Haute Ecole de Gestion & Tourisme
Hochschule für Wirtschaft & Tourismus

Filière Informatique de gestion
Studiengang Wirtschaftsinformatik

Travail de Bachelor 2017

Framework générique et unifié de social engineering

Source : https://www.expressvpn.com/internet-privacy/wp-content/uploads/sites/2/2015/12/Social-Engineering-Graphic-final.png

Étudiant : Dany Marques

Professeur : Xavier Barmaz

Déposé le : 09.08.2017

Résumé

L'objectif de ce travail est de fournir un framework générique et unifié permettant de générer des rapports ou audits sur la santé sécuritaire IT des entreprises sous l'angle du social engineering, en pilotant ou automatisant des outils Open-Source phares du marché ou en développement in house.

Afin de nous amener vers la réalisation de cet objectif, un état de l'art des frameworks de social engineering (SE), des formes de SE ainsi que des outils Open-Source permettant de réaliser le niveau de réponse de la cible visée face au social engineering est réalisé.

À l'aide de matrices décisionnelles, nous comparons les outils cités dans l'état de l'art afin d'en retenir les plus adaptés à ce projet. Au travers de cette étude, nous choisissons également le langage de programmation de l'outil ainsi que le framework à utiliser.

L'outil proposé est en mesure de générer des rapports anonymes ou nominatifs afin d'atteindre l'ultime but de ce projet, à savoir sensibiliser les utilisateurs de la société auditée aux risques auxquels ils s'exposent au quotidien en utilisant certains outils informatiques. Par rapport aux résultats de l'audit, la société peut organiser des séances de formation et/ou d'information afin que leurs employés ne reproduisent pas la même erreur.

Mots clés : social engineering, Django, Kali Linux, Python, SEToolkit

Avant-propos

De nos jours, les attaques de social engineering sont très courantes. Cette tendance est due au fait que la personne voulant réaliser une attaque de ce type n'ait pas besoin de connaissances informatiques pointues afin de se retrouver en possession d'informations confidentielles dérobées à un utilisateur. Nous pouvons donc en déduire que n'importe qui peut tenter une attaque de type social engineering.

Le but poursuivi par ce travail est celui de tester les employés d'une société sur leurs capacités à détecter ces attaques et par conséquent, à ne pas se faire piéger. Les résultats de ces tests sont ensuite présentés à qui de droit afin que des formations soient mises en place si nécessaire. Pour ce faire, l'outil réalisé dans ce travail couvre le spectre des attaques d'hameçonnage (phishing) ainsi que des récoltes d'identifiants (harvesting). Il peut donc servir comme point de départ à l'identification de faiblesses au sein de l'entreprise sous l'angle du social engineering.

Les raisons pour lesquelles j'ai choisi ce travail de Bachelor sont principalement l'actualité du sujet ainsi que la variété du type de travail (développement, système, sécurité).

Ce rapport est composé de toute l'analyse avant-projet ainsi que du détail du projet lui-même.

Structure CD annexé

En annexe à ce document, vous trouvez un CD avec le contenu suivant.

Figure 1 Structure CD

- 00_DonnéesTravailDeBachelor.pdf
  - Ce document contient les données du travail de Bachelor.
- 01_Rapport.pdf
  - Ce document contient le rapport du travail.
- 02_GuideInstallationUtilisateur.pdf
  - Ce document contient le guide d'installation pour l'utilisateur. Il détaille toutes les étapes nécessaires à l'installation de l'outil.
- 03_GuideInstallationDeveloppeur.pdf
  - Ce document contient le guide d'installation du développeur. Il est à suivre uniquement si le projet doit subir une modification.
- 04_JournalDesHeures.pdf
  - Ce document contient le journal des heures du projet.
- 05_Poster.pptx
  - Cette présentation contient le poster du travail.

Remerciements

Je tiens à remercier toutes les personnes ayant été présentes à mes côtés pendant toute la durée de ma formation et plus particulièrement M. Xavier Barmaz, professeur à la Haute École Spécialisée de Suisse Occidentale (HES-SO) pour m'avoir suivi, conseillé et corrigé durant le projet. De plus, je le remercie pour tout le temps qu'il a consacré à tester l'outil et à me rapporter les problèmes et les améliorations possibles.

Table des matières

1. Liste des tableaux

2. Liste des figures

3. Liste des abréviations

SE ..... Social Engineering

HES-SO ..... Haute École Spécialisée de Suisse Occidentale

SMS ..... Short Message Service

USB ..... Universal Serial Bus

CSV ..... Comma-separated values

AMI ..... Amazon Machine Image

SMTP ..... Simple Mail Transfer Protocol

SET ..... Social-Engineer Toolkit

Model-View-Template ..... MVT

Model-View-Controller ..... MVC

Object-relational mapping ..... ORM

Don't Repeat Yourself ..... DRY

Ruby on Rails ..... RoR

PHP Hypertext Preprocessor ..... PHP

Cross-Site Request Forgery ..... CSRF

Cross site scripting ..... XSS

Integrated Development Environment ..... IDE

Advanced Message Queuing Protocol ..... AMQP

HyperText Markup Language ..... HTML

Asynchronous Javascript and XML ..... AJAX

Simple Mail Transfer Protocol ..... SMTP

Mail Transfer Agent ..... MTA

Web Server Gateway Interface ..... WSGI

Cascading Style Sheets ..... CSS

Portable Document Format ..... PDF

CREATE, READ, UPDATE, DELETE ..... CRUD

Uniform Resource Locator ..... URL

What You See Is What You Get ..... WYSIWYG

Content Management System ..... CMS

4. Définitions

Baiting

« L'appâtage ou baiting, tel un véritable Cheval de Troie, utilise les médias physiques et exploite la curiosité de la cible. Il s'agit d'une escroquerie qui, sur bien des plans, est semblable à une attaque de phishing. Toutefois, ce qui distingue le baiting d'autres attaques d'ingénieries sociales, c'est la promesse d'un élément ou d'un bien utilisé par les pirates pour attirer leurs victimes. »

Phishing

« Technique frauduleuse utilisée par les pirates informatiques pour récupérer des informations (généralement bancaires) auprès d'internautes. »

Spear Phishing

« Le spear phishing désigne en sécurité informatique une variante de l'hameçonnage épaulée par des techniques d'ingénierie sociale. Contrairement à l'hameçonnage traditionnel basé sur l'envoi d'un message générique à un grand nombre de destinataires, le spear phishing se focalise sur un nombre limité d'utilisateurs (souvent un seul) auxquels est envoyé un message fortement personnalisé. »

Vishing

« Technique frauduleuse utilisée par les pirates pour récupérer des informations (généralement bancaires) auprès d'utilisateurs de téléphone portable. »

Smishing

« Phishing par SMS. »

Pretexting

« Le pretexting est une autre forme d'ingénierie sociale. Les assaillants cherchent à créer un prétexte adéquat ou à construire un scénario qu'ils peuvent utiliser pour tenter de voler des renseignements personnels à leurs victimes. Ces types d'attaques sont souvent orchestrées par un escroc qui prétend avoir besoin d'informations de la cible pour une confirmation d'identité. »

Quid Pro Quo

« Les attaques « quid pro quo » promettent un avantage en échange d'informations. Ces avantages ont souvent la forme d'un service alors que l'appât est souvent un bien. L'une des attaques « quid pro quo » les plus courantes est la suivante. Des fraudeurs se font passer pour des effectifs de services informatiques et passent de faux appels à des numéros directs de toutes les entreprises possibles. Ils proposent une assistance informatique à chacune de leurs victimes afin d'avoir un accès pour réaliser leurs propres actes malveillants. »

Tailgating

« Le tailgating ou piggybacking est un autre type d'attaques d'ingénierie sociale. Cette escroquerie implique que quelqu'un qui n'a pas l'authentification correcte suive un employé dans une zone

réglementée. »

Dumpster Diving

« De l'anglais « Dumpster diver » (littéralement : plongeurs de bennes) : ce sont des gens qui cherchent des choses que d'autres personnes ont jetées alors qu'elles sont encore utiles, peuvent être recyclées, et ont encore de la valeur. »

Harvest

Dans le sens premier du terme, cela signifie récolte. Dans ce travail, il sera plus spécifiquement utilisé pour la récolte d'information.

Hacker

« Personne qui, par jeu, goût du défi ou souci de notoriété, cherche à contourner les protections d'un logiciel, à s'introduire frauduleusement dans un système ou un réseau informatique. »

Plugin

« Plugiciel, module interagissant avec un programme principal. »

Benchmark

« Banc d'essai permettant de mesurer les performances d'un système pour le comparer à d'autres. »

Template

« Modèle de conception de logiciel ou de présentation des données. »

Responsive Design

« Conception de sites web adaptatifs. »

Payload

« Dans un virus ou parasite informatique, la charge active (PayLoad) est la partie du code qui est affectée à la fonction du virus ou du parasite, par opposition aux autres parties du code qui sont affectées à sa réplication, à sa propagation et à sa dissimulation. »

5. Introduction

Toute personne travaillant au sein d'une entreprise est en possession d'informations confidentielles. Si ces renseignements sont secrets, c'est qu'ils possèdent une valeur, économique ou non. Il faut donc impérativement que ces derniers soient en sécurité. La plupart des entreprises ont mis en place des systèmes de sécurité tels que des pare-feu, des proxys ou encore l'authentification des utilisateurs par mot de passe afin de protéger leurs données.

Vue de l'extérieur, une personne mal intentionnée, à l'instar d'un pirate informatique, peut exploiter deux points d'entrée principaux : les failles de sécurité liées aux systèmes informatiques utilisés et les relatives faiblesses liées à l'employé. C'est sur ce deuxième point que ce travail se concentre.

Définissons dans un premier temps la notion de social engineering ou, en français, l'ingénierie sociale : il s'agit d'un type d'attaque utilisée par les hackers dans le but dérober des informations leur permettant d'avoir un accès non autorisé sur des ordinateurs ou des informations confidentielles. Les attaques de SE ont comme particularité de tromper ou manipuler les humains afin d'obtenir des informations plus ou moins importantes en leur possession.

Afin d'illustrer cette problématique de social engineering en entreprise notamment, ce travail présente une solution développée et mise en place dans le but de tester les employés afin d'identifier le danger qu'ils représentent pour la société.

La première partie de ce rapport est dédiée à l'état de l'art. Celui-ci se focalise tout d'abord sur les frameworks de social engineering. La suite de cet état de l'art rapporte sur les formes de SE, les outils de SE ainsi que sur les frameworks de développement.

La seconde partie, quant à elle, porte principalement sur des matrices décisionnelles visant à faire des choix d'après certains critères définis.

La section suivante détaille le fonctionnement intégral du produit.

Pour finir, une conclusion sur la totalité du projet ainsi qu'une liste d'améliorations possibles sont proposées.

6. État de l'art

6.1. Frameworks de social engineering

Actuellement, il existe sur le marché très peu de frameworks permettant d'effectuer des simulations d'attaques de social engineering. Ci-dessous, nous parcourons les fonctionnalités des plus connus.

6.1.1. Wombat Security

Figure 2 Wombat Security

Source : www.wombatsecurity.com

Cette société propose différents outils permettant de sensibiliser les utilisateurs aux risques informatiques. Dans le domaine du SE, elle semble être la plus compétente. Wombat propose, entre autres, les outils d'ingénierie sociale suivants :

- CyberStrength Knowledge Assessments
  - Cet outil est un simple questionnaire proposant aux utilisateurs divers scénarios dans le but de connaître si les employés sont suffisamment formés.
- ThreatSim
  - Ce logiciel permet d'envoyer des emails d'hameçonnage aux utilisateurs et permet de récolter les résultats.
- SmishGuru
  - Très semblable à l'outil précédent, il permet lui aussi l'envoi de messages aux utilisateurs, toutefois, il utilise le Short Message Service (SMS) comme canal de distribution.
- USBGuru
  - Ce sont des clés Universal Serial Bus (USB) contenant un logiciel s'exécutant automatiquement lorsqu'elles sont insérées dans un ordinateur. Ce logiciel avise l'utilisateur qu'il ne doit pas introduire dans son ordinateur des périphériques provenant de sources inconnues.

L'entreprise a été fondée en 2008 et est toujours en fonction. Plus d'informations sur Wombat Security peuvent être trouvées à l'adresse www.wombatsecurity.com.

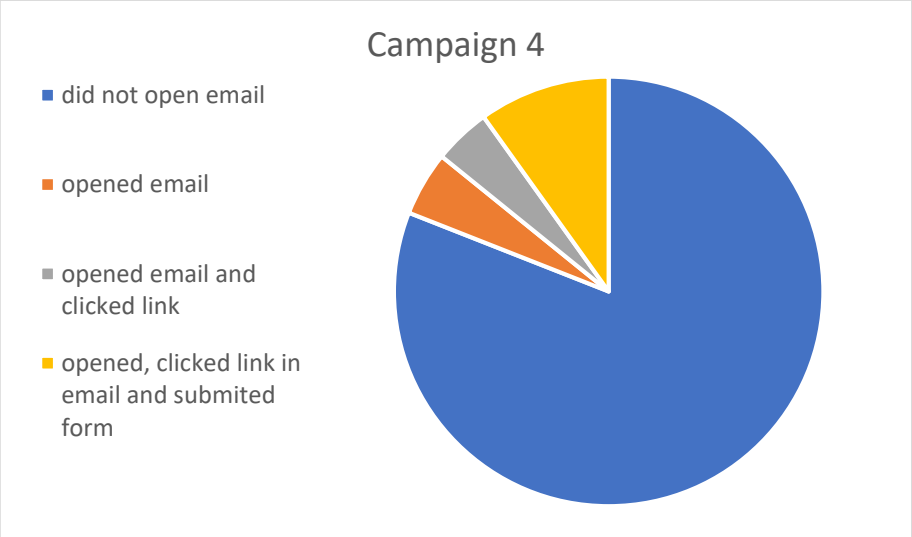
6.1.2. Inspired eLearning

Figure 3 Inspired eLearning

Source : www.inspiredelearning.com

Cette société propose des solutions sécuritaires s'exécutant sur une durée déterminée afin de pouvoir constater l'évolution entre les différentes analyses menées. Elle met principalement en place des formations sur le social engineering et propose également d'envoyer aux utilisateurs des rappels hebdomadaires sur les précautions à prendre. Cependant, elle propose uniquement un outil de phishing permettant de récupérer des métriques.

Figure 4 Inspired eLearning phishing report



Source : www.youtube.com/watch?v=C6iCW-pgPbM

L'entreprise a été fondée en 2003 et est toujours en fonction. Plus d'informations sur Inspired eLearning peuvent être trouvées à l'adresse www.inspiredelearning.com.

6.1.3. KnowBe4

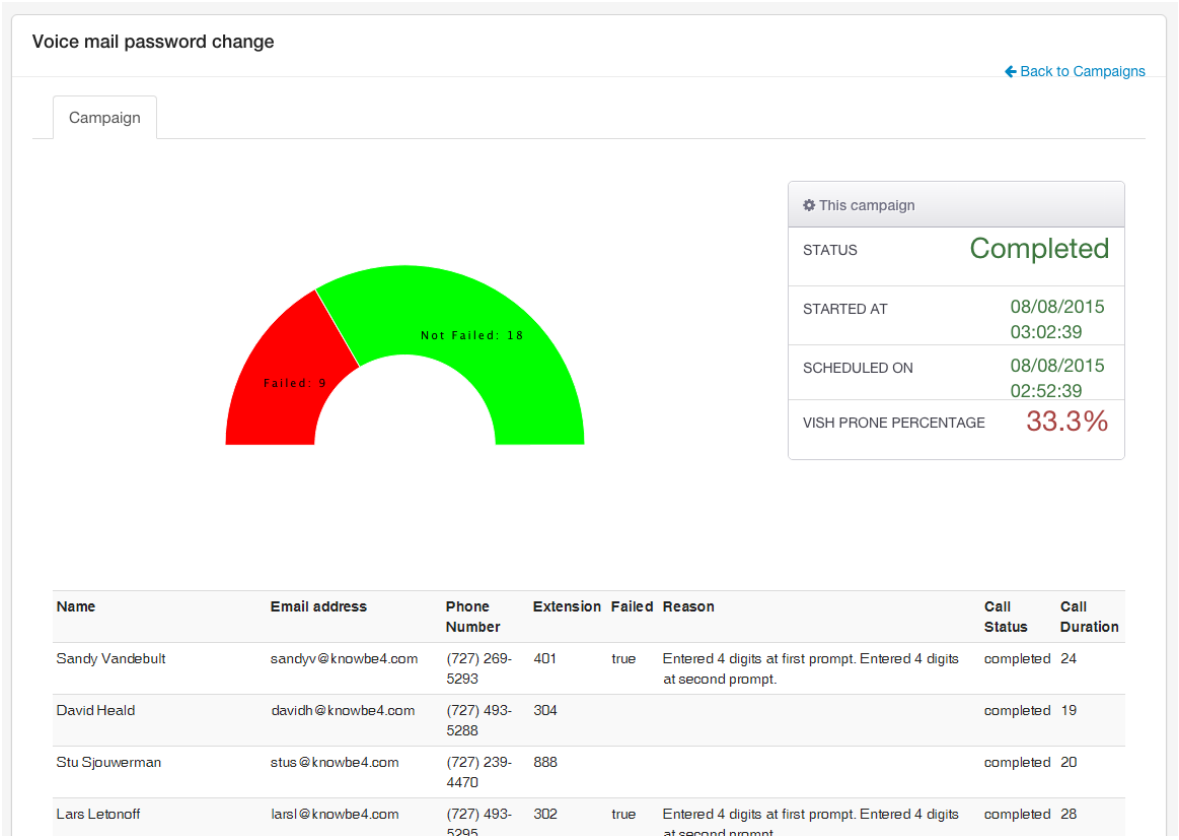
Figure 5 KnowBe4

Source : www.knowbe4.com

KnowBe4 met l'accent sur la formation continue des utilisateurs en proposant d'exécuter des tests puis d'analyser les résultats. Ces résultats servent à proposer des formations aux utilisateurs. D'autres

tests sont par la suite effectués et les résultats sont comparés afin de découvrir le degré d'évolution. KnowBe4 met également en avant un outil de phishing. De plus, ils proposent des tests de vishing basés sur des répondeurs prédéfinis et permettent l'analyse des résultats sous la forme suivante.

Figure 6 KnowBe4 Vishing report



Source : www.knowbe4.com/vishing

L'entreprise a été fondée en août 2010 et est toujours en fonction. Plus d'informations sur KnowBe4 peuvent être trouvées à l'adresse www.knowbe4.com.

6.1.4. Lucy

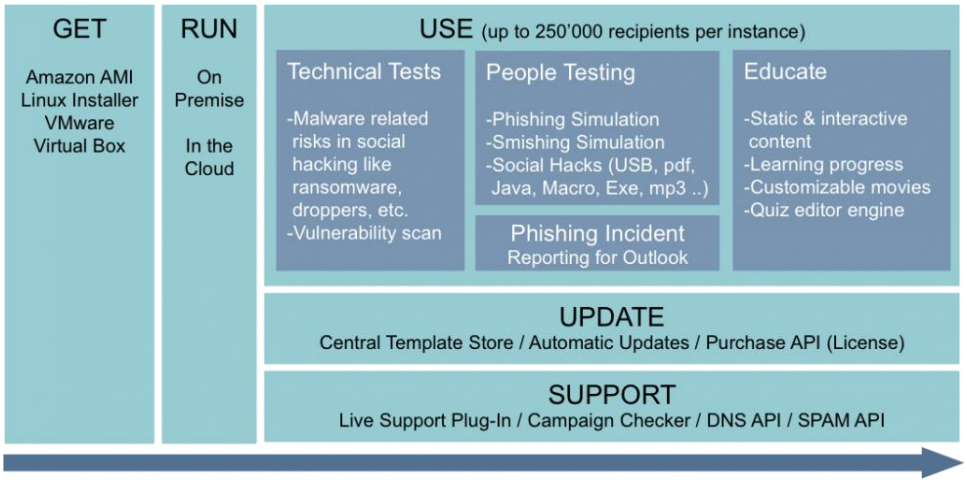
Figure 7 Lucy

Source : www.lucysecurity.com

Lucy Security propose une solution comprenant des outils de social engineering intéressante. Leur produit s'installe soit sur Amazon Machine Image (AMI), soit sur une machine Linux ou encore sur une

machine virtuelle de type VMWare[1] ou Virtual Box[2]. Une fois en fonction, nous pouvons exécuter des attaques de phishing, de smishing ou même d'autres types d'attaques sortant du cadre de l'ingénierie sociale. De plus, ils proposent des contenus afin de former les utilisateurs. L'intégralité des fonctionnalités sont schématisées sur l'image suivante.

Figure 8 LUCY in a nutshell



Source : www.lucysecurity.com

L'entreprise a été fondée en 1997 et est toujours en fonction. Elle a été créée pour aider l'industrie de la finance Suisse à lutter contre les attaques de cyber criminalité auxquelles elle était victime. Plus d'informations sur Lucy peuvent être trouvées à l'adresse www.lucysecurity.com.

6.2. Formes de social engineering

Dans ce chapitre, nous listons les formes les plus connues d'ingénierie sociale.

6.2.1. Baiting

Figure 9 Baiting Clé USB

Source : www.wyzguyscybersecurity.com/dont-take-bait

Le baiting, ou appât en français, peut être classifié d'attaques de social engineering comme

[1] https://www.vmware.com
[2] https://www.virtualbox.org

quelque chose donnant envie à la victime d'y toucher. Prenons l'exemple suivant : l'attaquant laisse volontairement trainer une clé USB dans un endroit où il est sûr qu'elle va être trouvée. Cette même clé USB sert d'appât pour la victime. Elle va probablement l'insérer dans son ordinateur afin de découvrir son contenu. Si ce contenu est exécuté d'une façon ou d'une autre, l'attaquant peut alors avoir accès sur l'ordinateur de la victime.

6.2.2. Phishing

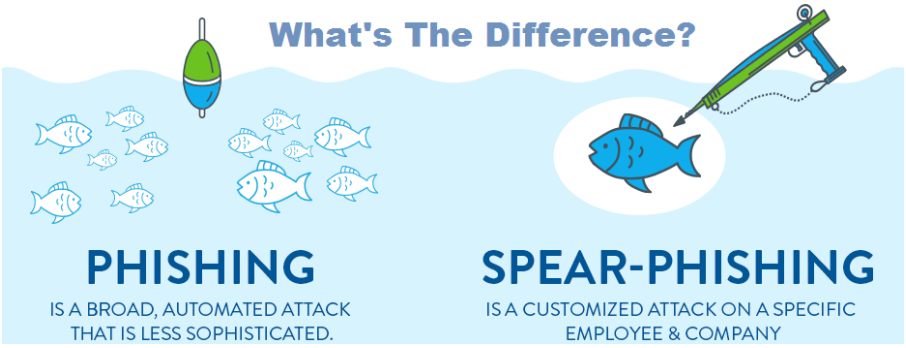
Figure 10 Phishing

Source : www.kylemarsh.co.uk/tag/social-engineering

Le phishing, ou hameçonnage en français, est le fait d'envoyer un email frauduleux en se faisant passer pour un email légitime. Le contenu du message sert à convaincre la victime d'installer un logiciel malveillant ou de fournir des informations confidentielles. Nous pouvons faire un lien entre le phishing et le baiting en prenant l'exemple suivant : l'attaquant envoie un faux email en indiquant à la victime qu'elle a remporté une somme d'argent dans un concours et lui demande ses coordonnées bancaires. Cette somme d'argent sert alors d'appât afin que le récepteur du message fournisse par exemple le numéro de sa carte de crédit ainsi que le code de sécurité. Très souvent, le contenu d'un email d'hameçonnage a un caractère d'urgence afin de stresser la victime et de lui laisser peu de temps de réflexion.

6.2.3. Spear phishing

Figure 11 Spear Phishing



Source : www.blog.nasafcu.com/2016/11/phishing-spear-phishing-yet-different

Le spear phishing est identique au phishing décrit ci-dessus. Il a la particularité d'être adapté à un individu ou une organisation tandis que le phishing se voit plus global.

6.2.4. Vishing

Figure 12 Vishing

Source : www.whatismyipaddress.com/vishing

Le vishing est, dans son but, très semblable au phishing. Cependant, les attaques de types vishing utilisent le canal de la téléphonie pour communiquer entre l'attaquant et la victime.

6.2.5. Smishing

Figure 13 Smishing

Source : www.computersecuritypgp.blogspot.ch/2017/04/what-is-smishing.html

Le smishing est, en autres mots, le phishing par SMS. Le contenu du message envoyé à la victime exige souvent une action à effectuer très rapidement. Ce message texte indique à la victime qu'elle doit appeler un certain numéro de téléphone ou qu'elle doit consulter un site Web

6.2.6. Pretexting

Figure 14 Pretexting

Source : www.lawtechnologytoday.org/2015/03/information-security-threat-social-engineering-and-the-human-element

Le pretexting est une autre forme utilisée par les ingénieurs sociaux afin d'avoir accès à des informations confidentielles. Les attaquants construisent tout un scénario bien réfléchi dans le but de ne laisser aucun doute à la victime quant à la véracité de l'histoire. Dans ce type d'attaques, l'hacker change son identité et se fait passer pour quelqu'un en qui la victime a confiance. Une fois le scénario prêt, il suffit de l'envoyer à la victime en utilisant un canal de communication quelconque. La victime se voit alors dans la nécessité de renseigner par exemple, son identifiant et mot de passe pour confirmer son identité.

6.2.7. Quid Pro Quo

Figure 15 Quid Pro Quo

Source : www.mirmay.mplore.com/images?keyword=Quid%20Pro%20Quo

Le quid pro quo, est une expression latine signifiant « quelque chose pour quelque chose ». Les attaques de ce type surgissent quand l'hacker exige des informations confidentielles à la victime en lui offrant une récompense.

6.2.8. Tailgating

Figure 16 Tailgating

Source : www.itgovernance.co.uk/blog/8-kickass-cyber-security-awareness-tips-to-inspire-change-in-your-organisation

Le tailgating est une technique physique d'ingénierie sociale. La personne malveillante suit de près un individu ayant le droit d'accéder à une salle sécurisée par un quelconque moyen de sécurité. C'est par exemple le cas quand l'attaquant demande gentiment à la victime de lui tenir la porte parce qu'elle a oublié son badge d'accès. Le but du tailgating est évidemment de dérober des informations confidentielles ou même des objets ayant de la valeur.

6.2.9. Dumpster Diving

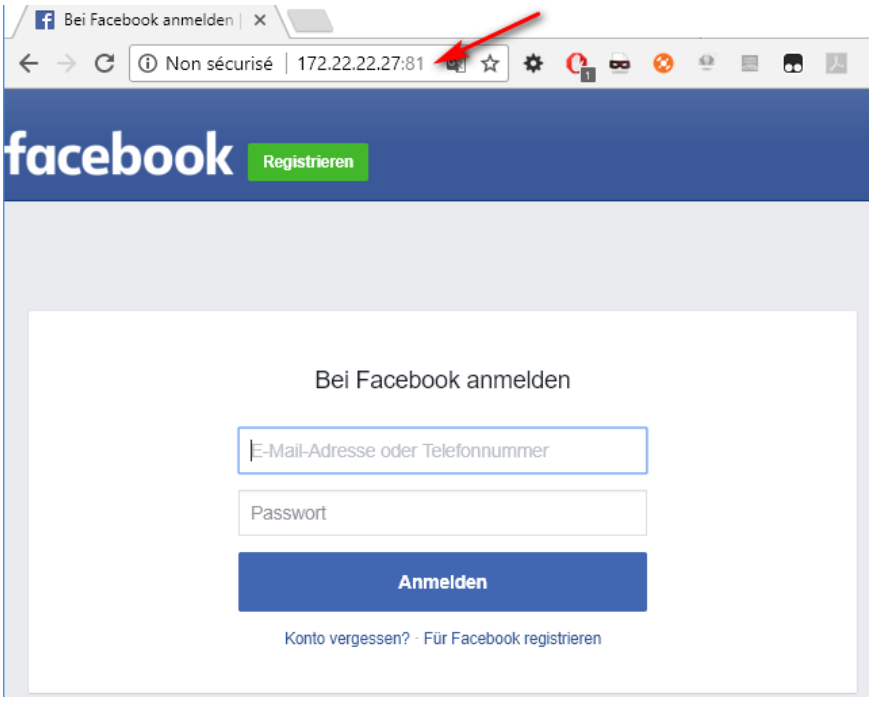
Figure 17 Dumpster diving

Source : www.seef.reputelligence.com

Le dumpster diving est l'action consistant à fouiller les déchets d'autrui avec comme objectif de récupérer quelque chose ayant de la valeur. Cela peut être simplement des feuilles de papier contenant les informations d'identification d'un utilisateur de la société. De nos jours, les entreprises sont de plus en plus impliquées dans l'écologie et essaient de réutiliser par exemple, des feuilles qui ont été imprimées d'un seul côté afin d'imprimer de l'autre. Dans la plupart des cas, les informations inscrites du premier côté de la feuille ne sont pas analysées de manière à définir le niveau de confidentialité du document et de le détruire si nécessaire. Nous donnons ici l'exemple du document papier, mais les dispositifs informatiques tels que les disques durs ou les clés USB contiennent aussi beaucoup d'informations pouvant nuire à la sécurité de l'entreprise. Il faut donc les détruire de façon à ce qu'ils ne soient pas réutilisables. À noter que le formatage d'un de ces appareils informatiques n'est en aucun cas une manière sûre de détruire l'information contenue, car parfois les données formatées peuvent quand même être récupérées.

6.2.10. Harvesting

Figure 18 Harvesting



L'harvesting est une technique, comme son nom l'indique, permettant de récolter quelque chose. Dans le cadre du social engineering, elle est très souvent utilisée afin de récupérer les identifiants d'un utilisateur se connectant à son webmail ou à un réseau social par exemple.

6.3. Outils de social engineering

Ce chapitre est dédié aux outils d'ingénierie sociale existants pour chaque forme décrite dans le chapitre précédent.

6.3.1. Outils de baiting

La plupart des attaques de ce type nécessitent une action physique de la part de l'attaquant. Il n'existe donc pas d'outils à proprement dit, de baiting. Cependant, comme nous l'avons expliqué dans le chapitre précédent, nous pouvons utiliser des outils de phishing cités ci-dessous afin de faire du baiting.

6.3.2. Outils de phishing

Nous listons ci-dessous les principaux outils de phishing. Il en existe d'autres qui ne sont pas listés, car ils possèdent les mêmes fonctionnalités que ceux que nous avons choisis.

6.3.2.1. SecurityIQ PhishSim

Figure 19 SecurityIQ

Source : www.securityiq.infosecinstitute.com

PhishSim de SecurityIQ est un outil de phishing en ligne. Nous l'avons testé et nous pouvons conclure que son interface est très intuitive et que la prise en main se fait très facilement. Comme mentionné, du fait qu'il est en ligne, nous n'avons besoin d'aucune installation sur notre machine. Il propose plus de 100 modèles prédéfinis permettant de gagner un temps non négligeable lors de la conception de l'email. Pour conclure, il propose des rapports complets nous permettant d'analyser les résultats de notre campagne de phishing.

L'entreprise a été fondée en 2015 et est toujours en fonction. Plus d'informations sur SecurityIQ peuvent être trouvées à l'adresse www.securityiq.infosecinstitute.com

6.3.2.2. Gophish

Figure 20 Gophish

Source : www.github.com/gophish/gophish

Gophish est une solution Open-Source de phishing. Elle s'installe autant sur un système d'exploitation Windows que Linux. Cette solution est relativement récente et ses fonctionnalités restent pour le moment encore limitées. Cet outil propose l'importation de carnets d'adresses au format Comma-separated values (CSV). De plus, il nous donne la possibilité de créer une campagne de phishing et de la programmer afin qu'elle s'exécute seule plus tard.

La première version (v0.1) de l'outil a été mise à disposition le 16 janvier 2016. La dernière version (v.0.3.0) de Gophish date du 2 mars 2017.

6.3.2.3. Simple Phishing Toolkit

Simple Phishing Toolkit est une solution de phishing multiplateforme. L'outil est pilotable par une interface Web et possède la particularité de permettre la création de faux liens qui redirigent les victimes vers des pages de sensibilisation aux risques informatiques.

La première version de l'outil est sortie le 15 octobre 2011. Malheureusement, cet outil n'est plus maintenu à jour. En effet, sa dernière version (0.70) a été publiée le 2 novembre 2012.

6.3.3. Outils de spear phishing

Il n'existe pas d'outils de spear phishing à proprement dit. Comme il est uniquement une façon plus précise de faire du phishing, tous les outils mentionnés précédemment sont capables d'établir cette tâche. Cependant, ces outils ne proposent pas des fonctionnalités permettant à l'utilisateur d'envoyer le même email à un carnet d'adresses sans que chaque destinataire sache que d'autres personnes ont reçu cet email.

6.3.4. Outils de vishing

Outre l'outil de vishing compris dans le framework proposé par KnowBe4 dont nous avons parlé précédemment, il existe l'outil ci-dessous.

6.3.4.1. Vishing as a Service

Figure 21 Vishing as a Service

Source : www.social-engineer.com/vishing-service

La société Social-Engineer propose une solution permettant de tester ses utilisateurs par rapport à leurs capacités à identifier une attaque de type vishing. Malheureusement, ils ne fournissent pas beaucoup de détails quant au fonctionnement de leur produit. Ils résument le cycle de vie de leur service à l'aide de l'image ci-dessus.

6.3.5. Outils de smishing

Il n'existe pas d'outils de smishing à proprement dit. En effet, pour faire du phishing nous nécessitions d'un simple serveur Simple Mail Transfer Protocol (SMTP). Pour faire du smishing nous avons donc simplement besoin d'un moyen permettant d'envoyer des SMS. Nous avons testé plusieurs services gratuits tels que sendatext.co, globfone.com, afreesms.com, mfreesms.com, mais le SMS est certainement filtré par l'opérateur téléphonique et n'arrive pas à destination. C'est pour cette raison que nous ne les avons pas analysés plus profondément. Cependant, il existe des applications

installables sur nos téléphones portables permettant de faire de l'envoi en masse de messages texte. Nous ne les avons pas non plus testées, car les messages sont envoyés depuis notre propre numéro et donc facturés. Pour terminer, nous avons trouvé des services tels que textedly.com permettant l'envoi de SMS en masse, dans le but de faire du marketing. Nous ne les avons pas testés non plus, car nous devons renseigner notre numéro de portable afin de confirmer la création du compte et nous ne voulons pas que notre numéro soit inscrit dans des listes qui pourraient un jour servir à faire du smishing.

6.3.6. Outils de pretexting

Le pretexting étant plutôt une façon bien particulière de monter un scénario crédible, il n'existe pas d'outils pour le faire. Néanmoins, une fois le scénario monté, il est possible d'utiliser un outil de phishing, vishing ou smishing dans le but de le distribuer à la victime.

6.3.7. Outils de Quid Pro Quo

Le quid pro quo est, en d'autres mots, une façon de faire du chantage dans le but de récolter des informations confidentielles. Il n'existe donc pas d'outils pour le faire. Toutefois, comme le pretexting, les mêmes outils de distribution de contenu sont utilisables.

6.3.8. Outils de tailgating

Le tailgating étant une forme physique d'ingénierie sociale, il n'existe pas d'outils permettant de le faire.

6.3.9. Outils de dumpster diving

Comme le tailgating, le dumpster diving étant une forme physique de social engineering, il n'existe pas d'outils pour le faire.

6.3.10. Outils d'harvesting

6.3.10.1. Social-Engineer Toolkit (SET)

Figure 22 TrustedSec

Source : www.trustedsec.com

Social-Engineer Toolkit est une boîte à outils développée par TrustedSec permettant de créer différents types d'attaques d'ingénierie sociale. Il propose entre autres, un outil permettant de cloner un site sur son propre serveur Web. Le but est alors de choisir une page sur laquelle il existe un formulaire d'authentification. Une fois la page clonée, l'outil récupère toutes les valeurs introduites par la victime et les retourne au serveur.

Nous ne connaissons pas la date de sortie exacte de la première version. Cependant, nous savons que la version V0.4 a été distribuée en février 2010. La dernière version (V7.7.1) de SET date du 23 juillet 2017.

6.4. Frameworks de développement

Dans ce chapitre, nous listons le framework le plus réputé pour les principaux langages de programmation actuels.

6.4.1. Django

Figure 23 Django

Source : www.djangoproject.com

Django est un framework gratuit et Open-Source pour le développement d'applications Web. Ce framework, écrit en Python, est régulièrement mis à jour et inclut des fonctionnalités telles que le login permettant un gain de temps considérable lors du développement. Django suit le modèle Model-View-Template (MVT) qui est très semblable au Model-View-Controller que nous connaissons tous. De plus, il inclut nativement son propre Object-relational mapping (ORM) compatible avec des bases de données telles que MySQL, PostgreSQL, SQLite, Oracle, SQL Server ou DB2.

La première version de Django (1.0) est sortie le 8 mai 2012. La dernière version (1.11.4) du framework date du 1er août 2017.

6.4.2. Ruby on Rails

Figure 24 Ruby on Rails

Source : www.rubyonrails.org

Ruby on Rails (RoR) offre à peu près les mêmes fonctionnalités que Django. Sa particularité est qu'il est écrit en Ruby. Il a été écrit en suivant les concepts suivants.

- Don't Repeat Yourself (DRY) : ce concept signifie que si plusieurs blocs de code ont le même but, il y a forcément une meilleure façon d'écrire ce code.
- Convention over Configuration : Ruby on Rails part du principe qu'un développeur suit une convention et qu'il s'y tient pendant toute la durée du développement. Dans un premier temps, RoR analyse notre convention, puis s'occupe lui-même de créer le nécessaire afin

de faire gagner du temps au développeur.

- Less Software : Ce concept prône la réduction du nombre de lignes de code. Il est lié au concept précédent dans le sens où RoR privilégie les conventions aux configurations, donc le code en lui-même. Le but ce Less Software est donc de diminuer la complexité du programme ce qui devrait résulter en une diminution de la quantité de bugs.

La première version de RoR (1.0) est sortie le 13 décembre 2005. La dernière version (5.1.2) du framework date du 26 juin 2017.

6.4.3. Express

Figure 25 Express

Source : www.github.com/expressjs/express

Express est un framework Open-Source pour le développement Web utilisant Node.js. Vu qu'il utilise Node.js, il va de soi qu'il est écrit en JavaScript et possède la particularité d'utiliser un langage de frontend en backend. Il est important de relever que les tâches Node.js sont asynchrones et donc non bloquantes. Cette fonctionnalité peut s'avérer très avantageuse pour le développement d'applications nécessitant une quantité importante de connexions à des bases de données telles que des logiciels de messagerie instantanée.

La première version d'Express (0.01) est sortie le 3 janvier 2010. La dernière version (4.15.3) du framework date du 17 mai 2017.

6.4.4. Symfony & Laravel

Figure 26 Symfony & Laravel

Source : www.valuecoders.com/blog/technology-and-apps/symfony-vs-laravel-php-framework-choose

Ce chapitre traite de deux frameworks PHP Hypertext Preprocessor (PHP) qui sont tous deux très bien réputés. En effet, ils proposent une palette de fonctionnalités semblables. Cependant, Laravel est plus rapidement pris en main, car il dispose de la meilleure documentation. Il est donc conseillé au détriment de Symfony dans le cas où le développeur n'a jamais travaillé avec un de ces deux frameworks. D'un autre côté, Symfony est préférable lorsque l'application à développer est complexe et qu'elle se projette sur le long terme. En effet, Symfony a établi les releases plans pour les prochaines années.

7. Choix

À travers ce chapitre, nous comparons les différents outils présentés lors de l'état de l'art précédemment établi. Pour chaque catégorie d'outils, des critères ainsi que les évaluations possibles sont définis afin d'établir une matrice décisionnelle. Chaque critère possède un coefficient de pondération allant de 1 à 2 où 1 reflète une importance moyenne et 2 une importance élevée. Le résultat de cette matrice nous mène au choix final.

7.1. Frameworks de social engineering

7.1.1. Critères d'analyse

- Gratuité = Le framework est gratuit
- Rapports = = Le framework permet d'exporter des rapports
- Formations = Le framework propose des formations basées sur les rapports
- Multi-Tests = Le framework propose plus d'un outil permettant de simuler des attaques

7.1.2. Évaluations possibles

- 0 = Le framework ne répond pas du tout au critère
- 1 = Le framework répond partiellement au critère
- 2 = Le framework répond totalement au critère

7.1.3. Matrice décisionnelle

Tableau 1 Matrice décisionnelle frameworks SE

Critère / Framework	Gratuité	Rapports	Formations	Multi Tests	Total
Pondération	***2***	***1***	***1***	***1***	
Wombat Security	0	2	2	2	**6**
Inspired eLearning	0	2	2	0	**4**
KnowBe4	0	2	2	2	**6**
Lucy	0	2	2	2	**6**

7.1.4. Conclusion

À l'aide de cette matrice, nous pouvons conclure que les différents frameworks analysés sont très semblables. Cependant, aucun d'entre eux ne répond à nos besoins, car le critère de la gratuité est éliminatoire. Nous avons donc pris la décision de créer notre propre framework.

7.2. Formes de social engineering

7.2.1. Critères d'analyse

- Gratuité = Une simulation d'attaque peut se réaliser sans frais
- Outils = Des outils Open-Source existent
- Informatisable = Une simulation d'attaque peut être informatisable

7.2.2. Évaluations possibles

- 0 = La forme de SE ne répond pas du tout au critère
- 1 = La forme de SE répond partiellement au critère
- 2 = La forme de SE répond totalement au critère

7.2.3. Matrice décisionnelle

Tableau 2 Matrice décisionnelle formes de SE

Critère / Forme de SE	Gratuité	Outils	Informatisable	Total
Pondération	***2***	***1***	***2***	
Baiting	1	1	1	**5**
Phishing	2	2	2	**10**
Spear Phishing	2	1	2	**9**
Vishing	0	2	2	**6**
Smishing	1	1	1	**5**
Pretexting	2	1	2	**9**
Quid Pro Quo	2	1	2	**9**
Tailgating	2	0	0	**4**
Dumpster Diving	2	0	0	**4**
Harvesting	2	2	2	**10**

7.2.4. Conclusion

Après avoir établi notre matrice décisionnelle, notre choix se porte sur le phishing et l'harvesting, car ils remplissent totalement nos critères. Cependant, comme expliqué dans le chapitre « Formes de social engineering », nous pouvons utiliser le canal de distribution du phishing, à savoir, l'email afin de faire du baiting, du quid pro quo, du pretexting ainsi que du spear phishing. Nous voulons donc que l'outil de phishing à intégrer à notre framework dispose d'une souplesse notable afin de pouvoir l'utiliser pour exécuter d'autres types d'attaques que le phishing.

7.3. Outils de social engineering

Dans ce chapitre, nous analysons uniquement les outils des formes d'ingénierie sociale retenues au chapitre précédent.

7.3.1. Critères d'analyse

- Gratuité = L'outil est gratuit
- Pilotable = L'outil propose une interface permettant de le piloter sans interaction humaine
- Exportation des résultats = L'outil permet d'exporter les résultats des tests
- Souplesse d'utilisation = L'outil propose une flexibilité d'utilisation lui permettant de réaliser d'autres simulations d'attaques que le phishing pur

7.3.2. Évaluations possibles

- 0 = L'outil ne répond pas du tout au critère
- 1 = L'outil répond partiellement au critère
- 2 = L'outil répond totalement au critère

7.3.3. Matrice décisionnelle

Tableau 3 Matrice décisionnelle outils de SE

Critère / Outil	Gratuité	Pilotable	Exportation	Souplesse	Total
Pondération	***1***	***1***	***1***	***1***	
SecurityIQ PhishSim	1	0	2	0	**3**
Gophish	2	2	1	1	**6**
Simple Phishing Toolkit	2	1	1	1	**5**
Social-Engineer Toolkit	2	2	2		**6**

7.3.4. Conclusion

Nous concluons que l'outil de phishing le plus adapté à notre framework est Gophish. Néanmoins, comme mentionné dans son descriptif, ce projet est relativement récent et nous ne connaissons pas sa pérennité. De plus, nous voulons que l'outil de phishing retenu soit suffisamment flexible afin qu'il puisse servir pour d'autres attaques. C'est pour ces raisons que nous décidons de développer notre propre outil de phishing. Quant à l'outil d'harvesting, il correspond pleinement à nos besoins et nous le retenons afin de l'intégrer dans notre framework.

7.4. Frameworks de développement

7.4.1. Critères d'analyse

- Gratuité = Le framework est gratuit
- Outils = Le framework propose des outils intégrés permettant d'accélérer le développement de notre framework
- Portabilité = Le framework peut s'exécuter sur plusieurs plateformes (Windows, Linux, macOS)
- Stabilité = Le framework a fait ses preuves dans le temps
- Communauté = Une communauté existe autour de ce framework
- Sécurité = Le framework propose des outils de sécurité (Cross-Site Request Forgery (CSRF), Cross site scripting (XSS), SQL injection)

7.4.2. Pondération

- 0 = Le framework ne répond pas du tout au critère
- 1 = Le framework répond partiellement au critère
- 2 = Le framework répond totalement au critère

7.4.3. Matrice décisionnelle

Tableau 4 Matrice décisionnelle frameworks développement

Critère / Framework	Gratuité	Outils	Portabilité	Stabilité	Communauté	Sécurité	Total
Pondération	***1***	***1***	***1***	***1***	***1***	***2***	
Django	2	2	2	2	2	2	**14**
Ruby on Rails	2	2	2	2	2	2	**14**
Express	2	1	2	1	2	1	**10**
Symfony & Laravel	2	2	2	2	2	2	**14**

7.4.4. Conclusion

Nous pouvons conclure que les frameworks basés sur des langages de programmation propres aux backends se trouvent sur un pied d'égalité. Cependant, comme l'outil d'harvesting que nous décidons d'intégrer à notre framework est développé en Python, nous optons pour Django. De plus, comme nous n'avons pas eu l'occasion d'apprendre Python durant notre formation, il nous paraît intéressant d'ajouter ce type de compétence comme nouveau langage de programmation.

8. Application

Dans ce chapitre, nous commençons par lister les outils utilisés pour le développement de notre framework ainsi que toutes les technologies utilisées par celui-ci. La deuxième partie est consacrée aux fonctionnalités implémentées ainsi qu'à leur fonctionnement.

Malgré le fait que Django fasse partie des outils de développements utilisés, il n'est pas cité dans ce chapitre, car toutes les informations le concernant ont été citées dans la section consacrée aux frameworks de développement dans l'état de l'art.

8.1. Outils de développement et technologies

8.1.1. PyCharm

Figure 27 PyCharm

Source : www.jetbrains.com/pycharm

Pycharm est un Integrated Development Environment (IDE) pour le développement en Python créé par l'entreprise JetBrains. C'est un outil robuste proposant des fonctionnalités telles que l'autocomplétion nous permettant de gagner du temps et d'éviter des erreurs de syntaxe. De plus, PyCharm propose une multitude de plugins tels que « GitLab Projects » nous habilitant à utiliser GitLab comme gestionnaire de dépôts. PyCharm est payant dans sa version professionnelle, mais nous possédons une licence en tant qu'étudiants à la HES-SO. Cependant, JetBrains propose une version allégée de PyCharm nommée « Community » qui elle, est gratuite. Plus d'informations sur PyCharm peuvent être trouvées à l'adresse www.jetbrains.com/pycharm.

8.1.2. GitLab

Figure 28 GitLab

Source : www.gitlab.com

GitLab est un gestionnaire de dépôts de logiciels. Il est similaire à GitHub[3] à l'exception que GitLab peut être installé sur un serveur nous appartenant tandis qu'en utilisant GitHub, le code est stocké sur leurs serveurs. Plus d'informations sur GitLab peuvent être trouvées à l'adresse www.gitlab.com.

8.1.3. MariaDB

Figure 29 MariaDB

Source : www.mariadb.org

MariaDB est une base de données relationnelle ressemblant à MySQL[4]. En effet, MariaDB a été créée par les mêmes personnes que celles qui ont créé MySQL suite au rachat de ce dernier par Oracle[5]. Nous choisissons d'utiliser MariaDB au détriment de MySQL principalement, car :

- MariaDB est Open Source
- MariaDB est installé d'office sur Kali Linux
- MariaDB propose des correctifs relatifs à la sécurité plus rapidement que MySQL. En effet, Oracle propose des mises à jour de ses produits tous les trois mois.
- MariaDB est plus performant que MySQL. En effet, les benchmarks prouvent que les bases de données MariaDB sont plus performantes. Cela est aussi valable si nous migrons une base de données MySQL vers MariaDB.

À noter qu'il existe d'autres systèmes de gestion de bases de données compatibles avec Django. Néanmoins, les plus courants sont MySQL et PostgreSQL[6]. Il n'y a pas vraiment de meilleur entre les deux, chacun possède ses avantages et ses inconvénients. Cependant, la tendance actuelle nous guide vers MySQL pour des projets de taille modeste et vers PostgreSQL pour des projets plus ambitieux comprenant des données dépassant la dizaine de téraoctets. De plus, MariaDB est intégré à Kali Linux.

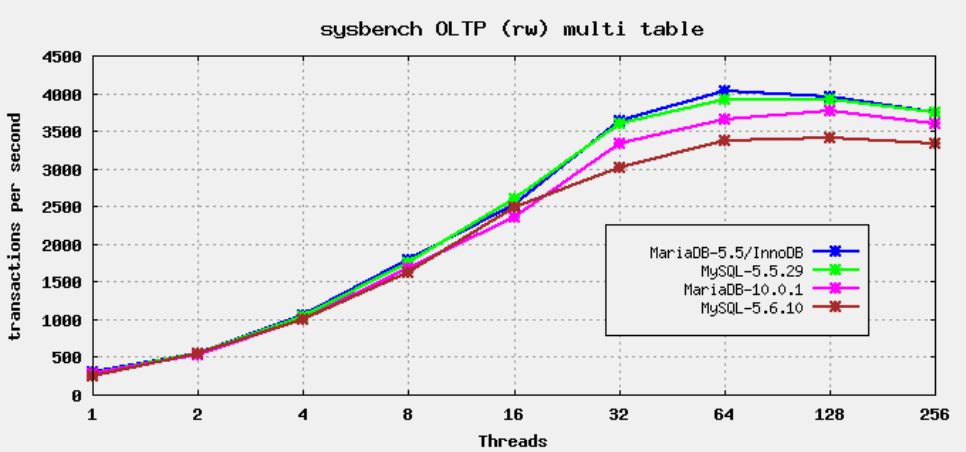
[3] www.github.com

[4] www.mysql.com

[5] www.oracle.com

[6] www.postgresql.org

Figure 30 Benchmark MariaDB / MySQL



Source : https://mariadb.org/wp-content/uploads/2013/02/20130213-sb-rw-tps.png

Plus d'informations sur MariaDB peuvent être trouvées à l'adresse www.mariadb.org.

8.1.4. MySQL Worbench

Figure 31 MySQL Workbench

Source : www.mysql.com/fr/products/workbench

MySQL Workbench est un outil permettant de gérer les bases de données graphiquement. Dans le cadre de notre projet, ce logiciel est utilisé principalement à des fins de tests. En effet, vu que Django gère les données grâce à son ORM, nous n'avons ni une vision de la structure, ni du contenu de la base de données et cela peut être un frein lors du processus de développement. Plus d'informations sur MySQL Workbench peuvent être trouvées à l'adresse www.mysql.com/fr/products/workbench.

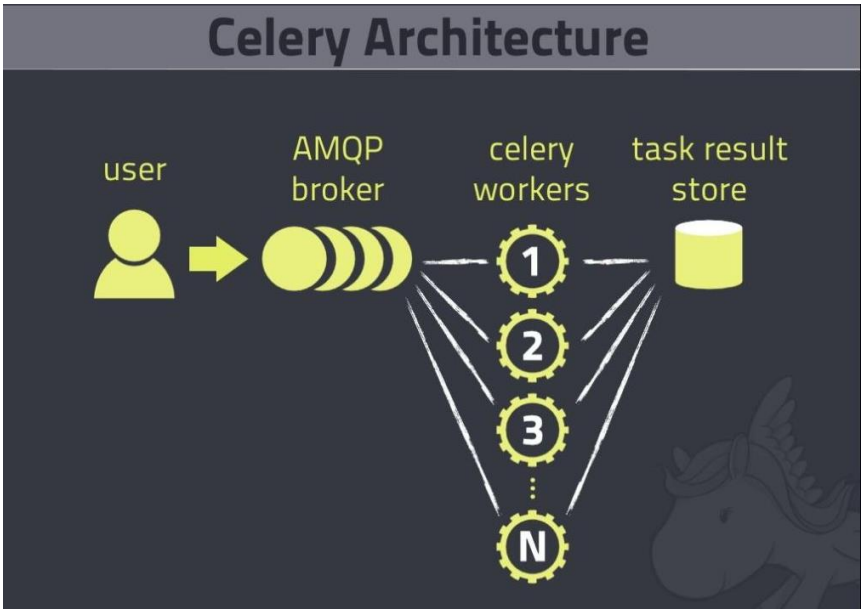
8.1.5. Celery

Figure 32 Celery

Source : www.celeryproject.org

Celery permet de gérer les tâches asynchrones de notre projet. Il permet l'exécution de tâches qu'il reçoit par le biais de messages sans compromettre l'interaction de l'utilisateur. En effet, ces dernières étant asynchrones, l'utilisateur ne doit pas attendre qu'elles se terminent afin de pouvoir continuer à travailler sur l'outil. L'illustration ci-dessous nous offre une vision globale du flux de notre application.

Figure 33 Celery Flux



Source : www.fr.slideshare.net/idangazit/an-introduction-to-celery/15-Celery_Architecture_celery_task_result

- User : Il s'agit de notre application Django
- AMPQ broker : Il s'agit du distributeur de messages.
- Celery workers : Ce sont eux qui sont chargés d'exécuter les tâches qui nous définissons d'asynchrones.
- Task result store : Ce sont des entrepôts de données dans lesquels Celery peut sauvegarder le résultat des tâches qu'il a accomplies.

Plus d'informations sur Celery peuvent être trouvées à l'adresse www.celeryproject.org.

8.1.6. RabbitMQ

Figure 34 RabbitMQ

Source : www.rabbitmq.com

RabbitMQ est un message broker Open Source. Tel que démontré à la figure expliquant le flux de travail de Celery, RabbitMQ est responsable par la distribution des messages provenant de Django aux workers de Celery. RabbitMQ utilise le protocole Advanced Message Queuing Protocol (AMQP) pour la distribution de messages.

Il existe d'autres workers compatibles avec Celery. Cependant, nous choisissons RabbitMQ, car celui-ci est conseillé par Celery. Plus d'informations sur RabbitMQ peuvent être trouvées à l'adresse www.rabbitmq.com.

8.1.7. jQuery

Figure 35 jQuery

Source : www.jquery.com

jQuery est la bibliothèque de fonctions JavaScript Open Source la plus populaire du moment. jQuery permet d'interagir avec les balises HyperText Markup Language (HTML) facilement. En effet, grâce à jQuery, il nous est possible d'effectuer certaines tâches en quelques lignes de code tandis que le même résultat en demande beaucoup plus en JavaScript pur. De plus, jQuery inclus la librairie Asynchronous Javascript and XML (AJAX) nous permettant de créer une interface dynamique. Plus d'informations sur jQuery peuvent être trouvées à l'adresse www.jquery.com.

8.1.8. Bootstrap

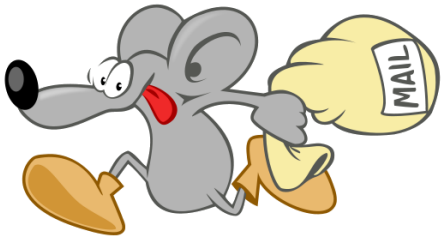
Figure 36 Bootstrap

Source : www.getbootstrap.com

Bootstrap est un framework front-end Open Source permettant de gagner un temps considérable lors du développement. Par défaut, les vues écrites à l'aide de Bootstrap sont « Responsive Design ». Dans le cadre de ce projet, toutes nos vues sont écrites à l'aide de ce framework.

8.1.9. Postfix

Figure 37 Postfix



Source : www.postfix.org

Postfix est un serveur Simple Mail Transfer Protocol (SMTP) Open Source. Il possède comme atout d'être simple à configurer et facile à maintenir. Dans le cadre de ce projet, il est utilisé afin de faire suivre l'email conçu par l'utilisateur au prochain Mail Transfer Agent (MTA). Plus d'informations sur postfix peuvent être trouvées à l'adresse www.postfix.org.

8.1.10. Apache

Figure 38 Apache

Source : www.apache.org

Apache est actuellement le serveur Web le plus populaire. Il est utilisé dans notre projet à plusieurs escients. Premièrement, nous l'utilisons pour héberger notre outil. Deuxièmement, il est utilisé par SET afin d'héberger les sites clonés dans le but de récolter des identifiants.

Par défaut, Apache n'est pas capable d'héberger des applications Web développées en Python telle que la nôtre. Nous devons donc installer un module suivant la convention Web Server Gateway Interface (WSGI) afin de permettre à Apache de communiquer avec notre application Django.

Figure 39 WSGI

Source : www.sametmax.com/wp-content/uploads/2013/07/wsgi.png

Plus d'informations sur Apache peuvent être trouvées à l'adresse www.apache.org.

8.1.11. CKEditor

Figure 40 CKEditor

CKEditor est un éditeur What You See Is What You Get (WYSIWYG) Open Source écrit en JavaScript. Il permet de créer le contenu du page à l'aide d'une palette graphique. Ce contenu est ensuite traduit en code HTML. Cet éditeur est intégré nativement à des Content Management System (CMS) bien réputés tels que Drupal[7] ou Joomla![8]. Dans le cadre de notre projet, il est utilisé afin d'aider l'utilisateur à composer un email.

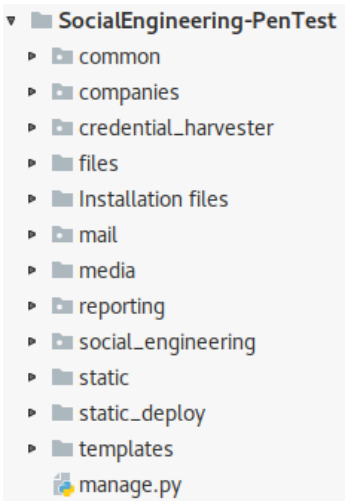
8.2. Architecture Django

Dans ce chapitre nous décrivons l'architecture de Django.

8.2.1. Structure des répertoires

Ci-dessous, nous trouvons l'architecture actuelle de notre projet.

Figure 41 Structure répertoires Django



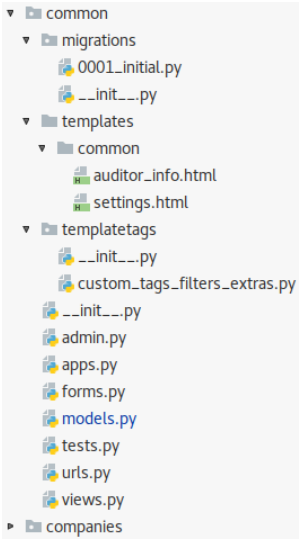
8.2.1.1. Common

Common est une application contendruant les paramètres globaux de notre framework. Son architecture est la suivante.

[7] https://www.drupal.org

[8] https://www.joomla.fr

Figure 42 Architecture common



Répertoire migrations

Ce répertoire contient les fichiers de migration des modèles. Ils servent à faciliter la migration de la base de données lors de mises à jour dans le but de ne pas écraser les données existantes. Tous les fichiers compris dans ce répertoire sont générés automatiquement par la commande ci-dessous. À noter que cette commande doit être exécutée dans le répertoire du projet.

```
python manage.py makemigrations
```

Fichier 0001_initial.py

Ce fichier est généré lors de la première migration. Son contenu est le suivant.

Figure 43 0001_initial.py

```python
class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Auditor',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('company_logo', models.ImageField(upload_to=b'')),
                ('company_name', models.CharField(max_length=200)),
                ('company_address', models.CharField(max_length=100)),
                ('company_zip', models.CharField(max_length=20)),
                ('company_city', models.CharField(max_length=50)),
                ('company_website', models.CharField(blank=True, max_length=2000, null=True)),
                ('company_phone', models.CharField(blank=True, max_length=50, null=True)),
                ('firstname', models.CharField(max_length=100)),
                ('lastname', models.CharField(max_length=100)),
                ('phone', models.CharField(blank=True, max_length=50, null=True)),
            ],
        ),
        migrations.CreateModel(
            name='Settings',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('sequence_credential_harvester', models.CharField(max_length=200)),
            ],
        ),
    ]
```

Cette classe contient la structure des tables à créer dans la base de données. Les attributs des champs y sont aussi définis. Elle reflète le contenu de notre modèle de données décrit dans le fichier « models.py » illustré à l'aide de la figure suivante. À noter que la classe « Auditor » possède une méthode « save » permettant de redimensionner l'image envoyée par l'utilisateur avant de la sauvegarder dans le système de fichiers du serveur. C'est une méthode héritée de la classe parente que nous personnalisons afin qu'elle réponde à nos besoins.

Figure 44 models.py

```
class Settings(models.Model):
    sequence_credential_harvester = models.CharField(max_length=200)

    def __str__(self):
        return self.sequence_credential_harvester


class Auditor(models.Model):
    company_logo = models.ImageField()
    company_name = models.CharField(max_length=200)
    company_address = models.CharField(max_length=100)
    company_zip = models.CharField(max_length=20)
    company_city = models.CharField(max_length=50)
    company_website = models.CharField(max_length=2000, null=True, blank=True)
    company_phone = models.CharField(max_length=50, null=True, blank=True)
    firstname = models.CharField(max_length=100)
    lastname = models.CharField(max_length=100)
    phone = models.CharField(max_length=50, null=True, blank=True)

    def save(self):
        super(Auditor, self).save()
        im = Image.open(self.company_logo)
        (width, height) = im.size

        factor = float(width) / float(height)

        if width > height:
            size = 310, int(310 / factor)
        else:
            size = int(310 / factor), 310

        im = im.resize(size, Image.ANTIALIAS)
        im.save(self.company_logo.path)

    def __str__(self):
        return self.company_name
```

Afin que l'ORM de Django crée la base de données d'après les modèles définis, nous devons exécuter la commande ci-dessous.

```
python manage.py migrate
```

La table générée par ce modèle se présente comme suit.

Figure 45 Auditor table inspector

Column	Type	Default Value	Nullable
id	int(11)		NO
company_logo	varchar(100)		NO
company_name	varchar(200)		NO
company_address	varchar(100)		NO
company_zip	varchar(20)		NO
company_city	varchar(50)		NO
company_website	varchar(2000)		YES
company_phone	varchar(50)		YES
firstname	varchar(100)		NO
lastname	varchar(100)		NO
phone	varchar(50)		YES

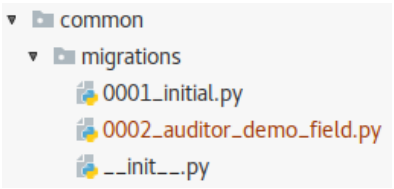
À titre d'exemple, nous ajoutons à notre modèle de données un champ nommé « demo_field » de type chaîne de caractères avec comme taille maximum 100 caractères. Après cette modification, notre classe « Auditor » se présente ainsi.

Figure 46 Class Auditor

```
class Auditor(models.Model):
    company_logo = models.ImageField()
    company_name = models.CharField(max_length=200)
    company_address = models.CharField(max_length=100)
    company_zip = models.CharField(max_length=20)
    company_city = models.CharField(max_length=50)
    company_website = models.CharField(max_length=2000, null=True, blank=True)
    company_phone = models.CharField(max_length=50, null=True, blank=True)
    firstname = models.CharField(max_length=100)
    lastname = models.CharField(max_length=100)
    phone = models.CharField(max_length=50, null=True, blank=True)
    demo_field = models.CharField(max_length=100)
```

Afin que ces modifications soient appliquées à la base de données, nous exécutons la commande de migration décrite précédemment. Le script lancé par cette commande analyse les différences entre le modèle initial et son état actuel. Dans notre exemple, nous avons ajouté un champ qui ne peut être nul. Nous sommes donc interrogés par le script sur la valeur par défaut de ce champ. Comme nous pouvons le constater à l'aide de l'illustration suivante, de l'exécution de ce script résulte un fichier Python placé dans le répertoire « migrations ».

Figure 47 Répertoire migrations après migration



Le contenu de ce fichier permet d'indiquer à l'ORM de Django quelles sont les modifications à apporter à la base de données. Ci-dessous, nous trouvons l'illustration du contenu de ce fichier ainsi que quelques explications permettant de comprendre son fonctionnement.

Figure 48 0002_auditor_demo_field.py

```
class Migration(migrations.Migration):

    dependencies = [
        ('common', '0001_initial'),
    ]

    operations = [
        migrations.AddField(
            model_name='auditor',
            name='demo_field',
            field=models.CharField(default='Content demo field', max_length=100),
            preserve_default=False,
        ),
    ]
```


À présent, nous pouvons exécuter la commande permettant de créer ou de modifier la base de données. Tel qu'illustré à l'aide de l'image ci-dessous, le champ a bien été ajouté.

Figure 49 Auditor table inspector avec demo_field

Column	Type	Default Value	Nullable
id	int(11)		NO
company_logo	varchar(100)		NO
company_name	varchar(200)		NO
company_address	varchar(100)		NO
company_zip	varchar(20)		NO
company_city	varchar(50)		NO
company_website	varchar(2000)		YES
company_phone	varchar(50)		YES
firstname	varchar(100)		NO
lastname	varchar(100)		NO
phone	varchar(50)		YES
demo_field	varchar(100)		NO

À noter que le champ nouvellement ajouté possède « Content demo field » comme valeur pour toutes les entrées déjà existantes de la table modifiée.

Répertoire templates

Ce répertoire est composé de tous les templates de l'application. En faisant l'analogie avec le modèle MVC, les templates correspondent aux vues. Ce sont des fichiers HTML comprenant en plus, des balises et filtres de gabarit intégrés propres à Django. L'illustration suivante affiche le contenu du « fichier auditor_info.html » ainsi que quelques explications concernant les balises propres à Django.

Figure 50 auditor_info.html

```
{% extends "base.html" %}
{% block content %}
    <div class="col-md-6">
        {% if message %}
            <div class="alert alert-danger" style="...">
                <strong>{{ message }}</strong>
            </div>
        {% endif %}
        <div class="panel panel-default">
            <div class="panel-heading">Auditor personnal information</div>
            <div class="panel-body">
                <form method="post" enctype="multipart/form-data">
                    {% csrf_token %}
                    <div class="form-group">
                        <label>
                            {{ form.company_logo.label }}
                        </label><br/>
                        {% if auditor.company_logo %}
                            <img src="{{ auditor.company_logo.url }}" style="max-width: 150px;max-height: 150px;"/>
                        {% endif %}
                        <input type="file" name="company_logo" id="id_company_logo" style="...">
                    </div>
                    <div class="has-errors text-danger small">
                        {{ form.company_logo.errors }}
                    </div>
                    {% for field in form.visible_fields %}
                        {% if field.name not in 'company_logo' %}
                            <div class="form-group">
                                <label>
                                    {{ field.label }}
                                </label>
                                {{ field }}
                            </div>
                            <div class="has-errors text-danger small">
                                {{ field.errors }}
                            </div>
                        {% endif %}
                    {% endfor %}
                    {% for field in form.hidden_fields %}
                        {% if field.name not in 'company_logo' %}
                            <div style="...">{{ field }}</div>
                        {% endif %}
                    {% endfor %}
                    {% for field in form.hidden_fields %}
                        <div style="...">{{ field }}</div>
                    {% endfor %}
                    <input type="submit" value="Submit" class="btn btn-default"/>
                </form>
            </div>
        </div>
    </div>
{% endblock %}
```

Etend le fichier base.html contenant l'entête de la page

Protection contre le CSRF

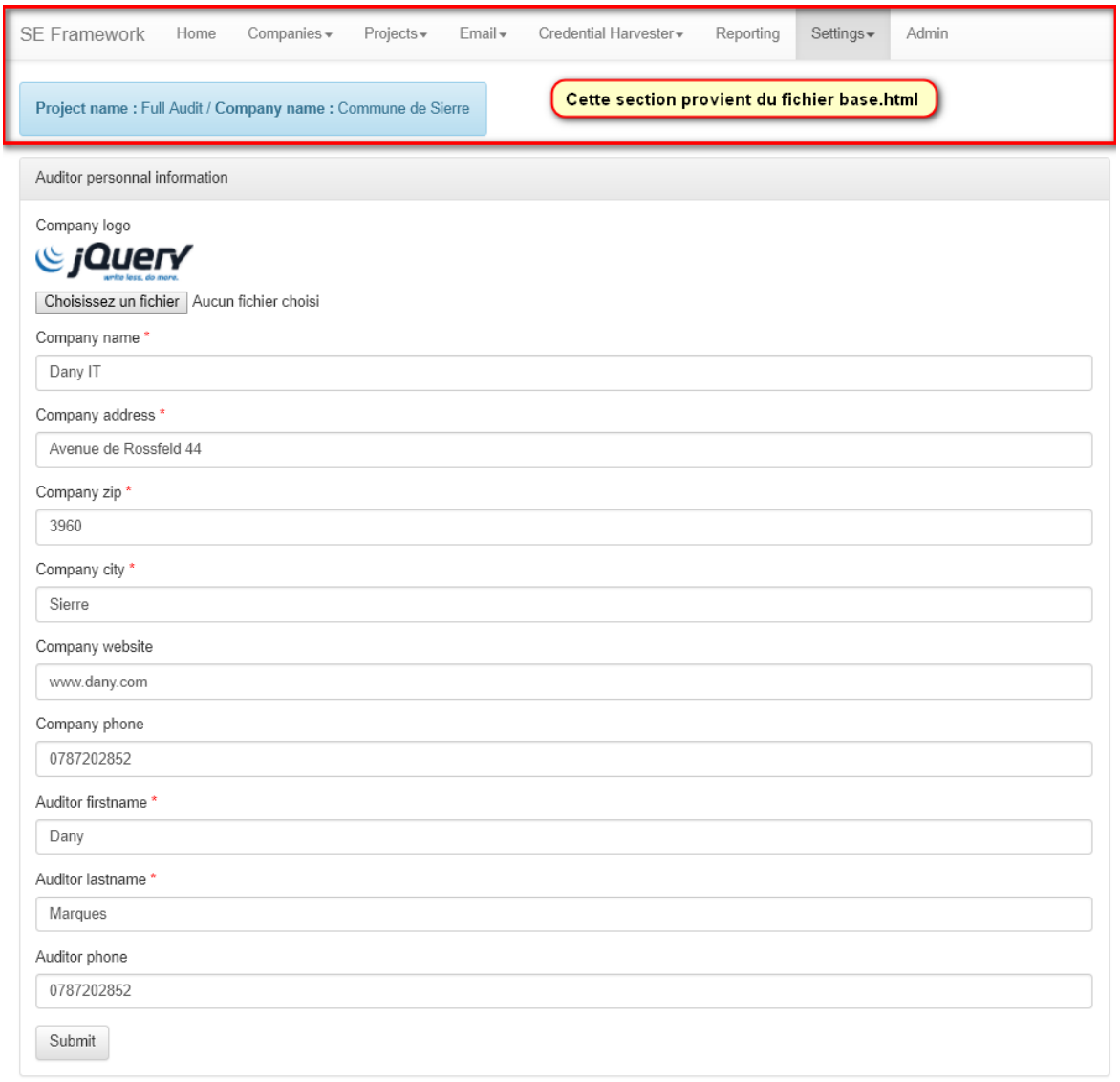
Test conditionnel

Boucle for parcourant tous les champs du formulaire

Affichage du contenu d'une variable

Ce template est rendu par notre navigateur Web de la manière suivante.

Figure 51 Page auditor info



Lors de l'inspection du code source de la page, nous apercevons une balise cachée contenant comme valeur le token généré par le tag {% csrf_token %}.

Figure 52 CSRF token

```
<input type='hidden' name='csrfmiddlewaretoken' value='ZItc4qEXEzD5Es8PxumQwjGbekDWkwJmzaXPg36w9IjpQ3WLoS01TobuJIrgtOq6' />
```

Plus d'informations sur les balises et filtres de gabarit intégrés à Django peuvent être trouvées à l'adresse https://docs.djangoproject.com/fr/1.11/ref/templates/builtins. Concernant la protection contre le CSRF, la documentation officielle de Django disponible à l'adresse https://docs.djangoproject.com/fr/1.11/ref/csrf/ explique son fonctionnement.

Répertoire templatestags

Ce répertoire est composé de fichiers contenant la définition de nos propres tags et filtres. En effet, Django offre la possibilité d'étendre ses fonctionnalités afin que le code garde une certaine homogénéité. Tel que mentionné précédemment, nos vues sont construites à l'aide de Boostrap. Nos balises HTML possèdent donc des classes propres à ce framework. Cependant, comme Django génère les balises HTML automatiquement en fonction du modèle de données, nous devons ajouter manuellement les classes propres à Bootstrap à ces balises. Pour ce faire, nous créons une méthode dans le fichier « custom_tags_filters_extras.py ».

Figure 53 custom_tags_filters_extras.py

```
from django import template
register = template.Library()

@register.filter(name='addcss')
def addcss(field, css):
   return field.as_widget(attrs={"class":css})
```

Ce filtre peut être appelé depuis tous les templates de notre application. Ci-dessous, nous illustrons un exemple de la manière d'utiliser le filtre créé.

Figure 54 Filtre addcss

```
<div class="form-group">
    {{ form.old_password.errors }}
    <label>
        {{ form.old_password.label }}
    </label>
    {{ form.old_password|addcss:"form-control" }}
</div>
```

En l'occurrence, ce filtre nous permet d'ajouter la classe CSS « form-control » à notre champ.

Fichier admin.py

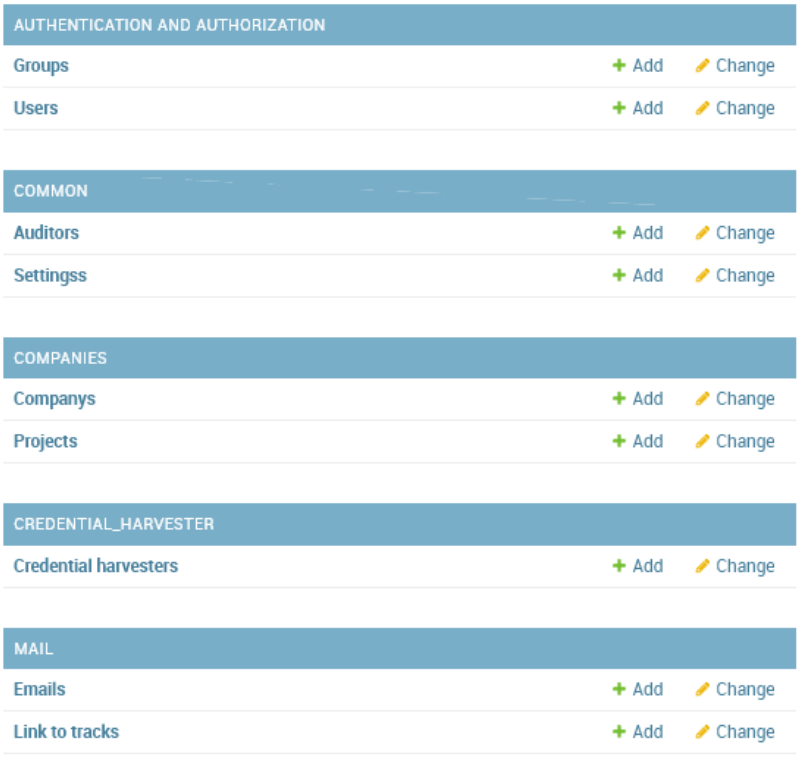
Django propose un système d'administration des modèles générés automatiquement. Il suffit de renseigner dans le fichier « admin.py » de chaque application le nom des classes qu'il doit gérer.

Figure 55 admin.py

```
admin.site.register(Settings)
admin.site.register(Auditor)
```

L'interface d'administration est atteignable à l'adresse http://IP_DE_NOTRE_SERVEUR/admin. L'administration de Django nous permet de faire des opérations CREATE, READ, UPDATE, DELETE (CRUD) sur chaque modèle enregistré dans ce fichier.

Figure 56 Admin Django



Fichier apps.py

Tel que nous l'avons défini au début de ce chapitre, « common » est une application appartenant au projet « SocialEngineering-PenTest ». Dans la théorie, chaque application contenue dans un projet est indépendante et peut être exportée afin d'être réutilisée dans un autre contexte. Ce fichier sert donc à définir des paramètres propres à l'application dans le but de ne pas tous les définir dans le fichier de configuration du projet.

Fichier forms.py

Ce fichier contient tous les formulaires de l'application. Cependant, Django offre la possibilité de créer les formulaires à partir de modèles. Nous utilisons pour cela, la sous-classe « Meta ». Nous pouvons définir les attributs de cette classe afin que notre formulaire réponde pleinement à nos attentes. Il est possible d'exclure des champs de manière à ce qu'ils n'apparaissent pas dans les vues, comme il est souvent le cas des champs dédiés aux relations entre les modèles. Plus d'informations concernant les formulaires peuvent être trouvées à l'adresse https://docs.djangoproject.com/fr/1.11/topics/forms/

Figure 57 forms.py

```
class SettingsForm(forms.ModelForm):
    class Meta:
        model = Settings
        fields = '__all__'
        labels = {
            'sequence_credential_harvester': 'Sequence of commands used to start the credential harvester tool'
        }
        widgets = {
            'sequence_credential_harvester': TextInput(attrs={'class': 'form-control'}),
        }

class AuditorForm(forms.ModelForm):
    class Meta:
        model = Auditor
        fields = '__all__'
        widgets = {
            'company_name': TextInput(attrs={'class': 'form-control'}),
            'company_address': TextInput(attrs={'class': 'form-control'}),
            'company_zip': TextInput(attrs={'class': 'form-control'}),
            'company_city': TextInput(attrs={'class': 'form-control'}),
            'company_website': TextInput(attrs={'class': 'form-control'}),
            'company_phone': TextInput(attrs={'class': 'form-control'}),
            'firstname': TextInput(attrs={'class': 'form-control'}),
            'lastname': TextInput(attrs={'class': 'form-control'}),
            'phone': TextInput(attrs={'class': 'form-control'}),
        }
        labels = {
            'firstname': 'Auditor firstname',
            'lastname': 'Auditor lastname',
            'phone': 'Auditor phone',
        }
```


Fichier models.py

Ce fichier contient les modèles de l'application. Nous ne l'illustrons pas dans ce chapitre, car nous l'avons déjà fait lors du chapitre dédié au fichier « migrations.py ».

Fichier tests.py

Ce fichier contient tous les tests unitaires de notre application. Le temps à disposition pour ce projet étant limité, nous décidons de ne pas faire de tests.

Fichier urls.py

Ce fichier contient le routage des Uniform Resource Locator (URL) de notre application. Afin de mieux illustrer son utilisation, le fichier présenté ci-dessous appartient à l'application « companies » et non pas à « common ». Il est possible d'ajouter des expressions régulières dans l'URL afin de pouvoir passer des paramètres à la méthode. Prenons comme exemple la dernière règle définie dans l'illustration suivante : son URL d'accès est /project/*paramètres_numérique*/gen_report, les requêtes arrivant sur cette dernière appellent la méthode « gen_report » décrite dans le fichier « views.py » et pour finir, la requête se nomme « gen_project ».

Figure 58 urls.py

```
urlpatterns = [
    url(r'^$', views.companies, name="companies"),
    url(r'^edit/(?P<id>\d+)/$', views.company_update, name="edit_company"),
    url(r'^new/$', views.company_create, name="add_company"),
    url(r'^count/$', views.companies_count, name="companies_count"),
    url(r'^(?P<id>\d+)/projects/$', views.projects_by_company, name="projects_by_company"),
    url(r'^(?P<id>\d+)/projects/new/$', views.project_create, name="add_project"),
    url(r'^(?P<id_project>\d+)/change_selected_project/$', views.change_selected_project, name="change_selected_project"),
    url(r'^projects/$', views.projects, name="projects"),
    url(r'^projects/edit/(?P<id>\d+)/$', views.project_update, name="edit_project"),
    url(r'^project/(?P<id_project>\d+)/report/$', views.report_project, name="report_project"),
    url(r'^project/(?P<id_project>\d+)/gen_report/$', views.gen_report, name="gen_report"),
]
```

Méthode à appeler

Nom de l'URL

Pattern de l'URL

Le nom de la règle sert à créer des liens vers l'URL dans les vues tel qu'illustré ci-dessous.

Figure 59 Exemple URL

```
<ul>
    {% for report in project.report_set.all %}
        <li><a target="_blank" href="{% url "open_gen_report" id_report=report.id %}">{{ report.date }}</a></li>
    {% endfor %}
</ul>
```

Dans cet exemple, nous créons une liste à puces en bouclant sur tous les rapports contenus dans un projet. À noter que nous définissions le paramètre passé à la méthode à chaque itération de la boucle.

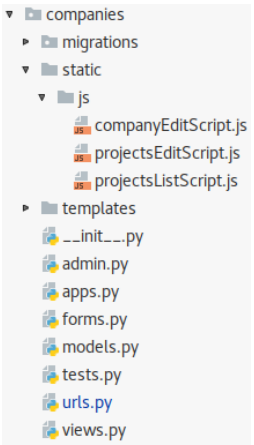
Fichier views.py

Ce fichier contient toutes les méthodes métier de l'application. Pour faire l'analogie avec le modèle MVC, ce fichier correspond au contrôleur.

8.2.1.2. Companies

« Companies » est l'application permettant de gérer les entreprises auditées ainsi que les projets. Ils se trouvent dans la même application, car ils sont étroitement liés. Sa structure est semblable à l'application décrite précédemment à l'exception qu'elle possède un répertoire « static » comprenant un sous-répertoire « js » dans lequel se trouvent les fichiers JavaScript propres à cette application.

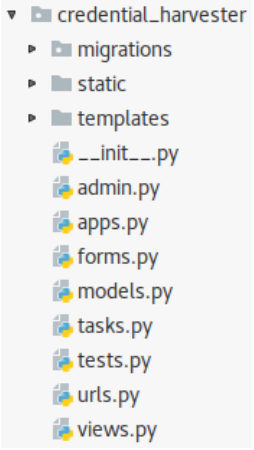
Figure 60 Companies



8.2.1.3. Credential_harvesting

Cette application permet de gérer les attaques permettant de récolter des identifiants. Sa structure est similaire aux applications précédentes à l'exception qu'elle possède un fichier « tasks.py » comprenant des méthodes asynchrones dont nous parlons plus tard dans ce rapport.

Figure 61 credential_harvester



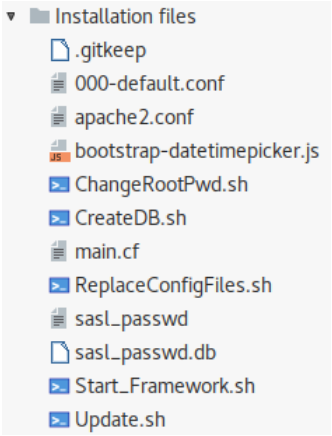
8.2.1.4. Files

Ce répertoire héberge les rapports générés par notre framework afin que l'utilisateur puisse les retrouver à tout moment.

8.2.1.5. Installation files

Ce répertoire contient tous les fichiers nécessaires au fonctionnement de notre framework.

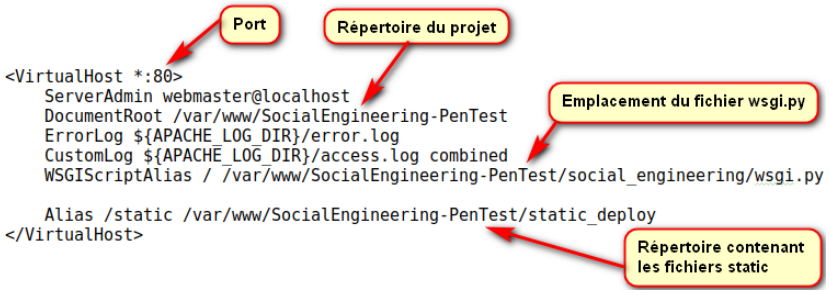
Figure 62 Installation files



- 000-default.conf

Fichier contenant la définition des VirtualHost d'Apache. Il est configuré afin de permettre à Apache de s'interfacer avec notre application Django à l'aide de WSGI.

Figure 63 000_default.conf



Fichier apache2.conf

Ce fichier contient la configuration basique d'Apache. Néanmoins, afin de permettre l'interfaçage entre Apache et Python, nous devons renseigner le répertoire du projet à l'aide de l'instruction ci-dessous.

```
WSGIPythonPath /var/www/SocialEngineering-PenTest
```

Fichier bootstrap-datetimepicker.js

Ce fichier JavaScript contient la version actuelle du module « bootstrap-datetimepicker » permettant de remplacer l'ancienne version incluse dans le widget « DateTime ».

- GitHub de django-datetime-widget
  - https://github.com/asaglimbeni/django-datetime-widget
- GitHub de bootstrap-datetimepicker
  - https://github.com/smalot/bootstrap-datetimepicker

Fichier ChangeRootPwd.sh

À l'installation de Kali Linux, une instance de MariaDB est déjà installée. Cependant, le compte « root » créé ne possède pas de mot de passe. Ce script permet donc de définir un mot de passe à l'utilisateur « root » de MariaDB.

Fichier CreateDB.sh

Ce script permet premièrement de créer un nouvel utilisateur dans MariaDB ainsi que de créer une nouvelle base de données. Deuxièmement, il modifie le fichier de configuration de notre projet Django afin de le lier à la base créée. Troisièmement, il crée toutes les tables de la base de données. Pour finir, il ajoute une entrée dans la table « common_settings » définissant la séquence de paramètres à renseigner à SEToolkit afin de lancer l'outil d'harvesting. Les données telles que le nom d'utilisateur à créer, son mot de passe ainsi que le nom de la base sont demandées à l'utilisateur lors de l'exécution du script. À noter que si nous renseignons un nom de base de données déjà existant, celle-ci est supprimée avant d'être créée à nouveau. Son contenu est donc perdu. Le même phénomène se produit avec l'utilisateur.

Fichier main.cf

Ce fichier contient la configuration du notre serveur SMTP, à savoir, postfix.

Fichier ReplaceConfigFiles.sh

Ce script permet de copier les différents fichiers de configuration contenus dans le répertoire actuel à leurs emplacements respectifs. De plus, il modifie les droits d'accès à certains répertoires afin de permettre à notre framework d'y écrire.

Fichier sasl_passwd

Ce fichier contient les informations d'authentification aux serveurs SMTP.

Fichier sasl_passwd.db

Comme un fichier texte plat ne peut être lu autant rapidement qu'une base de données, postfix exige que les informations d'authentification aux serveurs SMTP soient dans une base de données. Cette base de données est créée automatiquement avec la commande ci-dessous en se basant sur le contenu du fichier précédent.

```
postmap hash:/etc/postfix/sasl_passwd
```

Fichier Start_Framework.sh

Ce script permet de lancer tous les services nécessaires au bon fonctionnement de notre framework. De plus, il lance une instance d'un « worker » Celery afin de permettre l'exécution des tâches asynchrones définies dans le projet.

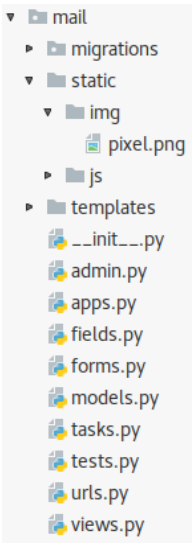
Fichier Update.sh

Ce script permet de récupérer les éventuelles mises à jour du projet dans le dépôt GitLab prévu à cet effet. Si la base de données doit être modifiée suite à ces mises à jour, le script le fait automatiquement.

8.2.1.6. Mail

Cette application permet de gérer nos attaques d'hameçonnage. Le contenu de celle-ci est semblable aux autres à l'exception qu'elle possède un répertoire « img » supplémentaire. Ce dernier contient une image d'un pixel transparent. Cette image est injectée dans l'email envoyé lorsque l'option « tracking email » est activée.

Figure 64 Mail



À l'intérieur de cette application, nous trouvons aussi un fichier « fields.py » contenant une classe permettant de définir un type de champ que nous nommons « MultiEmailField ». Dans cette classe, nous trouvons deux méthodes héritées de la classe parente nous permettant de définir nos propres règles de validation ainsi que nos propres messages en cas de non-validation du champ.

Figure 65 fields.py

```
class MultiEmailField(forms.Field):
    def to_python(self, value):
        # Return an empty String if no input was given.
        if not value:
            return ''
        return value

    # Check if value consists only of valid emails
    def validate(self, value):
        super(MultiEmailField, self).validate(value)

        # Only validate the field if it's not empty
        if value != '':
            value = "".join(value.split()) # Remove all spaces. More efficient than string replace method
            value = value.split(',') # Create an array with all the emails

            for email in value:
                try:
                    validate_email(email)
                except ValidationError:
                    raise forms.ValidationError("Only email addresses separated by commas are accepted.")
```

8.2.1.7. Media

Ce répertoire contient le logo de la HES-SO Valais affiché sur la page d'accueil de notre framework. De plus, ce dernier hébergera les images envoyées par l'utilisateur. À savoir, le logo de l'entreprise auditée ainsi que le logo de l'entreprise menant l'audit.

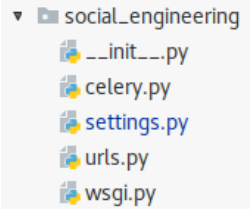
8.2.1.8. Reporting

Cette application gère la génération, le stockage ainsi que l'affichage des rapports.

8.2.1.9. Social_engineering

Ce répertoire contient les fichiers de configuration globaux.

Figure 66 social_engineering



Fichier celery.py

Ce fichier contient les réglages propres à notre gestionnaire de tâches asynchrones.

Figure 67 celery.py

```
# set the default Django settings module for the 'celery' program.
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'social_engineering.settings')

app = Celery('tasks', backend='rpc://', broker='pyamqp://')

# Using a string here means the worker don't have to serialize
# the configuration object to child processes.
# - namespace='CELERY' means all celery-related configura
#   should have a `CELERY_` prefix.
app.config_from_object('django.conf:settings', namespace='CELERY')

# Load task modules from all registered Django app configs.
app.autodiscover_tasks()


@app.task(bind=True)
def debug_task(self):
    print('Request: {0!r}'.format(self.request))
```

Définition du broker utilisé (RabbitMQ)

Méthode affichant les erreurs dans la console

Fichier settings.py

Ce fichier contient tous les paramètres de l'application. Nous listons les plus importants ci-dessous.

- DEBUG = True -> Ce paramètre permet d'afficher des détails lors d'une erreur. Il doit évidemment être défini à False lors de la mise en production de l'outil afin de ne donner aucune information au visiteur.
- ALLOWED_HOSTS -> Permet de définir depuis quelles adresses le serveur peut être atteint.
- INSTALLED_APPS -> Contient la liste de toutes les applications installées dans notre projet.

Figure 68 settings.py

```
DEBUG = True

ALLOWED_HOSTS = ['*']


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'mail',
    'companies',
    'datetimewidget',
    'common',
    'credential_harvester',
    'reporting',
]
```

- TEMPLATES -> Ce paramètre permet de spécifier l'emplacement des templates. À noter que le répertoire files y est inscrit. En effet, Django ne permet pas d'afficher une simple page HTML. Toute page devant être affichée par le projet doit donc être définie en tant que template.

Figure 69 Templates settings.py

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, "templates"), os.path.join(BASE_DIR, "files")],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

- DATABASES -> Ce paramètre permet de spécifier les paramètres d'accès à la base de données.

Figure 70 Databases settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'DBNAME',
        'USER': 'DBUSER',
        'PASSWORD': 'DBPASSWORD',
        'HOST': '127.0.0.1',
        'PORT': '',
        'OPTIONS': {
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
        },
    }
}
```

- STATIC_URL & STATICFILES_DIRS -> Ces paramètres permettent de spécifier l'emplacement des fichiers « statics » tels que les fichiers écris en JavaScript et les feuilles de styles CSS.
- STATIC_ROOT -> Ce paramètre permet de définir l'emplacement des fichiers une fois le serveur en production. Vu que les fichiers « statics » sont répertoriés dans chaque application, nous devons exécuter la commande ci-dessous afin que Django les rassemble tous dans un seul et unique répertoire.

```
python manage.py collectstatic
```

- LOGIN_REDIRECT_URL -> Ce paramètre indique à quelle adresse l'utilisateur doit être redirigé lorsqu'il s'est authentifié.
- LOGOUT_REDIRECT_URL -> Ce paramètre indique à quelle adresse l'utilisateur doit être redirigé lorsqu'il s'est déconnecté.
- MEDIA_ROOT & MEDIA_URL -> Ces paramètres permettent d'indiquer l'emplacement des fichiers « média » tels que les images.

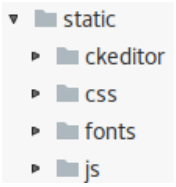
Figure 71 Static files settings.py

```
STATIC_URL = '/static/'

STATIC_ROOT = '/var/www/SocialEngineering-PenTest/static_deploy/'

STATICFILES_DIRS = (
    os.path.join(BASE_DIR, "static"),
)

# Redirect to home URL after login (Default redirects to /accounts/profile/)
LOGIN_REDIRECT_URL = '/'

# Redirect to home URL after login (Default redirects to /accounts/logout/)
LOGOUT_REDIRECT_URL = '/'

# Media files
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
```

8.2.1.10. Static

Ce répertoire contient tous les fichiers « statics » globaux tels que les fichiers sources de CKEditor ou la police de caractères « glyphicons ».

Figure 72 Static files



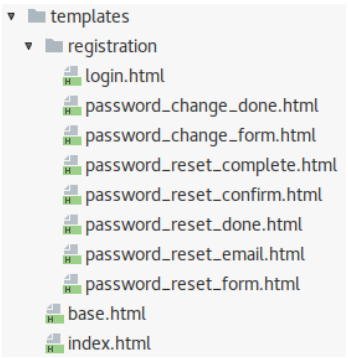
8.2.1.11. Static_deploy

Ce répertoire contient tous les fichiers « statics » de notre framework une fois en production.

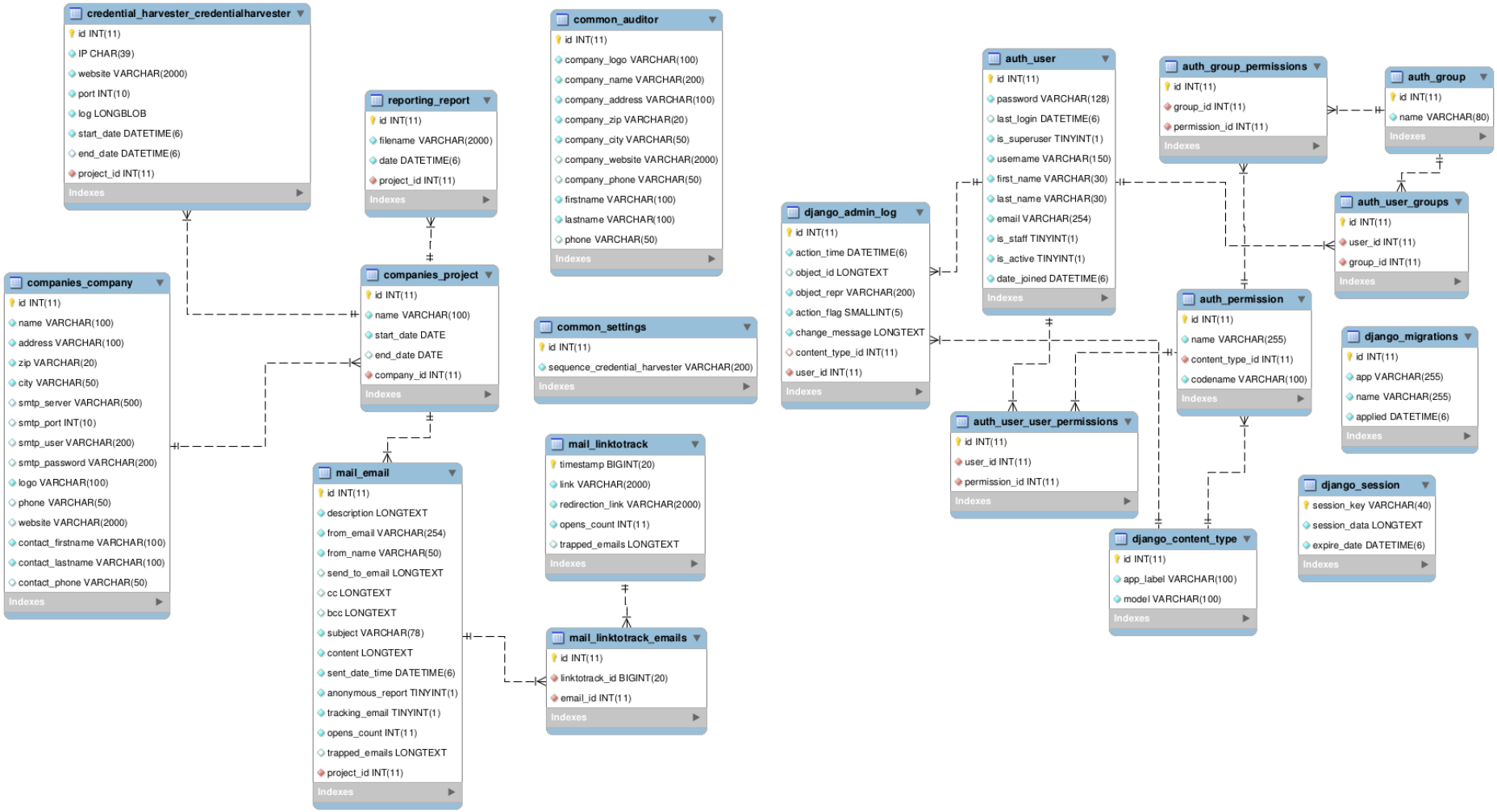
8.2.1.12. Templates

Ce répertoire contient les templates globaux de notre application tels que l'en-tête reproduit dans chaque page ainsi que la page d'accueil. De plus, il contient tous les templates utilisés par le module d'authentification intégré à Django.

Figure 73 templates



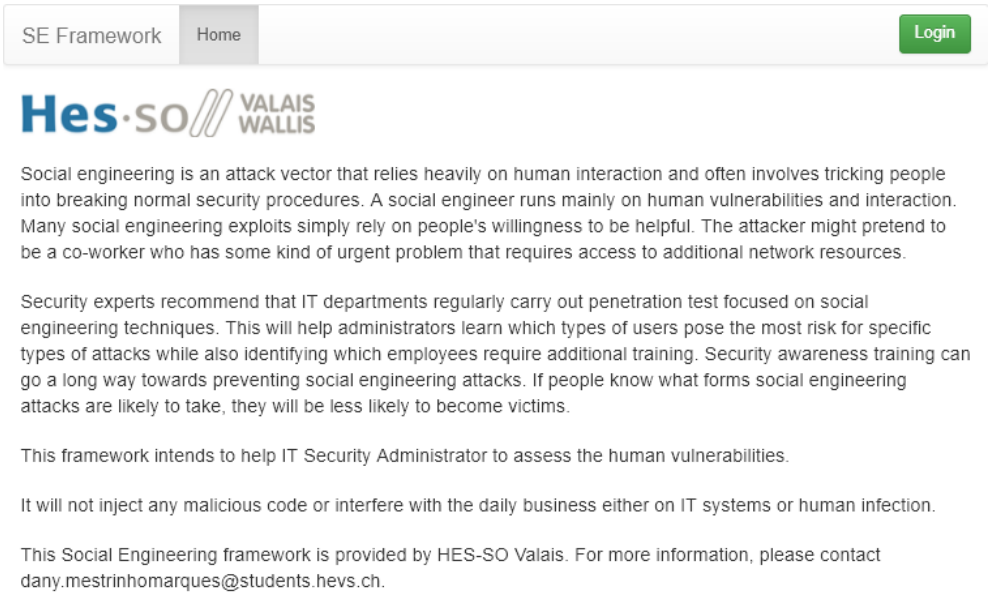
8.2.2. Digramme de classes



9. Pages

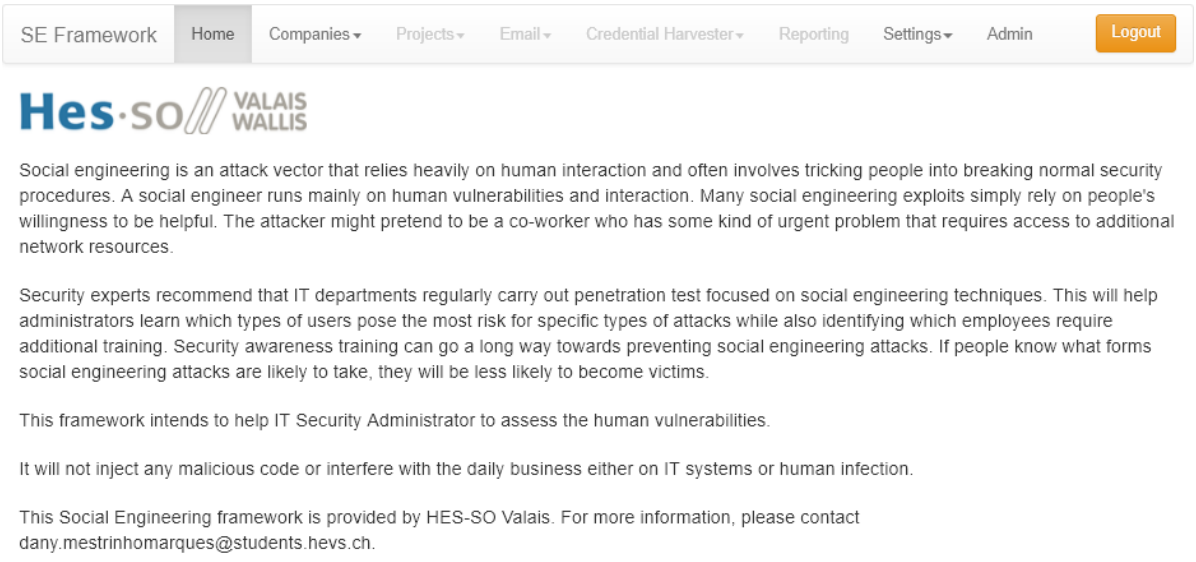
9.1. Page d'accueil de l'utilisateur déconnecté

Figure 74 Page accueil logoff



9.2. Page d'accueil de l'utilisateur connecté

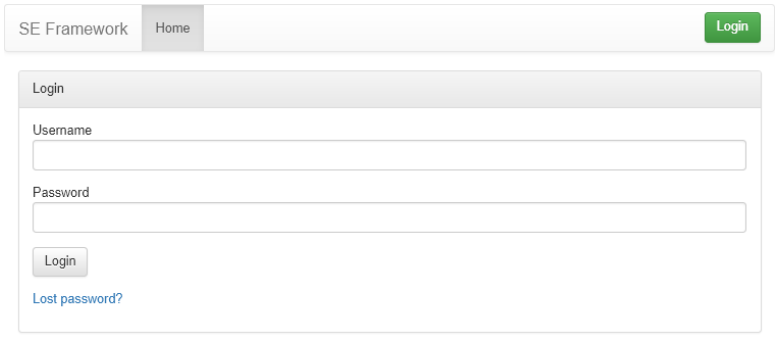
Figure 75 Page accueil logged in



Lorsque la base de données est vide, la majorité des boutons de l'en-tête sont grisés, car les fonctions ne sont pas utilisables.

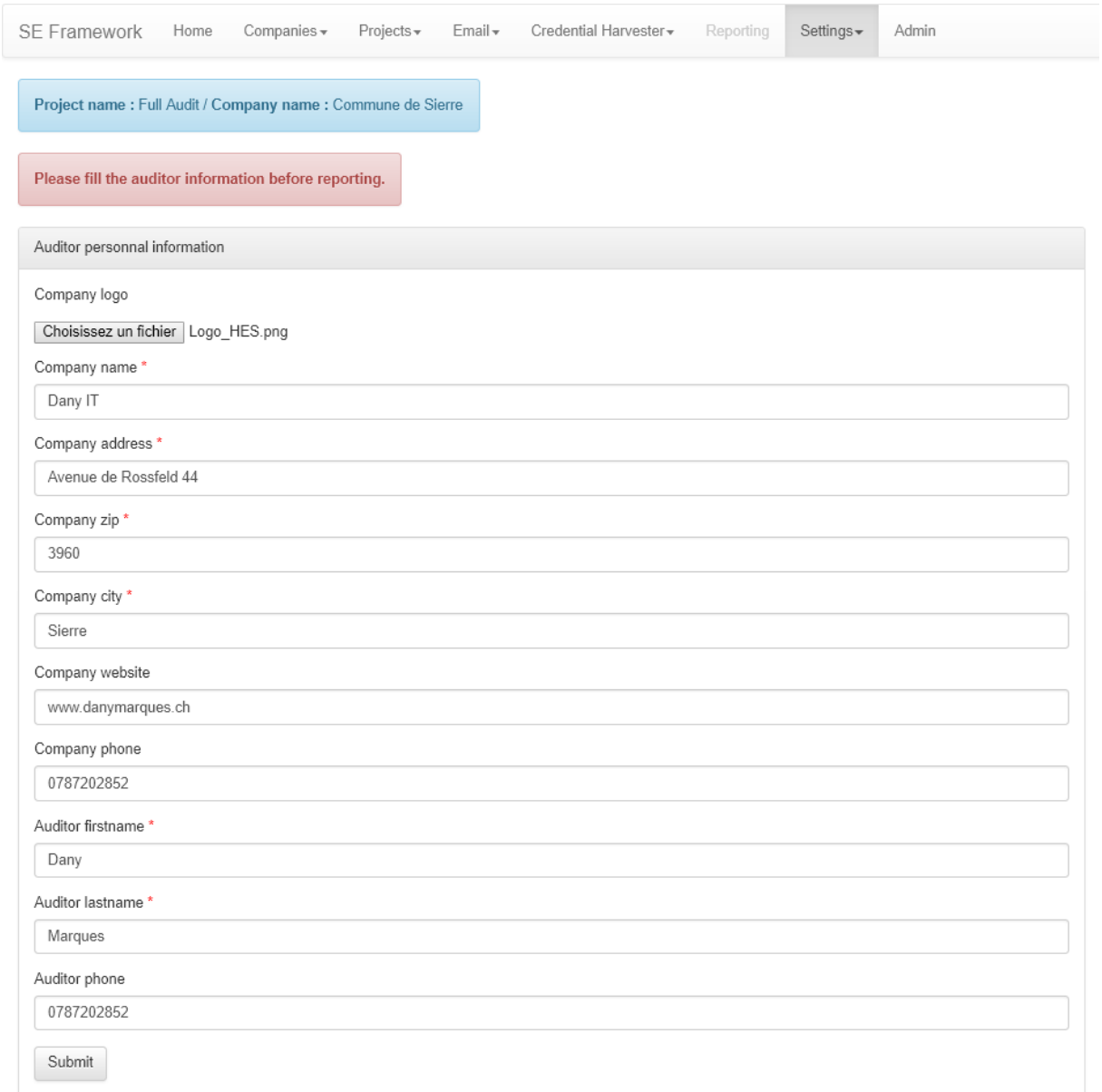
9.3. Formulaire d'authentification

Figure 76 Formulaire authentification



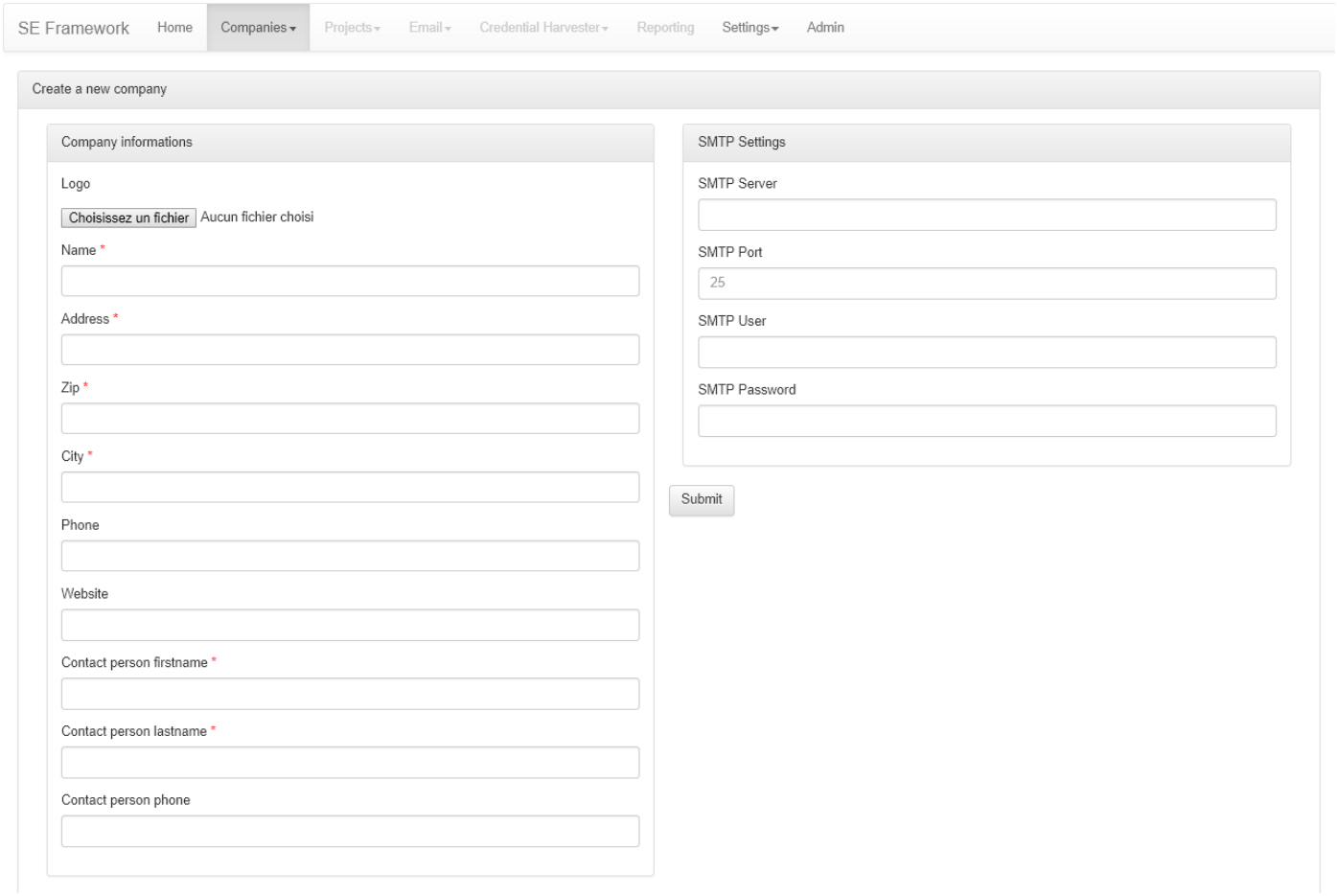
9.4. Formulaire d'édition des informations de l'auditeur

Figure 77 Formulaire d'édition des informations de l'auditeur



9.5. Formulaire de création d'une nouvelle entreprise

Figure 78 Formulaire de création d'une nouvelle entreprise

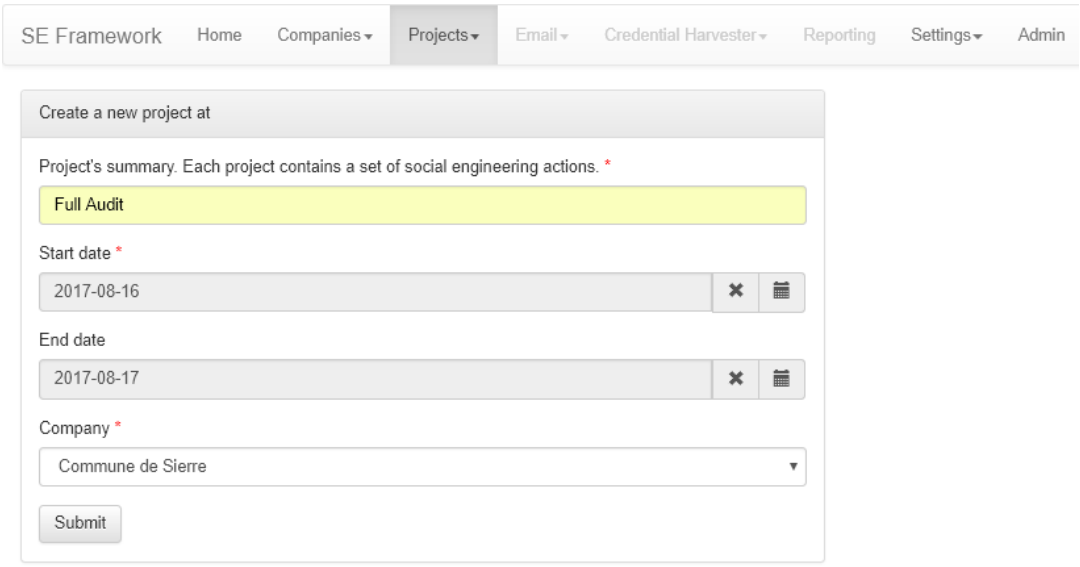


9.6. Liste de toutes les entreprises

Figure 79 Liste de toutes les entreprises

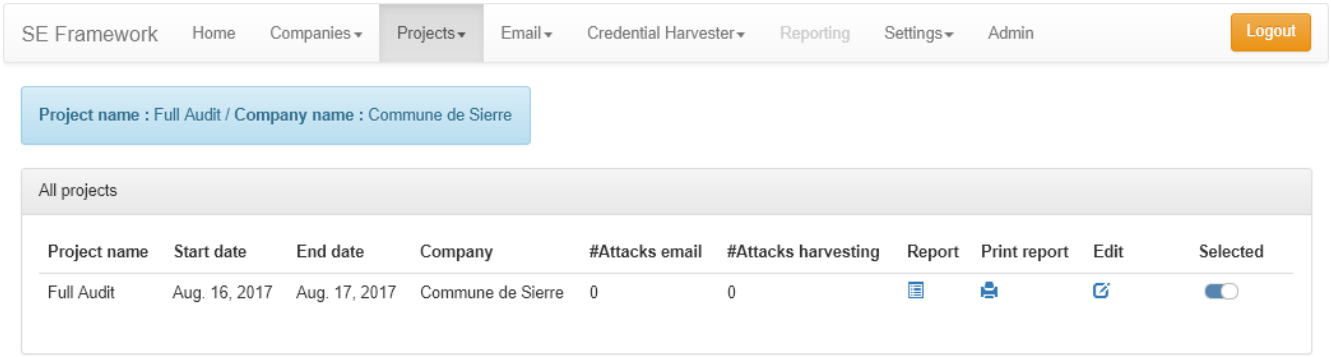
9.7. Formulaire de création d'un projet

Figure 80 Formulaire de création d'un projet



9.8. Liste de tous les projets

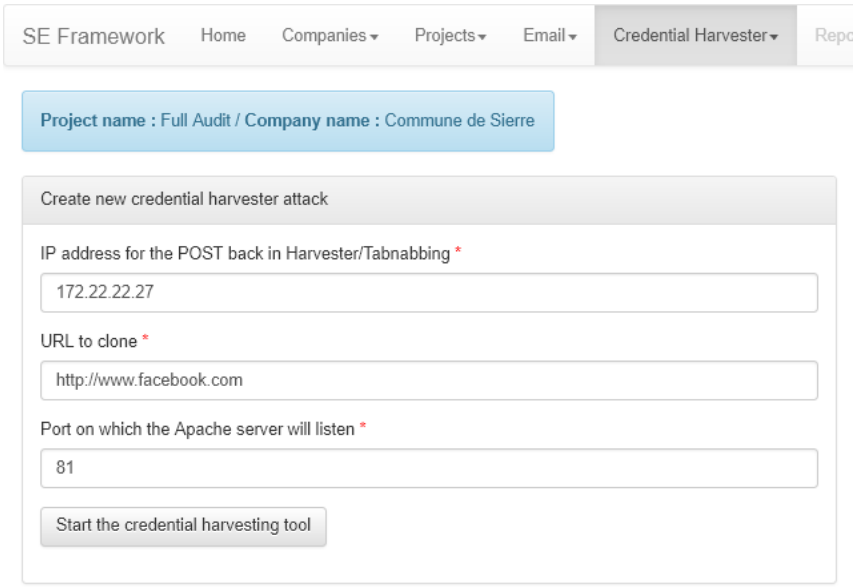
Figure 81 Liste de tous les projets



Lorsque nous sélectionnons un projet, les menus « Email » et « Credential Harverster » se déverrouillent. La balise bleue affichée sur la page indique sur quel projet nous travaillons ainsi que la société liée à ce dernier.

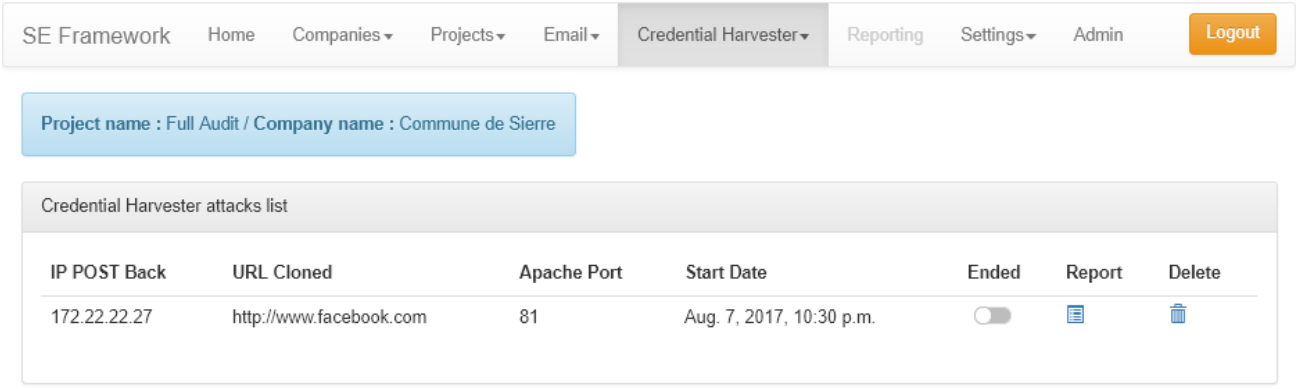
9.9. Formulaire de création d'un harvesting

Figure 82 Formulaire de création d'un harvesting



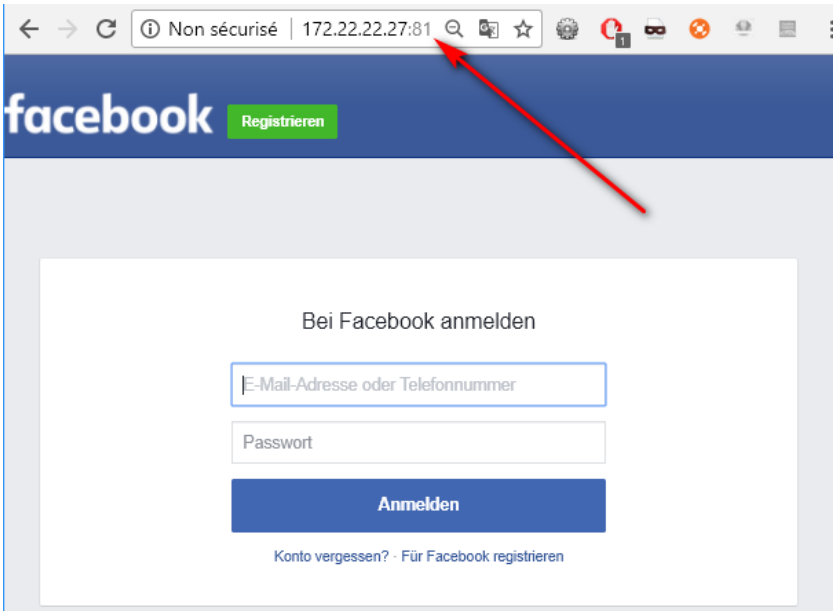
9.10. Liste de toutes les instances d'harvesting en cours

Figure 83 Liste de toutes les instances d'harvesting en cours



9.11. Site Web cloné

Figure 84 Site Web cloné



9.12. Rapport d'harvesting

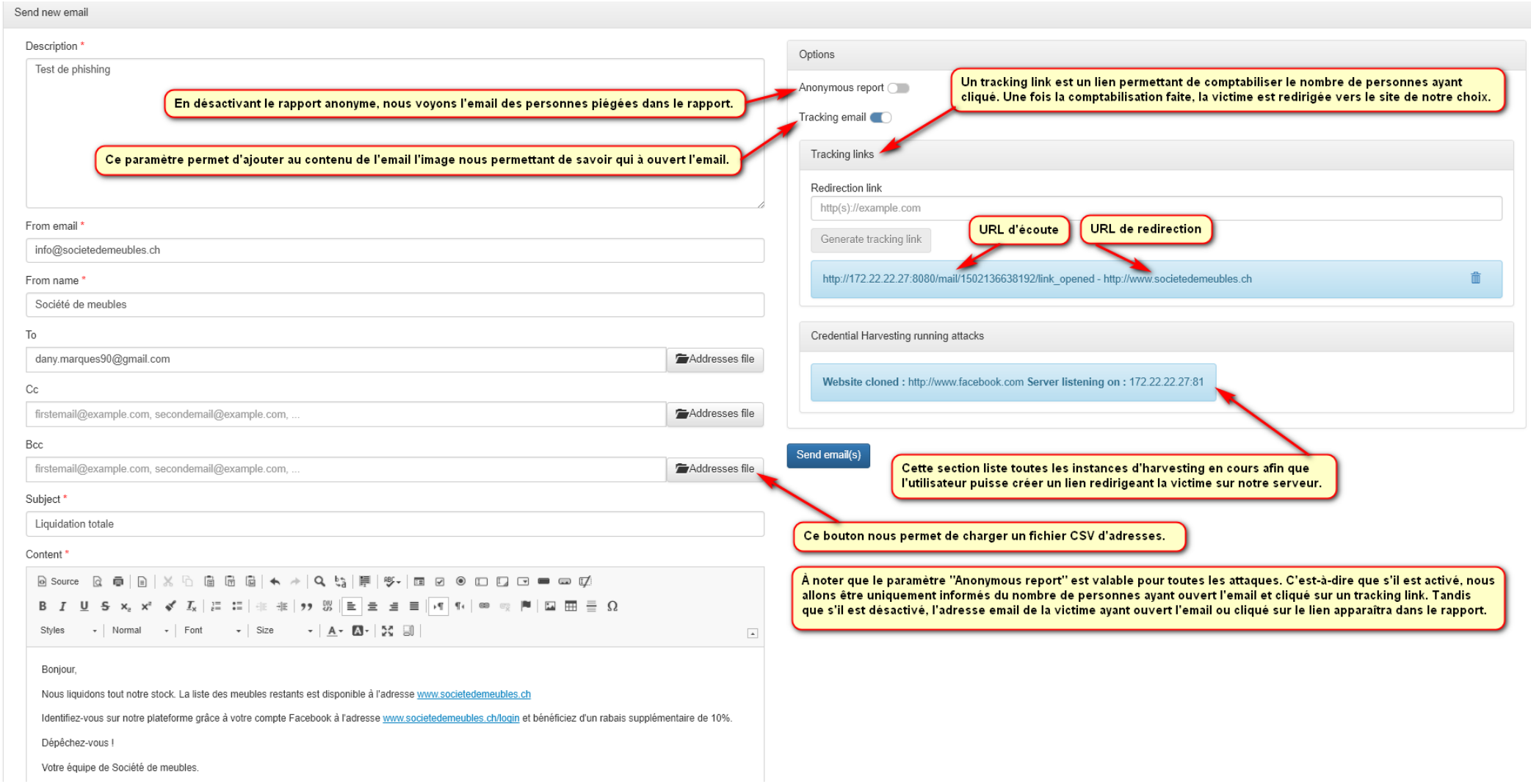
Figure 85 Rapport d'harvesting

Content harvested #1

lsd	AVo81MP-
display	
enable_profile_selector	
isprivate	
legacy_return	0
profile_selector_ids	
return_session	
skip_api_login	
signed_next	
trynum	1
timezone	-705
lgndim	eyJ3IjoxOTIwLCJoIjoxMDgwLCJhdyI6MTkyMCwiYWgiOjEwNDAsImMiOjI0fQ==
lgnrnd	133055_Cf5o
lgnjs	1502173013
email	test.demo@gmail.com
pass	demopassword
Client IP	172.22.22.25

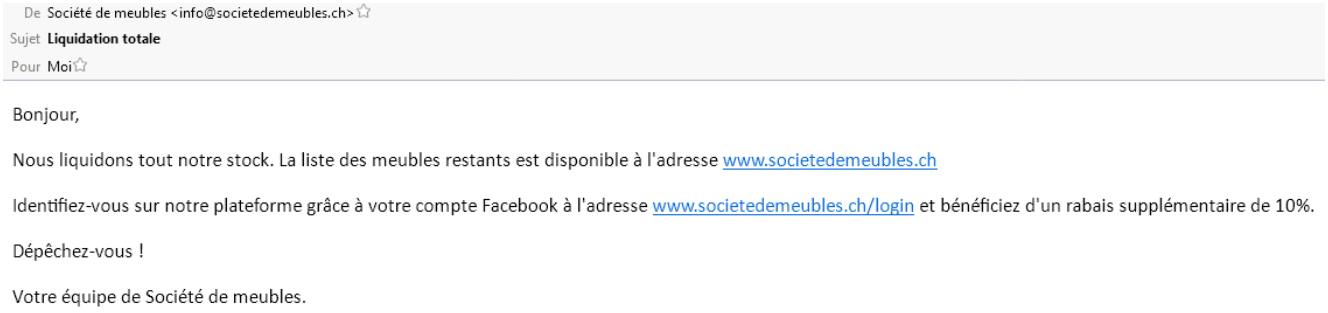
9.13. Formulaire de création d'un nouvel email

Figure 86 Formulaire de création d'un nouvel email



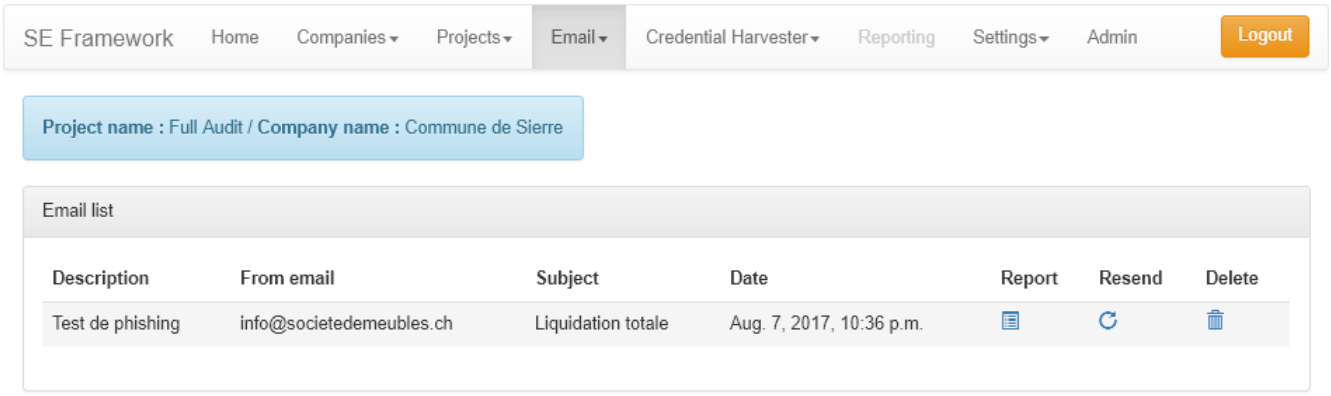
9.14. Aperçu de l'email reçu

Figure 87 Aperçu de l'email reçu



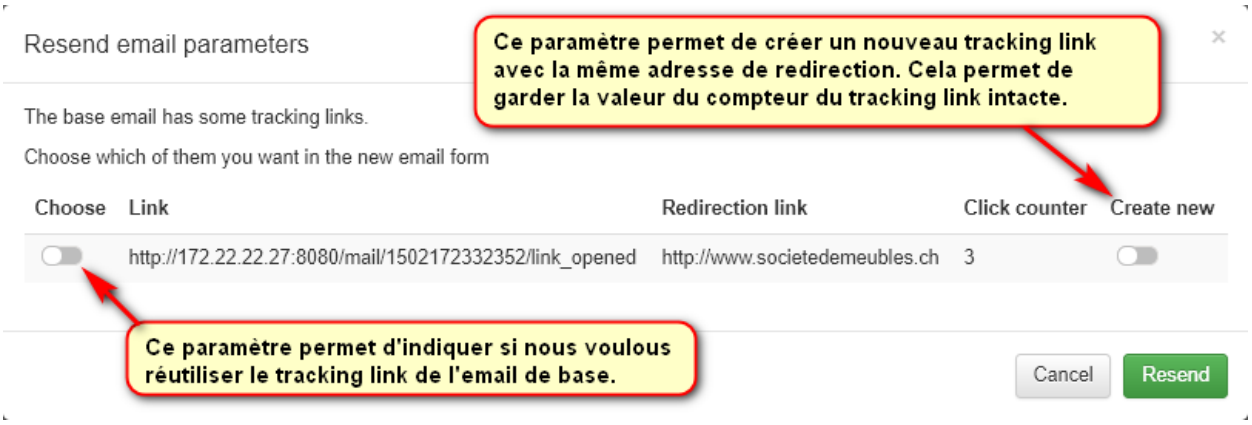
9.15. Liste de tous les emails envoyés

Figure 88 Liste de tous les emails envoyés



9.16. Paramètres de renvoi d'un email

Figure 89 Paramètres de renvoi d'un email

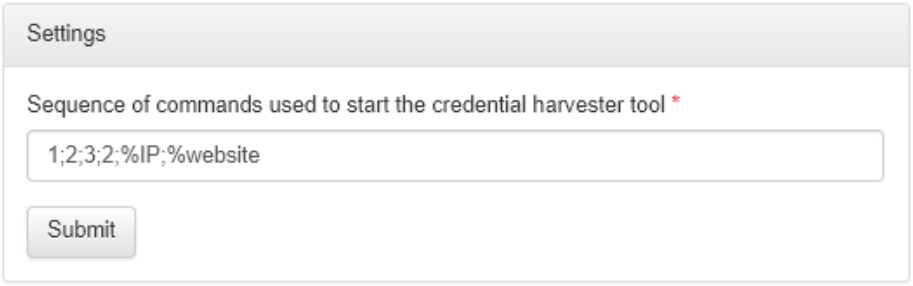


9.17. Rapport de l'envoi de l'email

Figure 90 Rapport de l'envoi de l'email

9.18. Paramètres généraux de l'application

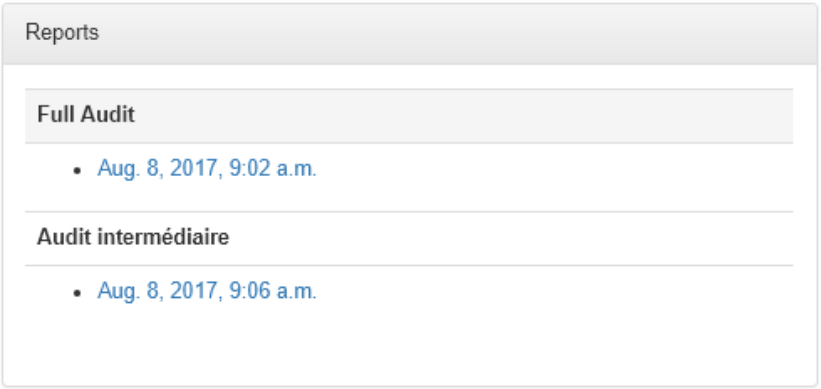
Figure 91 Paramètres généraux de l'application



9.19. Reporting

Afin que l'onglet de reporting s'active, nous devons générer au moins un rapport à l'aide du bouton « print report » présent dans la page listant les projets. La page de reporting affiche la liste des rapports générés par projet.

Figure 92 Reporting



10. Fonctionnement de l'application email

10.1. Entité email

Figure 93 Entité email

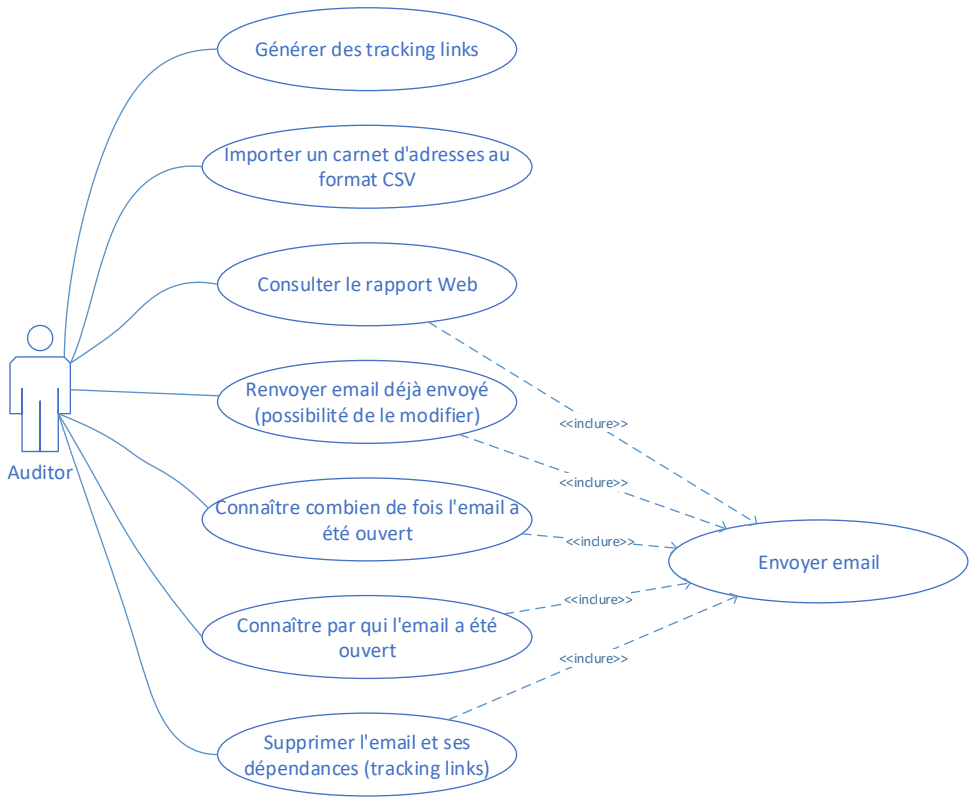
```
class Email(models.Model):
    description = models.TextField()
    from_email = models.EmailField()
    from_name = models.CharField(max_length=50)
    send_to_email = models.TextField(null=True)
    cc = models.TextField(null=True, blank=True)
    bcc = models.TextField(null=True, blank=True)
    subject = models.CharField(max_length=78)
    content = models.TextField()
    sent_date_time = models.DateTimeField(auto_now_add=True)
    anonymous_report = models.BooleanField(default=True)
    tracking_email = models.BooleanField()
    opens_count = models.IntegerField(default=0)
    project = models.ForeignKey('companies.Project')
    trapped_emails = models.TextField(null=True)
```

Figure 94 Entité email

Column	Type
id	int(11)
description	longtext
from_email	varchar(254)
from_name	varchar(50)
send_to_email	longtext
cc	longtext
bcc	longtext
subject	varchar(78)
content	longtext
sent_date_time	datetime(6)
anonymous_report	tinyint(1)
tracking_email	tinyint(1)
opens_count	int(11)
trapped_emails	longtext
project_id	int(11)

10.2. Fonctionnalités email

Figure 95 UML email

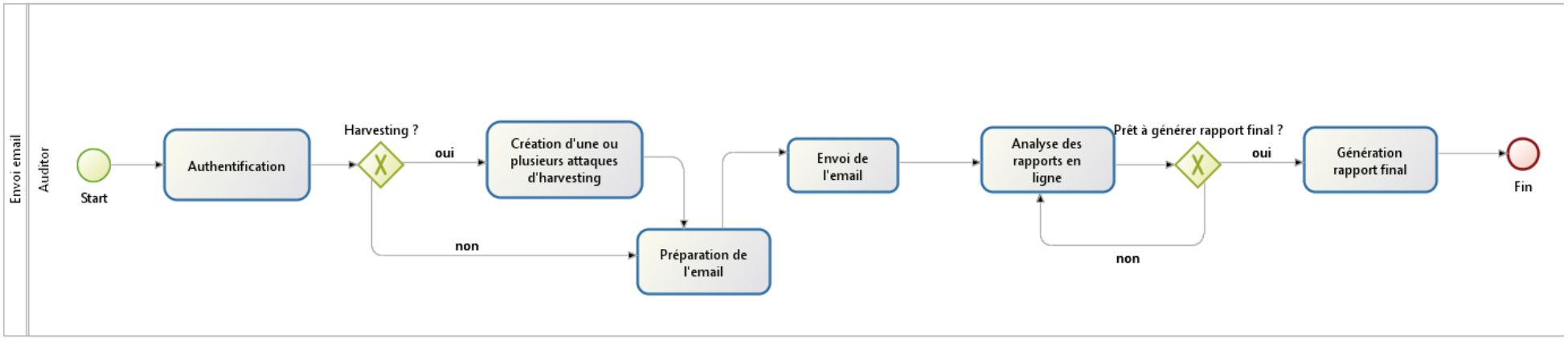


10.3. Méthodes de la classe métier email

- emails(request, id)
  - Permet d'afficher tous les emails par rapport à un projet
- new_email(request, id, id_email)
  - Permet de créer un nouvel email vierge ou à partir d'un email déjà existant.
  - Cette méthode appelle une méthode asynchrone. En effet, s'il y a beaucoup de destinataires, l'envoi de l'email peut prendre un certain temps dégradant l'expérience utilisateur.
- open_email(request, id)
  - Permet d'afficher un email envoyé.
- opened_email(request, id)
  - Permet d'incrémenter le compteur du nombre de fois que l'email a été ouvert.
- opened_link(request, timestamp)
  - Permet d'incrémenter le compteur du nombre de fois que le tracking link a été ouvert.
- create_tracking_link(request)
  - Permet de générer un nouveau tracking link.
- delete_tracking_link(request)
  - Permet de supprimer un tracking link existant
- get_tracking_links(request, id)
  - Permet de retourner un tracking link en fonction de son identifiant.
- clear_ltt(request)
  - Permet de vider le contenu de la variable de session contenant la liste des tracking links créés.
- delete_email(request, id_email)
  - Permet de supprimer un email ainsi que tous les trackings links liés à ce dernier à condition qu'ils ne soient pas utilisés dans un autre email.
- report_email(request, id_email)
  - Permet d'afficher le rapport de l'email.
- download_trapped_emails_ltt(request, id_ltt)
  - Permet de télécharger un fichier CSV avec la liste des adresses email ayant cliqué sur le lien.
- download_trapped_emails_email(request, id_email)
  - Permet de télécharger un fichier CSV avec la liste des adresses email ayant ouvert l'email.

10.4. Processus d’envoi d’un email

Figure 96 Processus d'envoi d'un email



11. Fonctionnement de l'application credential_harvester

11.1. Entité CredentialHarvester

Figure 97 Entité CredentialHarvester

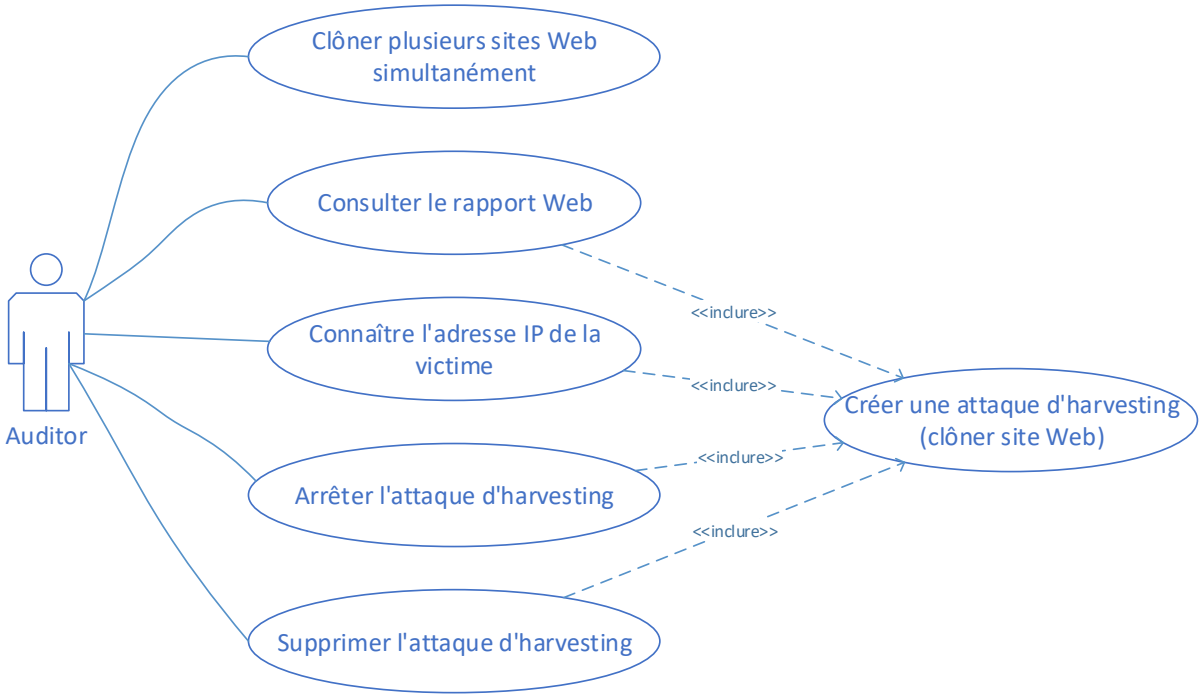
```
class CredentialHarvester(models.Model):
    IP = models.GenericIPAddressField()
    website = models.CharField(max_length=2000)
    port = models.PositiveIntegerField()
    log = models.BinaryField()
    start_date = models.DateTimeField(auto_now_add=True)
    end_date = models.DateTimeField(null=True)
    project = models.ForeignKey('companies.Project')
```

Figure 98 Entité Credential Harvester

Column	Type
id	int(11)
IP	char(39)
website	varchar(2000)
port	int(10) unsigned
log	longblob
start_date	datetime(6)
end_date	datetime(6)
project_id	int(11)

11.2. Fonctionnalités de l'application credential_harvester

Figure 99 Fonctionnalités de l'application crendtial_harvester

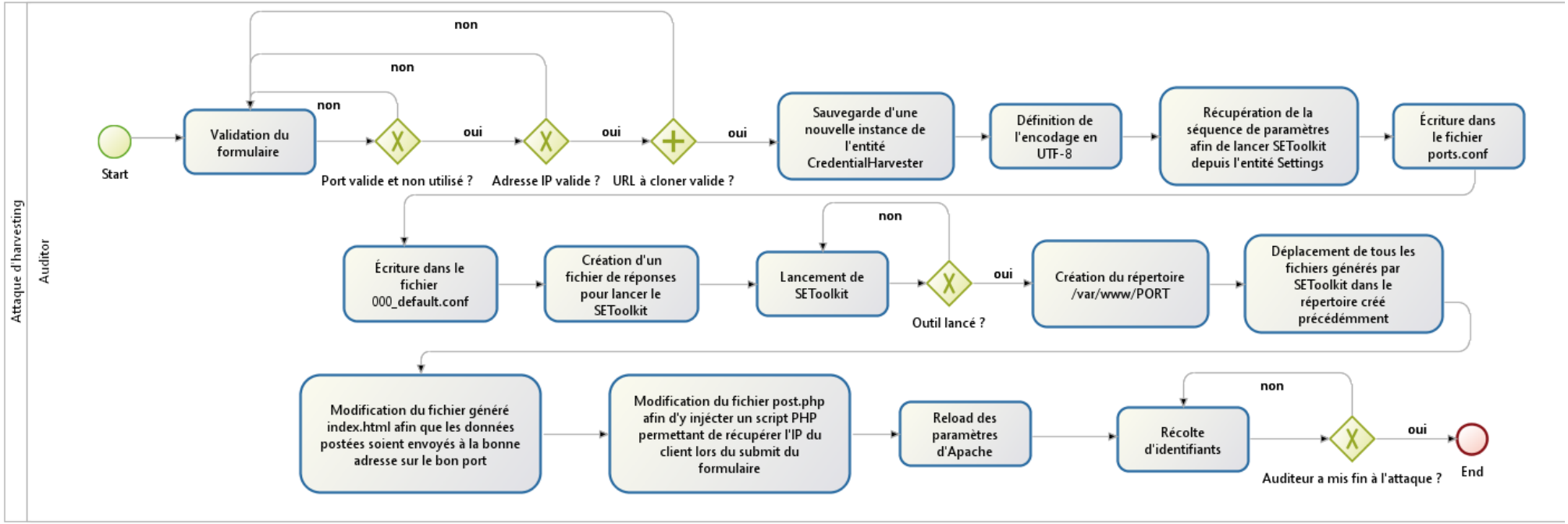


11.3. Méthodes de la classe métier credential_harvester

- attacks_ch(request, id_project)
  - Permet d'afficher la liste de toutes les attaques d'harvesting par rapport au projet sélectionné.
- new_attack_ch(request, id_project)
  - Permet de créer une nouvelle attaque d'harvesting. Cette méthode appelle la méthode asynchrone illustrée par le processus à la page suivante.
- check_port(request)
  - Méthode permettant de vérifier si le port est déjà utilisé par une autre application.
- get_ip_address(request)
  - Permet de récupérer l'adresse IP de la carte réseau.
- report_attack_ch(request, id_attack_ch)
  - Permet d'afficher le rapport Web de l'attaque d'harvesting.
- read_report_file(port)
  - Méthode permettant de lire le fichier de log généré par SEToolkit ainsi que de formater les données dans le but de les afficher ultérieurement dans un rapport.
- delete_attack_ch(request, id_attack_ch)
  - Permet de supprimer l'attaque d'harvesting
- open_attack_ch(request, id_attack_ch)
  - Permet d'afficher une attaque d'harvesting en cours afin de simplement consulter les paramètres renseignés ou alors d'en mettre fin.
- end_attack_ch(request, id_attack_ch)
  - Méthode permettant d'arrêter une attaque d'harvesting.

11.4. Processus d'une attaque d'harvesting (technique)

Figure 100 Processus d'une attaque d'harvesting (technique)



12. Fonctionnement de l'application reporting

12.1. Entité Report

Figure 101 Entité Report

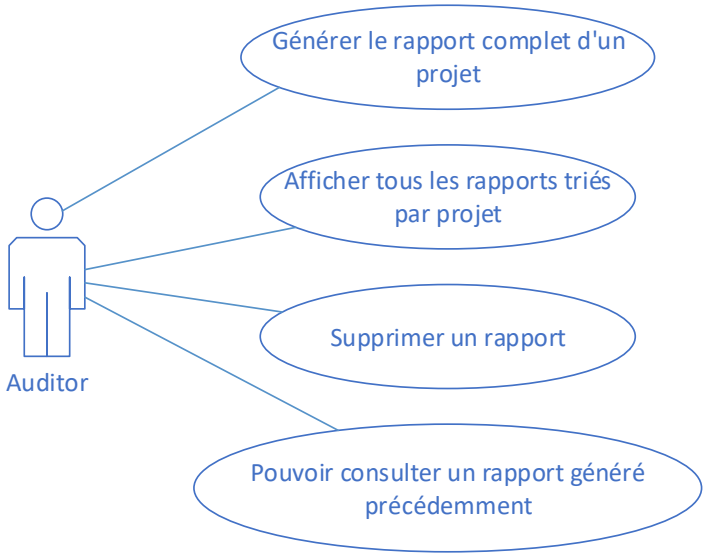
```
class Report(models.Model):
    filename = models.CharField(max_length=2000)
    date = models.DateTimeField(auto_now_add=True)
    project = models.ForeignKey('companies.Project')
```

Figure 102 Entité Report

Column	Type
id	int(11)
filename	varchar(2000)
date	datetime(6)
project_id	int(11)

12.2. Fonctionnalités de l'application reporting

Figure 103 Fonctionnalités de l'application reporting

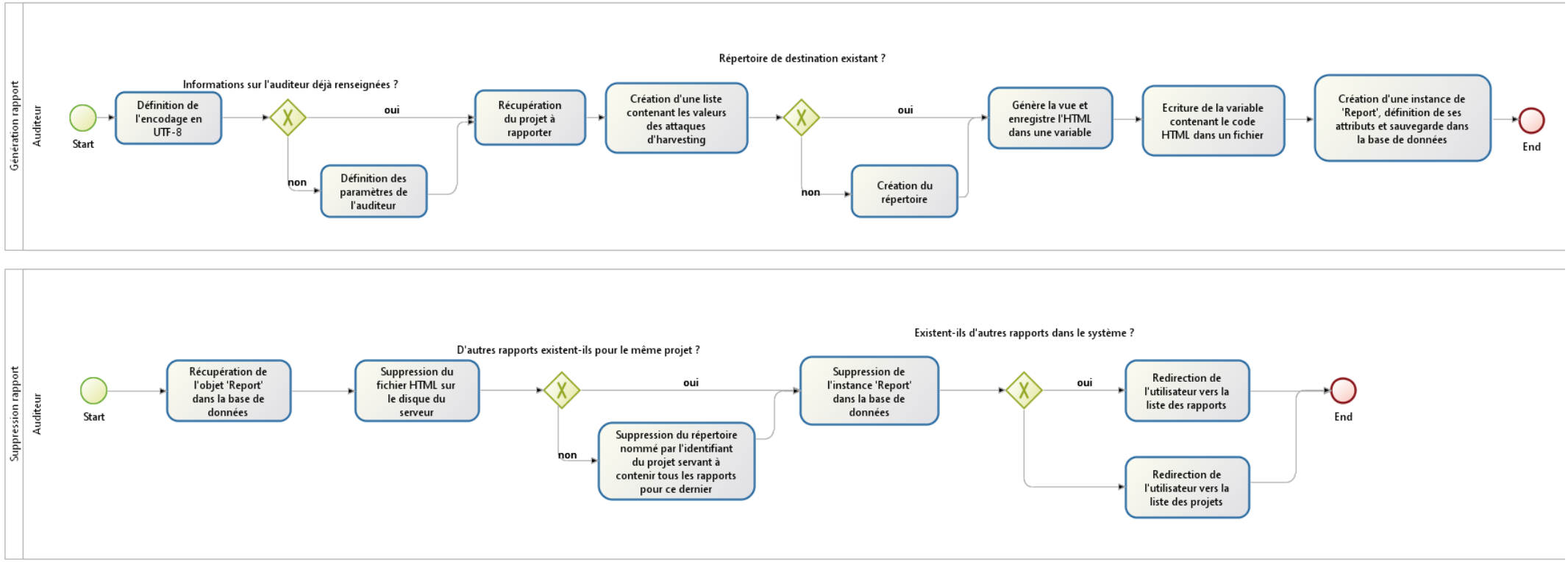


12.3. Méthodes de la classe métier reporting

- reporting(request)
  - Permet d'afficher la liste des rapports par projet.
- open_gen_report(request, id_report)
  - Permet d'ouvrir et d'afficher un rapport sauvegardé sur le disque du serveur.
- reports_count(request)
  - Méthode retournant le nombre total de rapports générés.
- delete_report(request, id_report)
  - Méthode permettant de supprimer un rapport de la base de données ainsi que le fichier sauvegardé sur le disque
- gen_report(request, id_project)
  - Permet de générer un rapport pour le projet en question et de le sauvegarder sur le disque dur de notre serveur.

12.4. Processus de génération et suppression d'un rapport (technique)

Figure 104 Processus de génération et suppression d'un rapport (technique)



13. Ajout d'un nouvel outil

1. Premièrement, nous devons créer une nouvelle application dans le projet Django à l'aide de la commande ci-dessous. Dans le cadre de cet exemple, notre application s'appelle « demoTest ».

```
python manage.py startapp demoTest
```

2. Afin de garder une structure propre et professionnelle, nous devons créer un fichier nommé « urls.py » à l'intérieur de notre nouvelle application. Ce fichier va contenir toutes les URL de l'application « demoTest ».

3. Dans le même but que le point précédent, nous devons créer un répertoire « templates » à la racine de notre nouvelle application. À l'intérieur de ce répertoire, nous devons créer un sous-répertoire nommé « demoTest ». Il est important de respecter cette directive afin d'éviter les conflits entre les applications de notre projet.

4. À présent, nous devons paramétrer notre projet afin qu'il prenne en compte l'application créée au premier point. Pour cela, nous devons ajouter une chaîne de caractères comprenant le nom de notre nouvelle application à la liste nommée « INSTALLED_APPS » comprise dans le fichier « settings.py » se trouvant dans le répertoire « social_engineering ».

5. Nous pouvons maintenant modifier le fichier « urls.py » se trouvant dans le répertoire « social_engineering » pour lui ajouter une URL parente. La ligne à ajouter à la variable « urlpatterns » est composée tel qu'indiqué ci-dessous.

```
url(r'^demoTest/', include('demoTest.urls')),
```

6. Désormais, notre projet intègre la nouvelle application. Nous pouvons donc commencer à développer son contenu. Afin de nous faciliter la prise en main, nous décrivons ci-dessous le lien entre les fichiers Python de Django et le modèle MVC classique.
    - M models.py
    - V tous les fichiers HTML placés dans le répertoire « templates/demoTest ».
    - C views.py

14. Améliorations

14.1. Intégration d'outils supplémentaires

Dans le cas où le temps à disposition pour réaliser ce travail serait plus conséquent, nous aurions voulu intégrer à notre framework plus d'outils. Plus particulièrement des outils inclus dans SEToolkit puisque la manière de les exécuter est identique à celle que nous avons utilisée afin de lancer nos attaques d'harvesting.

14.2. Rapports

Les rapports générés par notre framework peuvent atteindre une taille conséquente, car la quantité de résultats à présenter dépend du cadre de l'utilisation des outils ainsi que de la durée de l'attaque. Il serait donc judicieux de pouvoir effectuer une sélection des champs à inclure dans le rapport avant la génération de celui-ci afin qu'uniquement les informations importantes y soient présentées.

Nos rapports étant actuellement générés au format HTML avec une mise en page A4, nous devons effectuer une étape manuelle supplémentaire afin de les convertir au format PDF. Nous avons testé certaines librairies Python permettant de convertir des pages HTML en PDF, mais les résultats n'ont pas été concluants. En effet, tel que mentionné précédemment, nous n'avons pas la main ni sur la quantité ni sur la forme des données récupérées de nos attaques. C'est pourquoi l'utilisation de ces librairies est impossible à l'heure actuelle. Cependant, si une sélection des champs était implémentée telle que décrite ci-dessus, nous sommes persuadés que ces libraires permettraient de générer les rapports au format PDF sans problèmes.

14.3. Exploitation de vulnérabilités

L'outil d'hameçonnage que nous avons développé étant flexible, nous pouvons imaginer d'intégrer à notre outil d'autres utilitaires permettant de générer des payloads dans le but de les transmettre par email.

15. Conclusion

Avant de réaliser ce travail, la programmation Python était pour nous un domaine complètement abstrait. Le framework Django était également un outil dont nous ne connaissions pas le fonctionnement. Grâce à ce travail, nous avons pu améliorer nos connaissances dans ces domaines. Tel qu'espéré, Django permet effectivement d'accélérer le processus de développement de façon considérable

À travers les recherches effectuées dans le cadre de ce travail, nous nous sommes aperçu que le social engineering est une manière très utilisée d'accéder à des informations confidentielles. Néanmoins, les outils disponibles à l'heure actuelle possèdent des fonctionnalités restreintes. C'est pour cette raison que notre projet comporte le développement d'outils in house ainsi que l'interfaçage avec des outils Open Source du marché.

La force de notre travail est qu'il permet de mesurer le niveau de réponse des utilisateurs d'une société lorsqu'ils se retrouvent face à des attaques d'ingénierie sociale. Les rapports générés par les outils utilisés par notre framework permettent à la société d'organiser des formations ciblées.

Des failles techniques pouvant être exploitées par des personnes malveillantes seront toujours présentes malgré l'évolution technologique. Toutefois, il est de plus en plus difficile de les exploiter. C'est pourquoi notre travail porte sur l'aspect humain et non pas sur des composants informatiques.

En tenant compte de la difficulté de ce projet, nous sommes satisfait du résultat du travail que nous avons accompli. Les tests effectués sont concluants et nous avons acquis des connaissances professionnelles et personnelles de qualité. Nous avons mené à bien un projet conséquent et important, ce qui nous a permis de nous dépasser et d'être en constante évolution.

Ce projet étant désormais aboutit dans le cadre du travail de Bachelor, il est prêt à être utilisé dans un domaine professionnel tout en le développant et l'améliorant selon la tendance des attaques.

16. Références bibliographiques

Hacker. Dans *Larousse*. Repéré le 01.08.2017 à http://www.larousse.fr/dictionnaires/francais/hacker/38812

Vishing. Dans *Wiktionary*. Repéré le 01.08.2017 à https://fr.wiktionary.org/wiki/vishing

Phishing. Dans *Wiktionary*. Repéré le 01.08.2017 à https://fr.wiktionary.org/wiki/phishing

Smishin. Dans *Wiktionary*. Repéré le 01.08.2017 à https://fr.wiktionary.org/wiki/smishing

Mailfence Team. (5 février 2017). Ingénierie sociale : Qu'est-ce que le baiting ou appâtage ? Repéré à https://blog.mailfence.com/fr/ingenierie-sociale-quest-ce-que-le-baiting-ou-appatage/

Spear Phishing. Dans *Wikipédia*. Repéré le 01.08.2017 à https://fr.wikipedia.org/wiki/Spear_phishing

Mailfence Team. (17 janvier 2017). Qu'est-ce que l'ingénierie sociale ? Repéré à https://blog.mailfence.com/fr/quest-ce-que-lingenierie-sociale/

Les récupérateurs ou Dumpster Diving. Repéré à http://www.entoutefrugalite.com/les-recuperateurs-ou-dumpster-diving/

Harvest. Dans *Larousse*. Repéré le 01.08.2017 à http://www.larousse.fr/dictionnaires/anglais-francais/harvest/585590

Plugin. Dans *Wiktionary*. Repéré le 05.08.2017 à https://fr.wiktionary.org/wiki/plugin

Benchmark. Dans *Wiktionary*. Repéré le 05.08.2017 à https://fr.wiktionary.org/wiki/benchmark

Template. Dans *Wiktionary*. Repéré le 07.08.2017 à https://fr.wiktionary.org/wiki/template

Responsive web design. Dans *Wiktionary*. Repéré le 07.08.2017 à https://fr.wiktionary.org/wiki/responsive_web_design

Ellingwood, J. (2015). How To Use MySQL or MariaDB with your Django Application on Ubuntu 14.04. Repéré à https://www.digitalocean.com/community/tutorials/how-to-use-mysql-or-mariadb-with-your-django-application-on-ubuntu-14-04

Lorant, M. et Xhonneux, M. (2017). Développez votre site web avec le framework Django. Repéré à https://openclassrooms.com/courses/developpez-votre-site-web-avec-le-framework-django

Makai, M. (2012-2017). Task Queues - Full Stack Python. Repéré à https://www.fullstackpython.com/task-queues.html

Thapar, A. Social Engineering : An attack vector most intricate to tackle !. Repéré à http://www.infosecwriters.com/text_resources/pdf/Social_Engineering_AThapar.pdf

Matskas, C. (2014). Importing CSV files using jQuery and HTML5. Repéré à https://cmatskas.com/importing-csv-files-using-jquery-and-html5/

Van der Wijk, I. (2013). Adding css classes to formfields in Django Templates. Repéré à http://vanderwijk.info/blog/adding-css-classes-formfields-in-django-templates

Angert, E. (2017). Configure Postfix to Send Mail Using Gmail and Google Apps on Debian or Ubuntu. Repéré à https://www.linode.com/docs/email/postfix/configure-postfix-to-send-mail-using-gmail-and-google-apps-on-debian-or-ubuntu

Ti, S. (2014). Configure Postfix to Send Mail Using an External SMTP Server. Repéré à https://www.linode.com/docs/email/postfix/postfix-smtp-debian7

Base de données et Python. Repéré à http://apprendre-python.com/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite

17. Déclaration sur l'honneur

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 Responsable de filière 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 :

- Xavier Barmaz

Sierre, le 9 août 2017

Dany Marques