2019). Solutions RSA de gestion des risques numériques et de cybersécurité. Consulté 3 août 2019, à l'adresse RSA.com website: https://www.rsa.com/fr-fr/index

Sarosys LLC. (2019). Arachni—Web Application Security Scanner Framework. Consulté 4 août 2019, à l'adresse Arachni—Web Application Security Scanner Framework website: https://www.arachni-scanner.com/

Schwartz, A. (2017). Microservices. *Informatik-Spektrum*, *40*(6), 590-594. https://doi.org/10.1007/s00287-017-1078-6

Seydtaghia, A. (2019, janvier 10). *«La croissance des données sera exponentielle»*. Consulté à l'adresse https://www.letemps.ch/economie/croissance-donnees-sera-exponentielle

Spencer, J. (2017, septembre 4). 0–100 in Django : Starting an app the right way. Consulté 1 août 2019, à l'adresse https://hackernoon.com/0-100-in-django-starting-an-app-the-right-way-badd141ef439

sqlmap.org. (2019). sqlmap : Automatic SQL injection and database takeover tool. Consulté 4 août 2019, à l'adresse sqlmap—Automatic SQL injection adn database takeover tool website: http://sqlmap.org/

Sri Manikanta, P. (2019, mars 6). Operating System for Penetration Testing in a Nutshell; Kali Linux vs Parrot Security OS. Consulté 2 août 2019, à l'adresse https://hackernoon.com/operating-system-for-penetration-testing-in-a-nutshell-kali-linux-vs-parrot-security-os-384809e7b7ae

The Django Book. (2019). Django's Structure – A Heretic's Eye View—Python Django Tutorials. Consulté 1 août 2019, à l'adresse The Django Book website: https://djangobook.com/mdj2-django-structure/

The PostgreSQL Global Development Group. (2019). PostgreSQL: The world's most advanced open source database. Consulté 2 août 2019, à l'adresse https://www.postgresql.org/

Thebaud, O. (2019, février 10). OpenVAS 9 – audit de vulnérabilités – Ressources sur la Sécurité des Systèmes d'Information. Consulté 6 juillet 2019, à l'adresse https://www.thebaud.com/openvas-audit-de-vulnerabilites/

Thomas BAILET. (2012). *Architecture logicielle—Pour une approche organisationnelle, fonctionnelle et technique (2e édition)*.

Tipton, H. F., & Krause, M. (2006). *Information Security Management Handbook*. CRC Press.

TrustedSec. (2019). The Social-Engineer Toolkit (SET) [The Social-Engineer Toolkit (SET)]. Consulté 4 août 2019, à l'adresse TrustedSec website: https://www.trustedsec.com/social-engineer-toolkit-set/

tutorialspoint.com. (2019). Penetration Testing Vs. Ethical Hacking. Consulté 4 août 2019, à l'adresse Www.tutorialspoint.com website: https://www.tutorialspoint.com/penetration_testing/penetration_testing_vs_ethical_hacking

van Rossum, G., & Warsaw, B. (2013, août 1). PEP 8—Style Guide for Python Code. Consulté 1 août 2019, à l'adresse Python.org website: https://www.python.org/dev/peps/pep-0008/

W3schools. (2019). What is Bootstrap. Consulté 2 août 2019, à l'adresse https://www.w3schools.com/whatis/whatis_bootstrap.asp

Weaver, A. (2019). DefectDojo | CI/CD and DevSecOps Automation. Consulté 6 avril 2019, à l'adresse https://www.defectdojo.org/

Wesleti, F. (2019). *PenBox : A Penetration Testing Framework - The Tool With All The Tools , The Hacker's Repo - x3omdax/PenBox* [Python]. Consulté à l'adresse https://github.com/x3omdax/PenBox (Original work published 2016)

Wikipedia. (2013). Moteur de workflow. In *Wikipédia*. Consulté à l'adresse https://fr.wikipedia.org/w/index.php?title=Moteur_de_workflow&oldid=96884103

Wikipedia. (2018a). Fuzzing. In *Wikipédia*. Consulté à l'adresse https://fr.wikipedia.org/w/index.php?title=Fuzzing&oldid=153161872

Wikipedia. (2018b). Nmap. In *Wikipédia*. Consulté à l'adresse https://fr.wikipedia.org/w/index.php?title=Nmap&oldid=151141205

Wikipedia. (2018c). Shellshock (faille informatique). In *Wikipédia*. Consulté à l'adresse https://fr.wikipedia.org/w/index.php?title=Shellshock_(faille_informatique)&oldid=150547241

Wikipedia. (2019a). Cross-site scripting. In *Wikipédia*. Consulté à l'adresse https://fr.wikipedia.org/w/index.php?title=Cross-site_scripting&oldid=160727073

Wikipedia. (2019b). Exploit (informatique). In *Wikipédia*. Consulté à l'adresse https://fr.wikipedia.org/w/index.php?title=Exploit_(informatique)&oldid=156020648

Wikipedia. (2019c). Fork. In *Wikipédia*. Consulté à l'adresse https://fr.wikipedia.org/w/index.php?title=Fork_(d%C3%A9veloppement_logiciel)&oldid=159587384

Wikipedia. (2019d). Lockheed Martin. In *Wikipédia*. Consulté à l'adresse https://fr.wikipedia.org/w/index.php?title=Lockheed_Martin&oldid=161066943

Wikipedia. (2019e). Robot d'indexation. In *Wikipédia*. Consulté à l'adresse https://fr.wikipedia.org/w/index.php?title=Robot_d%27indexation&oldid=156603359

Wiktionnaire. (2017, septembre 22). pentester—Wiktionnaire. Consulté 4 août 2019, à l'adresse https://fr.wiktionary.org/wiki/pentester

xsser.03c8.net. (s. d.). XSSer : Cross Site « Scripter ». Consulté 4 août 2019, à l'adresse XSSer—The Cross Site Scripting framework website: https://xsser.03c8.net/

Yarbrough, M. (2019, mai 27). 10 Popular Websites Built With Django. Consulté 26 juillet 2019, à l'adresse https://hackernoon.com/10-popular-websites-built-with-django-906cc310aa0a

Annexe I : code source redondant

```
20  @login_required
21  def exploit(request, id=None):
22      pentest = get_object_or_404(Pentest, id=id)
23      project = get_object_or_404(Project, id=pentest.project_id)
24      if project.pentestType == 'application':
25          target = get_object_or_404(Target, pentest=pentest)
26          pentest_tools = PentestTool.objects.filter(
27              tooltype="expl_app").order_by("soustype")
28          soustypes = SousType.objects.filter(
29              pentesttool__tooltype='expl_app').distinct()
30          # enumpentest = EnumerationModel.objects.filter(pentest=pentest) @not used
31          # pentestrecontools = PentestTool.objects.filter(
32          #     tooltype="reco_app").count()
33          # userecontools = ReconnaissanceModel.objects.filter(
34          #     pentest=pentest).exclude(
35          #     result__isnull=True).exclude(
36          #     result__exact='').count()
37          # pentestenumtools = PentestTool.objects.filter(
38          #     tooltype="enum_app").count()
39          # useenumtools = EnumerationModel.objects.filter(
40          #     pentest=pentest).exclude(
41          #     result__isnull=True).exclude(
42          #     result__exact='').count()
43          # reconpercent = userecontools * 100 / pentestrecontools @not used
44          # enumpercent = useenumtools * 100 / pentestenumtools @not used
45          # report = Report.objects.filter(pentest=pentest) @not used
46          scanresinjection = ScanResult.objects.filter(
47              enumMod__pentest=pentest).filter(
48              cweid=89).values('id', 'url', 'attack').distinct()
49          scanresxss = ScanResult.objects.filter(
50              enumMod__pentest=pentest).filter(
51              cweid=79).values('id', 'url', 'attack').distinct()
52
53          injectionflaws = SqlExploit.objects.filter(
54              exploitMod__pentest=pentest)
55          dbsfound = ExploitDataBase.objects.filter(
56              sqlExploit__exploitMod__pentest=pentest)
```

Figure 73 views.py de l'application exploitation

```python
@login_required
def report(request, id=None):
    pentest = get_object_or_404(Pentest, id=id)
    project = get_object_or_404(Project, id=pentest.project_id)
    report = Report.objects.filter(pentest=pentest)

    if project.pentestType != 'social_engineering':
        target = get_object_or_404(Target, pentest=pentest)
    else:
        target=None
    # @not used
    # userecontools = ReconnaissanceModel.objects.filter(pentest=obj).exclude(result__isnull=True).exclude(result__exact='').count()
    # useenumtools = EnumerationModel.objects.filter(pentest=obj).exclude(result__isnull=True).exclude(result__exact='').count()
    # pentestrecontools = PentestTool.objects.filter(tooltype="reco_app").count()
    # pentestenumtools = PentestTool.objects.filter(tooltype="enum_app").count()
    # reconpercent = userecontools * 100 /pentestrecontools
    # enumpercent = useenumtools * 100 / pentestenumtools
    # if report.count() < 1:
    #     pass

    context = {
        'pentest': pentest,
        'target': target,
        # "nbrecontool": reconpercent,
        # "nbenumtool": enumpercent,
        "report": report,
    }
    return render(request, "rapport/rapport.html", context)
```

Figure 74 Application Report fichier views.py

Annexe II : fichier de dépendances

requirements_file.txt

```
amqp==2.4.2
billiard==3.6.0.0
celery==4.3.0
certifi==2019.3.9
chardet==3.0.4
Django==1.11.10
django-bootstrap-static==4.0.0
django-celery-results==1.1.0
django-ckeditor==5.7.1
django-crispy-forms==1.7.2
django-datetime-widget==0.9.3
django-debug-toolbar==1.11
django-fontawesome==1.0
django-froala-editor==2.9.5
django-jquery==3.1.0
django-js-asset==1.2.2
django-registration-redux==2.6
django-staticfiles-fontawesome==2.0.0
html5lib==1.0.1
idna==2.8
kombu==4.5.0
pep8==1.7.1
Pillow==6.0.0
psycopg2==2.7.3.1
PyPDF2==1.26.0
python-nmap==0.6.1
python-owasp-zap-v2.4==0.0.14
pytz==2019.1
PyYAML==5.1
reportlab==3.4.0
requests==2.22.0
six==1.12.0
sqlparse==0.3.0
urllib3==1.25.3
vine==1.3.0
webencodings==0.5.1
xhtml2pdf==0.2.3
```

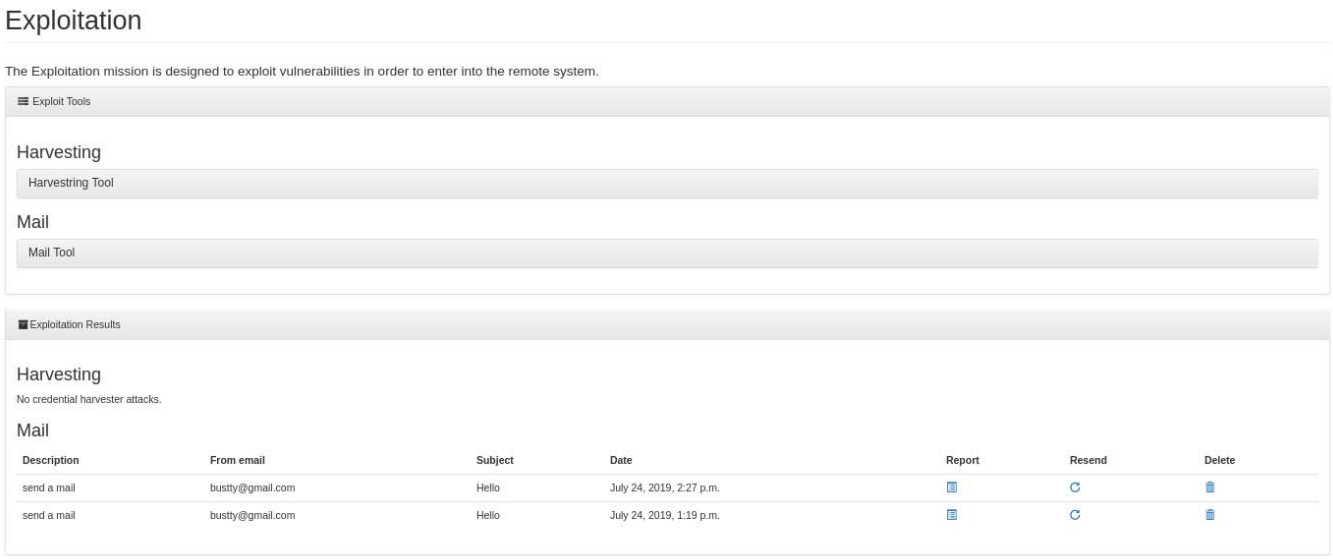
Figure 75 Dépendances du projet Application-PenTest

Annexe III : application du Framework Social Engineering

Name	Last commit
.idea	- Méthode ajax appelée lorsque l'on change le pr...
Installation files	bug fix
common	- Ajout champ email dans auditor
companies	Envoi du rapport par email
credential_harvester	- Ajout champ email dans auditor
files	Add new directory
mail	- Ajout champ email dans auditor
media	- Ajout d'une classe auditeur contenant ses infor...
reporting	send mail modif
social_engineering	Ajout de l'entité Auditor dans l'administration de ...
static	- Ajout champ email dans auditor
static_deploy	deploy
templates	- Affichage du nombre d'attaques contenues dan...
manage.py	Initial Commit

Figure 76 Structure du projet SocialEngineering-Pentest

Annexe IV : intégration de l'application mail au tableau de bord



Annexe V : Application des conventions PEP8

urls.py

```
1  urlpatterns = [
2      url(r'^(?P<id>\d+)/$', views.reconnaissance, name='reconnaissance'),
3      url(r'^reconresponse/(?P<id>\d+)/$', views.getReconResponses, name='getReconResponses'),
4  ...
```

Figure 77 urls.py non conforme PEP8

```
1  urlpatterns = [
2      url(r'^(?P<id>\d+)/$', views.discover, name='discover'),
3      url(r'^recon-response/(?P<id>\d+)/$', views.get_recon_responses,
4          name='get_recon_responses'),
5  ...
```

Figure 78 urls.py conforme PEP8

tasks.py

```
1  @app.task
2  def zapenumscantask(pentestid=None, enumid=None):
3      ...
```

Figure 79 tasks.py non conforme PEP8

```
1  @app.task
2  def zap_enum_scan_task(pentestid=None, enumid=None):
3      ...
```

Figure 80 tasks.py conforme PEP8

Annexe VI : code pour récupérer les outils de la phase du pentest

```
20  # Create your views here.
21  @login_required
22  def discover(request, id=None):
23      pentest = get_object_or_404(Pentest, id=id)
24      project = get_object_or_404(Project, id=pentest.project_id)
25      target = get_object_or_404(Target, pentest=pentest)
26
27      if project.pentestType == 'application':
28          pentest_tools = PentestTool.objects.filter(
29              tooltype="reco_app").order_by("soustype")
30          sous_types = SousType.objects.filter(
31              pentesttool__tooltype='reco_app').distinct()
32          reco_pentest = PentestPhases.objects.filter(pentest=pentest)
33          servers = Server.objects.filter(reconMod__pentest=pentest)
34          what_webs = Whatweb.objects.filter(reconMod__pentest=pentest)
35          what_webs_ids = Whatweb.objects.filter(
36              reconMod__pentest=pentest).values_list('id', flat=True)
37          techs = Technology.objects.filter(whatweb__id__in=what_webs_ids)
38          urls = Reconurl.objects.filter(reconMod__pentest=pentest)
39          niktos = Nikto.objects.filter(reconMod__pentest=pentest)
40          status = []
41          for tool in pentest_tools:
42              temp = "none"
43              for recon in reco_pentest:
44                  if tool.id == recon.pentesttool.id:
45                      if recon.result == "" or recon.result is None:
46                          if temp == "none":
47                              temp = "running"
48                      if recon.result != "":
49                          if temp == "none":
50                              temp = "finished"
51              status.append([tool.id, temp])
52
53          context = {
54              "target": target,
55              "pentest": pentest,
56              "tools": pentest_tools,
57              "soustypes": sous_types,
58              "reconpentest": reco_pentest,
59              "status": status,
60              "whatwebs": what_webs,
61              "techs": techs,
62              "urls": urls,
63              "niktos": niktos,
```

Figure 81 Application discover - views.py

Annexe VII: code pour lancer un outil de pentest

```python
@login_required
def launch_reco(request, id=None):
    if request.user.is_authenticated:
        if request.method == "POST":
            isajax = request.POST["ajax"]
            if(isajax):
                ajax_response = {}
                pentest = get_object_or_404(Pentest, id=id)
                tool = request.POST["tool"]
                pentesttool = get_object_or_404(PentestTool, id=tool)
                instance = PentestPhases()
                instance.pentest = pentest
                instance.phase_type = 'reco'
                instance.pentesttool = pentesttool
                instance.command = "temp"
                instance.save()
                logger.info("lauch_reco : " + str(instance.pentesttool.displayname))
                if(instance.pentesttool.displayname == 'NMAP Network'):
                    nmap_reco_server_scan_task.delay(pentest.id, instance.id)
                elif(instance.pentesttool.displayname == 'NMAP fingerprint'):
                    nmap_reco_waf_identif_task.delay(pentest.id, instance.id)
                elif(instance.pentesttool.displayname == 'NMAP http-waf-detect'):
                    nmap_reco_http_waf_detect_task.delay(pentest.id, instance.id)
                elif(instance.pentesttool.displayname == 'WAFW00F'):
                    wafwoof_detect_task.delay(pentest.id, instance.id)
                elif(instance.pentesttool.displayname == 'WHOIS'):
                    whois_reco_task.delay(pentest.id, instance.id)
                elif(instance.pentesttool.displayname == 'WHATWEB'):
                    whatweb_reco_task.delay(pentest.id, instance.id)
                elif(instance.pentesttool.displayname == 'DIRBUSTER'):
                    dirb_reco_task.delay(pentest.id, instance.id)
                elif(instance.pentesttool.displayname == 'NIKTO'):
                    nikto_reco_task.delay(pentest.id, instance.id)
                newrecon = serializers.serialize(
                    'json', PentestPhases.objects.filter(id=instance.id))
                ajax_response['response'] = newrecon
                return HttpResponse(json.dumps(ajax_response), content_type="application/json")
            return HttpResponse(json.dumps({"ajax": "this isn't ajax"}), content_type="application/json")
        return HttpResponse(json.dumps({"nothing to see": "this isn't happening"}), content_type="application/json")
    return redirect("auth_login")
```

Figure 82 Application discover exécution d'un outil- views.py

Annexe VIII : code pour lancer une tâche Celery

Une tâche Celery est exécutée à l'aide de la méthode « .delay » et est annotée d'un décorateur « @app.task » ou « @shared_task ». La tâche exécute ensuite le sous-processus dans Kali Linux qui utilise l'outil. Dans l'illustration ci-dessous c'est l'outil « NMAP waf fingerprint » qui est utilisé.

```
91  @app.task
92  def nmap_reco_waf_identif_task(pentestid=None, recoid=None):
93      print("nmap_reco_waf_identif_task() started...")
94      pentest = get_object_or_404(Pentest, id=pentestid)
95      target = get_object_or_404(Target, id=pentest.target_id)
96      recon = get_object_or_404(PentestPhases, id=recoid)
97      hosts = Host.objects.filter(reconMod__pentest=pentest)
98      logger.info("target host: " + str(target.host))
99      targetUrl = target.host
100     try:
101         result = subprocess.check_output('nmap --script=http-waf-fingerprint ' + targetUrl, shell=True)
102         recon.command = "nmap --script=http-waf-fingerprint " + targetUrl
103         recon.phase_type = 'reco'
104         recon.result = result
105         recon.save()
106         for host in hosts:
107             print("ebom")
108         pentest.reconnaissanceResult = True
109         pentest.save()
110         print("[+] nmap_reco_waf_identif_task() finished")
111     except Exception as e:
112         print('[-] Something bad happened during the scan: ' + str(e))
113         recon.delete()
114
```

Figure 83 Application discover - tasks.py

Annexe IX : code pour récupérer les résultats des outils

```python
@login_required
def get_recon_responses(request, id=None):
    ajax_response = {}
    if request.method == "POST":
        pentest = get_object_or_404(Pentest, id=id)
        reco_pentest = serializers.serialize(
            'json',
            PentestPhases.objects.filter(
                pentest=pentest), fields=('pentesttool', 'result'))
        pentestser = serializers.serialize(
            'json',
            Pentest.objects.filter(id=id), fields=('reconnaissanceResult'))
        whatwebser = serializers.serialize(
            'json',
            Whatweb.objects.filter(reconMod__pentest=pentest))
        whatwebsids = Whatweb.objects.filter(
            reconMod__pentest=pentest).values_list('id', flat=True)
        techser = serializers.serialize(
            'json',
            Technology.objects.filter(whatweb__id__in=whatwebsids))
        urlser = serializers.serialize(
            'json',
            Reconurl.objects.filter(reconMod__pentest=pentest))
        niktoser = serializers.serialize(
            'json',
            Nikto.objects.filter(reconMod__pentest=pentest))
        hosts = serializers.serialize(  # ajout pour le host tb
            'json',
            Host.objects.filter(reconMod__pentest=pentest))
        ports = serializers.serialize(  # ajout pour le port tb
            'json',
            Host.objects.filter(reconMod__pentest=pentest))
        ajax_response['reconpentest'] = reco_pentest
        ajax_response['pentest'] = pentestser
        ajax_response['whatwebs'] = whatwebser
        ajax_response['techs'] = techser
        ajax_response['urls'] = urlser
        ajax_response['niktos'] = niktoser
        ajax_response['hosts'] = hosts
        ajax_response['ports'] = ports
        return HttpResponse(json.dumps(ajax_response), content_type="application/json")
    return HttpResponse(json.dumps({"nothing to see": "this isn't happening"}), content_type="application/json")
```

Figure 84 Application discover - views.py get_recon_responses()

Annexe X: login et mot de passe des services

```
#
#
#This file contains the credentials informations about
#The configuration connections with db and rabbitmq server
#
#

##########################################################
#postgresql settings
DBNAME=''
DBUSER=''
DBPWD=''
#rabbitmq settings
RABBITVHOST=''
RABBITUSER=''
RABBITPWD=''
#zap Api Key
ZAPAPIKEY=''
##########################################################
```

Figure 85 credentials.py

```

#-------------------------------------------------#
#---------------OWN SETTINGS----------------------#
#-------------------------------------------------#
# postgresql settings
DBNAME = credentials.DBNAME
DBUSER = credentials.DBUSER
DBPWD = credentials.DBPWD
RABBITVHOST = credentials.RABBITVHOST
RABBITUSER = credentials.RABBITUSER
RABBITPWD = credentials.RABBITPWD
# ZAP API KEY
ZAPAPIKEY = credentials.ZAPAPIKEY
#-------------------------------------------------#
#-------------------------------------------------#
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/1.11/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'z98irr-++veu0rm#1(3x-e1%j8&0i)+cdsg#hl3h+&2^&#j0uo'

```

Figure 86 settings.py

DÉCLARATION DE L'AUTEUR

Je déclare, par ce document, que j'ai effectué le travail de Bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de Bachelor, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après :