
TRAVAIL DE FIN D'ÉTUDES

Analyse, conception et développement d'une plateforme autonome
d'évaluation des systèmes informatiques et son implémentation par le biais
des nano-ordinateurs.

Présenté par : Rémi STREIGNARD
Promoteur : Benjamin VAN DAMME
Maître de stage : Laura RAMONFOSSE

ANNÉE ACADEMIQUE 2019 - 2020

Remerciements

Avant toute chose, je tiens à remercier, Alexandre Marchal et le Dr. Julien Lecointre, directeur du département technique de Marche-en-Famenne, de m'avoir offert l'opportunité d'apprendre aux côtés de l'équipe du FoRS et d'avoir pu travailler sur ce projet.

Ensuite, je remercie l'équipe du FoRS de m'avoir accueilli à leurs côtés et tout particulièrement Laura Ramonfosse, mon maître de stage, qui m'a accompagné tout au long de la réalisation de mon projet en m'apportant ses conseils et une aide précieuse.

Je remercie aussi mes parents, amis et collèges avec qui j'ai pu partager mes découvertes et mes questionnements au cours de cette période.

Finalement, je tiens à remercier l'ensemble des professeurs de la section Sécurité des systèmes et plus particulièrement mon promoteur, Benjamin Van Damme pour les compétences, conseils et aides qu'ils ont pu m'apporter au cours de ces trois années.

Table des matières

1	Introduction	1
1.1	Présentation de l'entreprise.....	1
2	Présentation du projet	2
2.1	Objectifs du projet.....	3
3	Concepts théoriques	4
3.1	Introduction au test d'intrusion	4
3.2	Introduction au Vulnerability assesment.....	6
3.2.1	La métrique de base	7
3.3	Introduction au <i>scripting</i> et à l'automatisation.....	8
3.4	Introduction au nano-ordinateur	8
4	État de l'art des logiciels existants	10
4.1	Nessus	10
4.1.1	Méthodologie	11
4.2	Metasploit Framework	12
4.2.1	Portée des modules.....	13
4.3	Conclusion	13
5	Analyse	14
5.1	Partie nano-ordinateur.....	15
5.1.1	Raspberry	16
5.1.2	Choix du système d'exploitation	16
5.1.3	Accès à distance	17
5.1.4	Émetteur / récepteur Wifi Moglor AC1200	17
5.2	Langages de programmation.....	18
5.2.1	Python	19
5.2.2	JavaScript.....	19
5.2.3	C & C++	20
5.2.4	PHP	20
5.2.5	SQL.....	20
5.2.6	Conclusion	21
5.3	Processus d'automatisation	21
5.4	Choix de la base de données	22
5.4.1	Comparatif des SGBDR	24
5.4.2	Choix du SGBDR.....	25

5.5	Dashboard	26
5.5.1	Serveur web.....	26
6	Implémentation.....	27
6.1	Configuration de l'environnement	27
6.1.1	Installation de la Raspberry	27
6.2	Conception de la base de données.....	29
6.2.1	Installation du service de base de données et du SGBD	29
6.2.2	Chiffrement du trafic	30
6.2.3	Configurations supplémentaires et sécurité du service	30
6.2.4	Conception et structure	31
6.3	Procédure d'automatisation	36
6.4	Conception du Dashboard.....	39
6.4.1	Développement de la maquette.....	39
6.4.2	Analyse du Dashboard développé	42
7	Analyse rétrospective	44
7.1	Améliorations possibles.....	44
8	Conclusion	45
9	Webographie.....	46
10	Annexes	48
10.1	Rapport de stage	48
10.1.1	Présentation de l'entreprise.....	48
10.1.2	Présentation du projet	48
10.1.3	Rapports journaliers	48
10.1.4	Programme ON STAGE	68
10.1.5	Organisation du travail (Gantt).....	69
10.1.6	Evaluation du stage	71
10.1.7	Conclusion	72
10.2	Documentation.....	73
10.2.1	Documentation de l'architecture	73
10.2.2	Documentation du code.....	89
10.2.3	Documentation du Dashboard	102

Table des figures

Figure 1 : Localisation du DTM – MASI.....	2
Figure 3 : EC-Council CEH Méthodologie.....	4
Figure 4 : Arduino & Raspberry pi	9
Figure 5 : Interface de Nessus	10
Figure 6 : Interface web de Metasploit Framework.....	12
Figure 7 : Architecture du projet.....	14
Figure 8 : Graphique des tendances en langage de programmation - GitHub 2020	18
Figure 9 : Tableau des tendances GitHub 2020.....	19
Figure 10 : Tendances - base de données 2020	22
Figure 11 : Répartition NoSQL et SQL.....	23
Figure 12 : Tendances SGBDR 2020.....	24
Figure 13 : Interface MySQL workbench	31
Figure 14 : Modèle ER : Partie client	32
Figure 15 : Modèle ER : Partie hôte.....	32
Figure 16 : Structure des feeds au format JSON.....	33
Figure 17 : Modèle ER : Partie service.....	34
Figure 18 : Modèle ER complet	35
Figure 19 : Modèle BPMN des trois phases.....	36
Figure 20 : Modèle BPMN de la phase 1 : Recon	37
Figure 21 : Modèle BPMN de la phase 2 : Scanning	38
Figure 22 : Modèle BPMN de la phase 3 : Exploitation	38
Figure 23 : Maquette site web main page.....	39
Figure 24 : Maquette site web scan page	40
Figure 25 : Maquette site web host page.....	41
Figure 26 : Page de comparaison de scan	42
Figure 27 : Page hôte.....	43

Glossaire

Terme	Acronyme	Définition
Backdoor	/	Accès à un logiciel, un service ou un système dont l'utilisateur n'a pas connaissance.
Red teaming	/	Équipe de consultants dont le rôle est d'orchestrer une simulation d'attaque contre une organisation. En opposition à la Blue team qui dont le rôle est la défense.
Scripting	/	Activité qui consiste en l'écriture de script.
Script	/	Bout de code non compilé dont l'objectif est l'automatisation d'actions et d'outils, en opposition avec les programmes qui sont compilés.
Intrusion Detection System	IDS	Les systèmes de détection d'intrusion sont des outils destinés à repérer les anomalies ou des activités anormales sur des systèmes.
Common Vulnerabilities and Exposures	CVE	Dictionnaire des vulnérabilités de sécurité créé et maintenu par l'organisme MITRE. Ces références sont composées d'une description ainsi que des résultats issus des CVSS.
Common Vulnerability Scoring System	CVSS	Ce système est un standard dont l'objectif est l'évaluation de la sévérité d'une vulnérabilité en se basant sur différente métrique et en leur attribuant des scores.
Framework	/	Ensemble de composants logiciels et d'outils.
Script kiddies	/	Terme désignant les néophytes en cybersécurité qui utilise des outils sans en connaître le fonctionnement.
Sniffer	/	Outil destiné à l'analyse du trafic réseau.
Secure Shell	SSH	Protocole et service de communication sécurisée offrant une connexion à distance et qui prend en charge l'exécution de commandes et le transfert de fichiers.
Virtual Network Computing	VNC	Service de contrôle distant sur un système informatique utilisé notamment pour l'administration ou la maintenance du système.
Command-Line interface	CLI	Interface qui permet l'appel et l'exécution de commandes sous forme textuelle.
Cross-site scripting	XSS	Vulnérabilité des applications web par laquelle du code est injecté afin d'être exécuté côté client.
Back end	/	Partie du développement qui se concentre sur les fonctionnalités non visibles à travers l'utilisation de langages comme le PHP, Ruby, Python, SQL.
Front end	/	Partie du développement qui se concentre sur les fonctionnalités visibles et l'interface utilisateur. Comprends l'utilisation des langages HTML, CSS, JavaScript.
Dynamic Link Library	DLL	Bibliothèque logicielle qui offre des fonctions pouvant être chargées dynamiquement par d'autres logiciels durant leur exécution.

Shell code	/	Code exécutable écrit en binaire dont l'objectif est de détourner l'utilisation d'un programme.
Système de gestion de base de données	SGBD	Désigne un programme dont le rôle est la manipulation et la gestion des informations stockées dans une base de données.
SGBD relationnel	SGBDR	Système de gestion de base de données qui repose sur utilisation d'un modèle relationnel afin de mettre en évidence les interactions entre ses tables.
SGBDR orienté objet	SGBDRO	Système de gestion de base de données relationnel dont la nature est orientée objet où chaque information stockée est alors considérée comme un objet.
Content Management System	CMS	Application conçue pour aider les néophytes lors de la création et la gestion d'un site web.
Fork	/	Désigne un logiciel créé sur base d'un code source d'un logiciel existant.
Application Programming Interface	API	Consiste en un ensemble de données et de fonctionnalités mises à disposition pour le développement d'autres logiciels.
Modèle Entité-Association	EA	Modèle qui illustre la manière dont des entités vont interagir les unes envers les autres. Il sert ensuite à concevoir une base de données.
Root	/	Utilisateur privilégié sous Linux, similaire à l'administrateur sous Windows.
Headless Graphical User Interface	/	Solution qui fonctionne sans nécessiter une interface graphique.
GUI		Interface graphique.
Command & Control	C&C	Désigne un système distant utilisé pour communiquer avec un système compromis.
Auto-login	/	Authentification automatique d'un utilisateur au démarrage d'un système.
National Institute of Standards and Technology	NIST	Agence du département du Commerce des États-Unis dont l'objectif est de promouvoir l'économie par le développement de technologies, la métrologie et des standards.
National Vulnerability Database	NVD	Base de données regroupant les vulnérabilités connues sous forme de référence CVE.
Lightweight	/	Qui est léger et peu demandant en ressources.

1 Introduction

Dans le cadre de ma dernière année de bachelier en sécurité des systèmes à la Haute École Namur-Liège-Luxembourg, j'ai réalisé un stage d'intégration en entreprise d'une durée de quinze semaines.

Ce stage se révèle être une opportunité pour parfaire mes connaissances, tout en fournissant un environnement de travail concret où les mettre en application. De plus, il permet, en cette fin de cursus, de constituer un premier pas dans le milieu professionnel.

Après plusieurs recherches infructueuses, j'ai eu contact avec le centre de recherche et de développement FoRS, affilié à l'Henallux, qui m'a proposé un stage dans le milieu du vulnerability assessment et du pentesting. Sujet que j'affectionne particulièrement notamment grâce à ma formation et à mes centres d'intérêt. Il m'a par la même occasion permis de travailler dans un environnement qui fait primer des valeurs comme la créativité et l'innovation afin de répondre à des besoins communautaires précis.

Au travers de ce document, les différentes étapes qui ont mené à la réalisation de mon projet sont détaillées.

1.1 Présentation de l'entreprise

Le centre FoRS¹ (Formations continues, Recherches appliquées et Services à la société), fondé en 2012, fait partie intégrante de l'Henallux et se répartit sur différents départements.

Ayant connu une forte croissance durant ces cinq dernières années, le centre FoRS représente environ 25 équivalents temps plein (ETP) qui travaillent dans la formation continue et la recherche appliquée et ce, dans différents domaines dont l'informatique, l'ingénierie, le paramédical, le social et la pédagogie.

Mon stage se déroule dans l'un des départements de ce centre, le Département Technique de Marche-en-Famenne (DTM), dirigé actuellement par le Docteur Julien Lecointre. Ce département comprend le centre de recherche ainsi qu'un Master en Architecture des Systèmes informatiques (MASI).

Leurs objectifs se résument en trois axes principaux :

- La proposition de services avec la collaboration d'entreprises locales et internationales
- La promotion de la recherche appliquée et la réalisation d'études déterminant la faisabilité théorique et pratique de projets
- L'accompagnement de différentes entités lors du processus de réalisation d'un projet

¹ <https://www.fors-ing-henallux.be/>

La portée de leurs recherches est multiple :

- En local avec la ville de Marche
- En Belgique avec notamment la collaboration d'universités
- En Europe à travers des projets intégrés
- À l'international avec notamment le continent américain



Figure 1 : Localisation du DTM – MASI

2 Présentation du projet

Le projet qui m'a été confié par Alexandre Marchal, chercheur au DTM et membre du Fors, est la conception d'une solution visant à détecter et à évaluer les faiblesses d'un ou plusieurs systèmes informatiques, principalement pour les petites et moyennes entreprises.

En effet, la quasi-totalité des articles s'accorde à dire qu'environ 43% des attaques informatiques subies en 2019 ciblaient ces dernières. Chiffres qui avaient été annoncés par Symantec en 2015 et qui semblent se poursuivre depuis. Il est aussi reporté que ces attaques portent leurs fruits dus au manque de sensibilisation et de connaissance relative à la sécurité informatique.

En réalité, la majeure partie des PME ne se sentent pas concernées par les attaques et ne perçoivent pas la cybersécurité comme un enjeu prioritaire. Le manque de mesures relatives à cette sécurité n'est dès lors que trop peu pris en considération malgré le gouffre financier qu'il peut impliquer. De plus, le taux de cyberattaques a explosé ces dernières années, les statistiques parlent d'une augmentation de plusieurs centaines de pour cent. En effet, la tendance actuelle porte davantage sur les attaques de type phishing, technique qui consiste à tromper l'utilisateur pour qu'il exécute des actions non désirées, et pouvant mener à des attaques de plus grandes envergures.

Afin de concevoir ces attaques, des informations propres à l'entreprise et à ses employés sont nécessaires. Par exemple, sur base des noms, prénoms et emails, des attaquants peuvent générer des dictionnaires d'identifiants qui sont ultérieurement utilisés lors d'une attaque ciblée. Ces informations sont dans la plupart des cas récoltées dues à la verbosité des équipements informatiques, ainsi qu'à l'utilisation excessive des réseaux sociaux. Il est possible d'imaginer un scénario où, sur base de ces mêmes informations, un attaquant décide d'entreprendre une action directement au sein de l'entreprise muni d'un faux badge ou affirmant avoir un rendez-vous urgent afin de gagner l'accès au réseau.

2.1 Objectifs du projet

Le contexte fourni par M. Marchal ne constituait qu'une ligne directrice à suivre et il ne tenait qu'à moi d'user de créativité afin de concevoir une solution innovante. Les exigences formulées étaient que la solution soit la plus autonome possible, discrète sur base de son utilisation soit défensive, soit offensive, et qu'elle reflète de manière aisément interprétable les résultats récoltés.

L'évaluation des vulnérabilités n'est cependant pas chose facile devant la multitude et la diversité des systèmes utilisés de nos jours. La solution imaginée se doit d'être la plus évolutive possible tout en restant simple à ajuster et à modifier selon les tâches qui lui seront attribuées.

Mais pourquoi se limiter à un usage professionnel ? La solution par son faible cout et sa simplicité pourrait aussi être conçue et répliquée pour un usage personnel afin de fournir à tout un chacun un outil de détection.

Finalement, celle-ci pourra tout aussi bien permettre de se prémunir d'une attaque que d'en orchestrer une. L'évaluation des vulnérabilités et les tests d'intrusion, étant deux activités effectuées de manière complètement opposée et arborant pourtant des méthodologies similaires.

3 Concepts théoriques

Afin de poursuivre la lecture de ce document, quelques concepts théoriques sont importants à bien comprendre, notamment les diverses activités dans lesquelles doivent évoluer et venir se développer le projet ou encore des technologies sur lesquelles il repose.

3.1 Introduction au test d'intrusion

Les tests d'intrusion, plus communément appelés par leur équivalent anglophone "penetration test" ou encore par leur forme raccourcie Pentest, représentent une activité qui consiste en l'analyse et l'évaluation des actifs informatiques d'une entreprise.

Cette activité se réalise à travers une simulation d'attaque informatique qui tente de mettre en évidence les failles propres à un système d'information. Contrairement aux évaluations de vulnérabilité, le test d'intrusion est effectué le plus souvent manuellement, sous certaines contraintes et ne vise pas uniquement la détection de failles, mais aussi leur utilisation afin d'en déterminer les conséquences éventuelles.

Ces tests sont dès lors réalisés sur base d'un modèle qui peut être propre à l'équipe de consultance ou plus souvent conçu par une organisation. Il sera ensuite divisé en phases qui représentent chaque partie de la simulation.

Un des modèles de référence les plus connus est celui du EC-Council², le CEH³. Il en existe cependant beaucoup d'autres, bien plus modernes et précis que celui-ci quant à la réalisation des tâches. Néanmoins le CEH représente les cinq phases [Figure 2] primaires à travers lesquelles chaque pentest doit se dérouler afin de fournir des résultats convaincants.



Figure 2 : EC-Council CEH Méthodologie

Sont alors représentées à travers le schéma ci-dessus, les phases suivantes :

² <https://www.eccouncil.org/>

³ <https://www.verifyit.nl/wp/?p=175054>

- De reconnaissance : qui vise à récolter de manière passive ou active un maximum d'information à propos de la cible. Cette phase est probablement la plus importante. C'est en effet grâce à elle que les informations nécessaires à celles qui la succèdent seront récoltées.

Ces informations et données peuvent être récoltées de diverses façons, par exemple, via des sources publiques, lors de fuites d'informations qui résultent d'une attaque précédemment subie ou encore d'un employé véreux. Dans chacun de ces cas, tout ce qui peut être découvert représente un trésor pour le pentester.

- De scan : dont l'objectif est de concevoir un mapping complet de l'infrastructure de la cible, de ses réseaux, de ses systèmes, de ses services et de ses hôtes. C'est à travers cette cartographie de la cible et l'évaluation de tous les vecteurs d'attaque que l'objectif qu'est la compromission peut ainsi être atteint.
- De gain d'accès : une fois les deux phases précédentes terminées, phases qui consistent en la préparation et la planification, l'offensive peut réellement débuter. Toutes les informations et données qui auront alors été récoltées, tels les identifiants, mots de passe, failles des services, algorithme de cryptographie, etc. vont ici permettre l'exploitation des faiblesses de l'infrastructure.
- De maintien d'accès : il est courant lors d'une attaque réelle, qu'en plus d'exploiter les systèmes, les hackers y laissent une *backdoor* ou une persistance. Celles-ci désignent tout programme ou configuration qui peut fournir à l'attaquant un nouvel accès au réseau de l'entreprise. Il pourra ainsi revenir plus aisément dans le réseau ultérieurement ou exfiltrer des données pour une prochaine opération.
- De couverture : où l'objectif est la suppression des traces qui ont été laissées au cours des phases précédentes. Si des fichiers et exécutables ont été déposés sur les systèmes, hormis les *backdoor*, ils seront alors supprimés, tout comme les logs générés durant l'attaque. En effet, si la cible n'entre pas en connaissance de l'attaque, les futures opérations seront alors facilitées.

Dans les modèles plus récents, une sixième phase peut venir se greffer aux précédentes : la phase de reporting. Celle-ci se concentre sur l'écriture d'un rapport qui a pour but de mettre en évidence et de documenter les faiblesses rencontrées lors des tests. Ainsi, une fois le rapport analysé, l'entreprise peut prendre des mesures afin de corriger les failles ou de limiter leur impacte.

Trois facteurs sont alors primordiaux lors de ces tests. Ces facteurs sont :

- Le temps d'exécution : qui peut varier de quelques jours à plusieurs semaines, selon l'envergure de l'opération et la taille de l'entreprise. Dans des cas de *red teaming* plus spécifiques, la durée d'un test peut même s'étaler sur plusieurs mois.

- La portée : il est en effet extrêmement important de correctement définir le *scope* de l'opération afin de ne pas endommager l'infrastructure de la cible ou de ne pas ralentir ses activités. La disponibilité étant une des règles d'or de la sécurité informatique. De même, elle permet à l'entreprise d'interdire l'accès à des données confidentielles ou critiques.
- Le taux d'information : connu et fourni par l'entreprise avant le déroulement des scans. C'est ce dernier qui permet de définir quel type de test sera effectué.

Parmi ces différentes catégories de test, les plus connus se trouvent être :

- Le *Black box pentest* : caractérisé par le peu d'information connu de la part des *pentester*. C'est cette catégorie qui se rapproche le plus d'une attaque réelle.
- Le *White box pentest*, où la quasi-totalité des informations est partagée à l'équipe des testeurs. Ici, l'objectif est de tester l'ensemble des ressources de la cible.
- Le *Grey box pentest*, qui consiste en un hybride des deux catégories précédentes. Peu d'informations sont partagées entre les *pentesters* et la cible, cependant une communication entre les deux partis est établie durant toute la durée du test.

Une fois l'entreprise cible en connaissance de ces informations, les modalités sont définies et acceptées par les deux partis que sont l'entreprise cible et l'équipe de *pentester*.

Le test d'intrusion peut alors débuter.

3.2 Introduction au Vulnerability assessment

L'évaluation des vulnérabilités désigne le processus d'identification, d'analyse et de classement par criticité des failles et vulnérabilités d'une infrastructure informatique. L'objectif de celui-ci est d'amener l'entreprise à établir une meilleure posture de sécurité par la connaissance de ses actifs informatiques présentant des menaces et le risque qu'elles représentent.

Similairement aux tests d'intrusion, les évaluations de vulnérabilité ont comme but de mettre en évidence les faiblesses d'un environnement, bien que la procédure appliquée soit toute autre. En effet, là où lors d'un *pentest* est simulé une attaque, ici c'est un *listing* des technologies employées par l'entreprise qui va être établi. Cette liste de technologie matérielle et logicielle est ensuite analysée plus en profondeur afin d'identifier les faiblesses de configuration, d'installation, ou propre à la version de chaque composant.

La force de ces systèmes de détection de vulnérabilité réside dans l'utilisation de systèmes d'évaluation de criticité. L'ensemble des CVE est alors référencé et chacune présente un profil précis basé sur des critères objectifs et mesurables. Chaque vulnérabilité est alors caractérisée par un identifiant, une description, des attributs quant à sa complexité et sa criticité ainsi qu'une foulée d'autres métriques. Ce système d'évaluation se nomme CVSS⁴.

⁴ <https://www.first.org/cvss/>

Ce dernier se base sur trois métriques principales que sont :

- La métrique de base : qui se base sur les attributs propres à la vulnérabilité indépendamment des systèmes sur lesquels elle est repérée.
- La métrique temporelle : qui définit l'impact au cours du temps.
- La métrique environnementale : qui, contrairement à la première, est fortement liée et influencée par l'environnement qui entoure la CVE.

Bien que les métriques environnementale et temporelle soient assez compliquées à caractériser dû à leur forte dépendance avec l'environnement qui les entoure, la métrique de base en revanche est caractérisée par des critères purement statiques décrits ci-dessous.

3.2.1 La métrique de base

Cette métrique est alors caractérisée par deux types de sous métrique que sont la métrique d'exploitabilité et celle d'impact.

Les trois critères représentés dans le tableau ci-dessous constituent alors la métrique d'exploitabilité et permettent de définir la complexité de réalisation d'une vulnérabilité.

Métrique	Description
Complexité d'accès	Comme son nom l'indique, il définit la complexité de réalisation de la CVE. Trois niveaux existent : haut, moyen ou bas, dépendamment de conditions spécifiques requises.
Authentification	Caractérisé par le nombre d'authentifications requises sur le système afin de réaliser l'exploitation : multiple, simple ou inexistant.
Vecteur d'accès	Il définit si la CVE peut être exploitée soit directement sur le système, soit via le réseau interne ou à distance.

La seconde est la métrique d'impact et permet de définir l'impact qu'a la vulnérabilité sur la triade de la sécurité.

Métrique	Description
Disponibilité	Permet de déterminer si l'impact d'une CVE sur la disponibilité sera nul, partiel ou complet.
Confidentialité	Permet de déterminer si l'impact d'une CVE sur la confidentialité sera nul, partiel ou complet.
Intégrité	Permet de déterminer si l'impact d'une CVE sur l'intégrité sera nul, partiel ou complet.

Bien évidemment, les systèmes de détection de vulnérabilité ne se limitent pas à ce *scoring*. Il est nécessaire afin de fournir un résultat de qualité que chaque information récoltée lors de cette procédure d'évaluation soit mise en évidence. Ainsi, la cible pourra prendre des mesures quant à la mitigation des fuites d'informations qui peuvent subvenir lors d'une attaque réelle.

Ces informations sont notamment constituées de :

- Données sur l'entreprise et ses employés
- Identifiants et mots de passe

- Clés de chiffrement
- Cartographies de réseau
- Bannières propres aux différents services
- Politiques de mots de passe
- etc.

3.3 Introduction au *scripting* et à l'automatisation

En programmation, un script désigne une suite de commandes ou d'actions prédéfinies afin de former une procédure. Le script est alors constitué d'une superposition de fonctions, chacune renvoyant un résultat qui permet ou non d'accéder à la prochaine étape. L'un des intérêts majeurs de ces langages est qu'ils sont interprétés et non compilés et cela implique une facilité à être adaptés, optimisés et corrigés.

Dans le milieu de la cybersécurité, les langages de *scripting* listés ci-dessous sont extrêmement courants :

- Python
- Shell
- Batch
- PowerShell
- Perl
- Ruby
- PHP

Les systèmes d'exploitation basés sur la sécurité informatique comme Kali Linux, Arch ou Parrot en sont d'ailleurs presque entièrement composés.

De plus, les plateformes d'hébergement de code source comme GitHub regorgent de projets écrits dans ces langages de par leur simplicité à être partagés entre les utilisateurs. Cet ensemble de langages laisse alors la possibilité d'imaginer autant de procédures que possible afin d'automatiser les tâches désirées.

Par exemple un script pourrait exécuter un premier programme capable de récupérer des mots de passe chiffrés avant de lancer un deuxième afin de détecter le chiffrement. Et enfin, un troisième qui pourra ensuite les déchiffrer.

En allant plus loin, un script parent pourrait utiliser ces mots de passe afin d'exécuter une autre suite de commandes afin d'essayer les mots de passe découverts sur un système.

3.4 Introduction au nano-ordinateur

Les ordinateurs sont depuis quelques décennies au cœur de notre société. Il faut cependant rappeler qu'à leur début, ils étaient imposants, très couteux, peu performants et certainement pas à la portée de tout un chacun. Leur taille pouvait atteindre celle d'une cuisine avec une consommation qui n'était certainement pas un atout. Cependant, avec l'évolution des technologies, leur taille a pu être fortement réduite pour donner forme à ceux qui sont connus aujourd'hui.

Parmi cette nouvelle génération de systèmes, on retrouve les micro-ordinateurs, mais aussi les nano-ordinateurs. La taille de ces derniers est d'à peu près dix centimètres et leurs performances ne cessent de croître ces dernières années. Ils se retrouvent alors au centre de nombreux projets de multiples domaines par leur utilisation quasi illimitée. Leurs utilisations les plus courantes concernent la domotique, la vidéosurveillance ainsi que l'apprentissage de la programmation.

Néanmoins, étant donné la liberté qu'ils offrent, ils se retrouvent désormais au centre de projets plus insolites et plus inventifs les uns que les autres.

Ceux-ci peuvent notamment servir de chaîne Hi-Fi, de centre de stockage, de station jeux vidéo, ou plus simplement comme ordinateur de bureau.

Une utilisation dans le milieu de la sécurité informatique particulièrement intéressante est de combiner un de ces nano-ordinateurs à une antenne capable de capter les différents points d'accès wifi à plusieurs centaines de mètres à la ronde afin de les attaquer. Ensuite accroché sous un drone, il peut ainsi permettre de couvrir rapidement et efficacement la surface d'un village ou d'une ville.

Dans le cadre de ce projet, ces ordinateurs miniaturisés seront d'une grande aide, car ils offrent un support mobile, discret et performant.



Figure 3 : Arduino & Raspberry pi

4 État de l'art des logiciels existants

4.1 Nessus

Nessus est ce qu'on appelle communément un système d'évaluation des vulnérabilités. Développé depuis une vingtaine d'années par Tenable Network Security⁵, il s'inscrit aujourd'hui parmi les meilleurs outils dans ce milieu. Le but est simple, mettre en évidence les faiblesses encourues par le système.

Parmi ces faiblesses, on retrouve :

- Des configurations obsolètes ou erronées qui peuvent causer des fuites d'informations.
- Des services vulnérables et exploitables.
- Des paires d'identifiants par défaut ou recensées dans les dictionnaires couramment utilisés.

Afin de repérer ces vulnérabilités, Nessus comptabilise une base de données relatant pas moins de 55000 CVE qui sont testés.

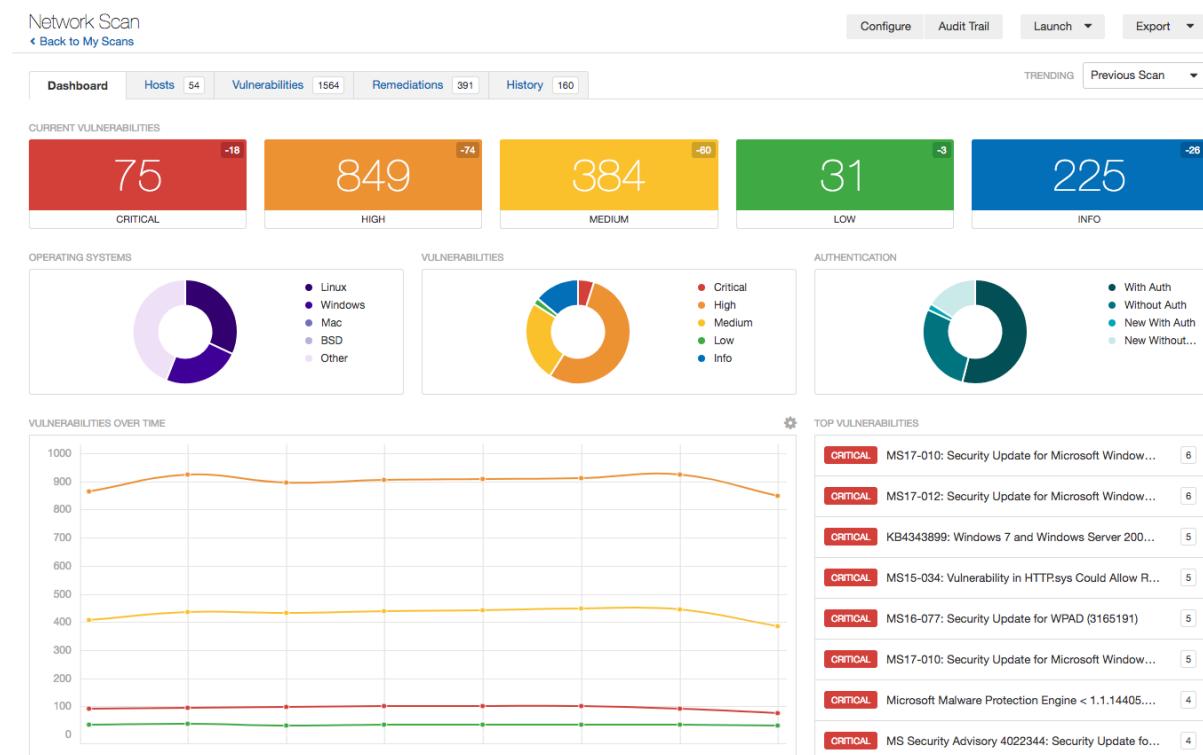


Figure 4 : Interface de Nessus

⁵ <https://fr.tenable.com/>

4.1.1 Méthodologie

Brièvement détaillé dans le document Tenable Scan Strategy⁶ la méthodologie exécutée par Nessus permet de mieux comprendre le fonctionnement et le raisonnement derrière le programme.

Cette méthodologie est d'ailleurs semblable à la majorité des programmes de vulnerability assessment et présente bien des points communs avec celles utilisées dans le camp opposé, le pentesting.

Ainsi, sont mises en évidence les étapes suivantes :

- *Host discovery*
- *Port scan*
- *Service detection*
- *Os detection*
- *Vulnerability analysis*

Ces dernières visent à récolter un certain type d'information et à concevoir un mapping du réseau permettant à terme de détecter si oui ou non un service est vulnérable à une CVE donnée.

Mais ce que nous apprend réellement le document, cité ci-dessus, ce sont les fonctionnalités plus spécifiques à Nessus comme le nombre d'hôtes pouvant être scanné en simultané ou encore les nombreux paramètres qui permettent au programme de s'adapter à la situation. Paramètres qui offrent par exemple, la possibilité d'effectuer plusieurs fois le même *check* sur un hôte afin de s'assurer de l'information reçue ainsi que d'instaurer des timeouts afin d'améliorer la rapidité du scan.

Des fonctionnalités très importantes dans ce genre d'application qui, dépendamment du réseau à scanner et du nombre d'hôtes découvert, peut rapidement prendre plusieurs dizaines de minutes. Ainsi la répartition des tâches et la gestion du travail en simultané sont à réellement prendre en considération lors d'un *vulnerability assessment*.

De plus, dans le but d'améliorer ses performances, Nessus permet à l'utilisateur une exécution centralisée à l'aide d'agent. Contrairement au mode standard où c'est le programme installé sur un serveur qui effectue les scans des différents hôtes, l'utilisation d'agents installés sur les hôtes à scanner permet de mieux répartir la charge de travail à effectuer tout en gardant une centralisation des données.

Dès lors, chaque hôte disposant d'un agent pourra être analysé individuellement, de manière plus spécifique selon le type d'équipement dont il s'agit et à intervalle défini.

⁶ https://docs.tenable.com/other/nessus/Tenable_ProServ_Scan_Strategy_Guide.pdf

4.2 Metasploit Framework

Développé par Rapid7⁷ et par la communauté open source d'enthousiastes en termes de sécurité informatique, Metasploit est probablement l'outil le plus connu et utilisé dans le milieu du pentesting. Il s'agit ici, d'un *framework* d'exploitation de vulnérabilité regroupant un nombre incalculable de modules visant à prendre l'ascendant sur une large gamme de services et de systèmes d'exploitation. Son rôle est donc d'effectuer des simulations d'attaques réelles pour ainsi vérifier la fiabilité des systèmes et fournir aux IDS des informations relatives à ces dernières.

Malgré la similitude dans leurs objectifs, là où Nessus se présente comme un système d'évaluation de vulnérabilité, Metasploit est bel et bien un outil de pentesting. Contrairement à ce dernier qui effectue un scan massif et automatisé, Metasploit est plus maniable et permet de vérifier la criticité d'un *exploit* sur un système donné de manière plus spécifique de par l'ensemble de modules et de fonctionnalités qu'il propose.

Il existe donc plusieurs versions de ce *framework*, présentant chacune des spécificités. La version gratuite propose l'utilisation de plus d'un millier d'*exploit* et l'import-export de code, ce qui représente un atout non négligeable.

Cependant ceci ne présente qu'une infime partie de la réelle puissance du *framework*. En effet, la version pro offre des fonctionnalités telles que des découvertes réseau, des modules d'exploitation avancés, la génération de rapports, l'automatisation de tâches ou encore des solutions permettant d'échapper à la détection des antivirus. Cet ensemble de modules et de fonctionnalités en fait sans aucun doute un des outils les plus puissants concernant le monde du *pentesting* et se voit être un incontournable.

Un incontournable qui, comme bien d'autres, n'est pas toujours utilisé dans un but louable, mais aussi lors d'attaques réelles.

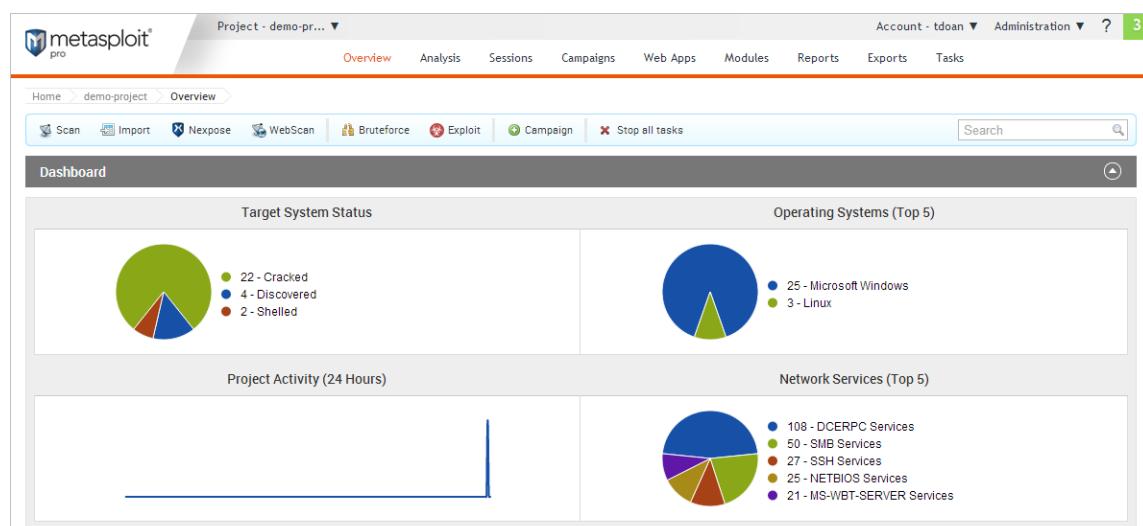


Figure 5 : Interface web de Metasploit Framework

⁷ <https://www.rapid7.com/#>

4.2.1 Portée des modules

Lors d'une attaque, il est important dans un premier temps de pouvoir récolter des informations à propos de la cible et de connaître les vecteurs d'attaques. Pour cela le *framework* propose des outils tels que des *sniffer* afin d'écouter le trafic ou encore des scanners qui visent à récolter les bannières et autres informations pouvant fuiter des systèmes.

Ensuite le *framework* couvre principalement les phases de pré-exploitation, exploitation et post-exploitation grâce à la multitude d'*exploit* conçu par Rapid7 et par la communauté qui atteint un haut niveau de précision. Mais aussi grâce à toutes les autres fonctionnalités qui englobent ces derniers et qui offrent la possibilité d'enumerer complètement un réseau, de définir des actions chainées qui seront exécutées de manière intelligente ou encore de créer une persistance sur le système cible.

Finalement, l'interface web de Metasploit permet aussi la génération de rapport ainsi qu'une clarté au niveau des informations reçues et l'affichage de statistiques quant aux scans précédemment effectués à travers divers graphiques. Malgré qu'il soit trop souvent associé au *script kiddies*, il n'est pas moins un des outils les mieux conçus et les plus fiables quand il s'agit de pentesting.

4.3 Conclusion

Cette analyse des solutions déjà existantes a permis d'apporter des éléments de réponses quant aux fonctionnalités et à la méthodologie qui doivent être adoptées dans ce projet.

L'analyse de Nessus a permis de mettre en évidence les étapes maîtresses à effectuer lors d'un scan, mais aussi l'intérêt de réfléchir à une architecture qui offre une gestion du travail en simultané ainsi que l'utilisation d'agents afin d'étendre la portée des scans.

Quant à l'analyse du *framework* Metasploit, elle met l'accent sur l'intérêt de regrouper un grand nombre d'outils et de les intégrer au programme principal sous forme de modules. Ces modules peuvent ensuite couvrir plusieurs types de scans et de tests afin de vérifier l'état des hôtes scannés.

C'est cette association des méthodologies et des fonctionnalités de ces deux programmes de *vulnerability assessment* et de pentesting qui va, dans la suite de ce document, permettre de concevoir une solution hybride de scanning.

5 Analyse

Dans ce chapitre est développée l'analyse du projet à réaliser ainsi que ses deux objectifs.

Le premier est la création d'un cahier des charges, une liste non exhaustive des fonctionnalités propres au projet. Cette liste énumère les composants nécessaires ainsi que les fonctionnalités de chacun. Parmi ces composants se trouvent tout d'abord le nano-ordinateur, suivi par le processus d'automatisation et enfin le *dashboard* dont les rôles respectifs vous seront présentés dans la suite de cette section.

Le second est de définir les technologies aussi bien logicielles que matérielles sont utilisées tout au long de ce projet et de son développement. Bien que certaines soient nécessaires, d'autre en revanche sont complémentaires elles offrent au projet la possibilité d'être porté à plus grande échelle ou encore de faciliter son utilisation. Parmi ces technologies se trouvent les divers services à installer, les langages de programmation utilisés, les services d'accès à distance et le matériel pour la gestion de la connectivité.

Le schéma ci-dessous [figure 5] permet de visualiser l'ensemble des parties du projet et leur agencement les uns vis-à-vis des autres.

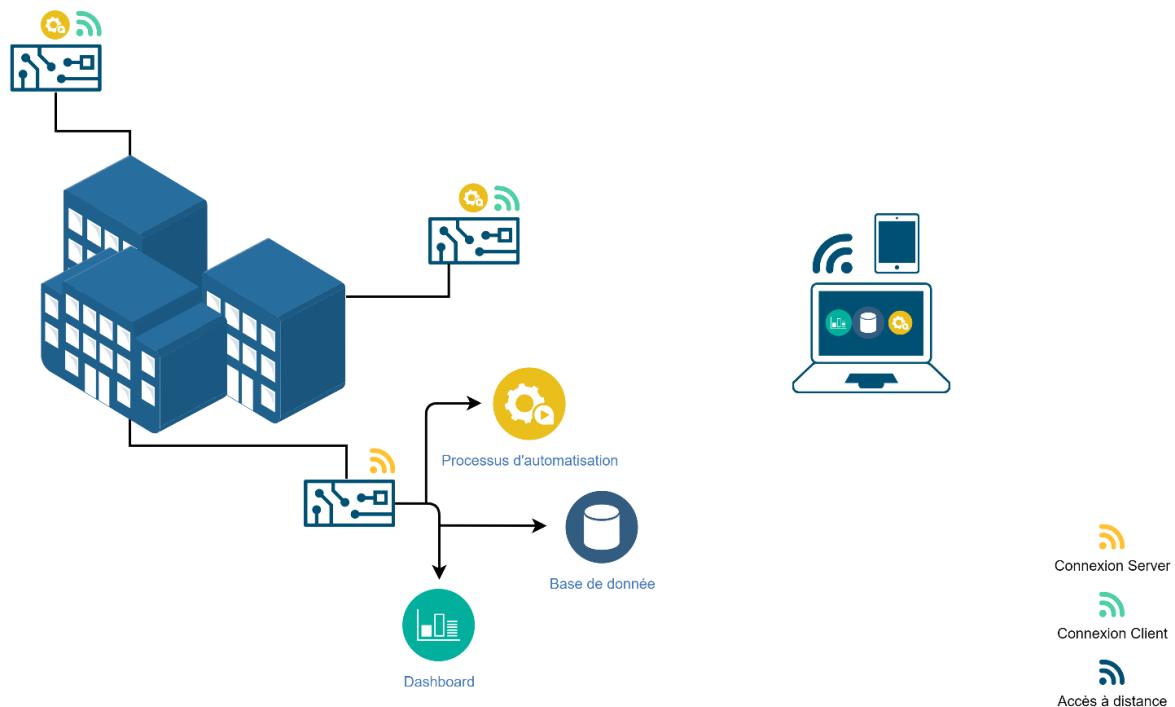


Figure 6 : Architecture du projet

5.1 Partie nano-ordinateur

Dans le cadre de ce projet, l'objectif principal du nano-ordinateur est donc de proposer un support performant, peu couteux et mobile. Sa taille étant légèrement supérieure à celle d'une carte de crédit, en fait un outil facilement déployable qui, couplé à un peu d'imagination, peut aussi s'avérer d'une grande discréetion.

L'une des fonctionnalités majeures de ce projet est d'effectuer toute la procédure de scan par une ou plusieurs machines. En effet, les tests d'intrusion et les évaluations de vulnérabilités présentées précédemment nécessitent souvent beaucoup de ressources. L'idée est donc de pouvoir, grâce à l'utilisation d'une base de données, répartir les différentes tâches entre les machines qui effectueront les scans.

Le nano ordinateur serveur, dans le cas présent, est donc celui qui hébergera la base de données, le service web ainsi que le *hotspot*. C'est d'ailleurs à travers ce réseau wifi hébergé par le serveur que les communications vers la base de données seront effectuées.

Quant aux clients, ce seront tous les autres appareils dotés du script de scan et connectés à ce dernier pouvant aussi être désigné comme des agents. Lors de ce projet, ces appareils seront majoritairement utilisés, cependant tout système quel qu'il soit, et configuré de façon similaire, peut venir soutenir la procédure de scan. De même en ce qui concerne la base de données. Bien qu'elle soit par défaut installée sur le serveur, elle peut tout aussi bien être configurée sur un serveur externe.

Grâce à ce modèle aussi bien indépendant que centralisé, la solution proposée dans ce projet permet à l'utilisateur de lui-même définir ses besoins en termes de puissance cumulée et de portée. Lors d'un test de vulnérabilité, chaque réseau à scanner pourrait être doté de plusieurs appareils embarqués afin d'accélérer la procédure. Là où lors d'une simulation d'attaque dans un réseau interne, seulement un ou deux appareils dissimulés peuvent permettre à l'attaquant de scanner le réseau sans trop attirer l'attention.

Fonctionnalités	Détails
Héberger les services	La fonctionnalité principale du nano-ordinateur est d'héberger les services nécessaires à la réalisation du projet. Ces services comprennent la base de données, le site web et les accès à distances.
Connectivité à distance	La solution se doit d'être mobile, pour cela des accès à distance sont configurés afin de prendre la main sur le nano-ordinateur sur une courte distance. D'où l'intérêt de la puce wifi incorporée.
Modèle indépendant	L'idée majeure de ce stage est l'automatisation, l'utilisation du nano-ordinateur se doit par conséquent d'être la plus autonome possible quant aux actions qu'il doit effectuer.

Modèle Client – Server	Malgré son autonomie, un modèle client – serveur permet au nano-ordinateur de répartir sa charge de travail grâce à une base de données centralisée. Il en résulte une amélioration de sa portée lors d'un déploiement et garantit un temps d'exécution réduit.
-------------------------------	---

5.1.1 Raspberry

Le Raspberry pi (RPI) est un nano-ordinateur commercialisé depuis 2012 coutant quelques dizaines d'euros. Sa première version était destinée à l'apprentissage de la programmation, mais l'augmentation croissante de ses performances, notamment avec sa troisième version, ouvre les portes à de nouvelles applications.

Son utilisation la plus connue dans le milieu du piratage est de le combiner à une antenne, dont la puissance est variable, dans le but de capter les réseaux aux alentours avant d'effectuer des actions sur ceux-ci.

C'est la quatrième version du RPI qui est ici utilisée. En termes de performance elle propose un processeur *Quad-core* 64-bit cadencé à une fréquence de 1.5 GHz et 4 Go de RAM en LPDDR4. Il comporte aussi, comme les anciens modèles, une solution wifi en 2.4 GHz et 5 GHz et une série d'autres connectiques. Il intègre ici un port Ethernet, plusieurs ports micro HDMI, un port jack et des ports USB 3.0. Ce qui permet au nano-ordinateur de parfaitement exécuter les actions qui lui sont définies.

5.1.2 Choix du système d'exploitation

Afin de garantir une utilisation *lightweight* correspondant aux ressources disponibles ainsi qu'aux outils utilisés lors de ce projet, le choix du système d'exploitation ne doit pas être négligé. Les systèmes d'exploitation propriétaire sont bien évidemment écartés, il reste cependant plusieurs OS basés sur le noyau Linux et axés sur la cybersécurité.

À cette enseigne on retrouve Parrot et Kali Linux basés sur Debian, mais aussi ceux basés sur Arch Linux respectivement Black Arch et Arch Strike. Cependant, parmi les quatre cités, seul Kali Linux est porté sur l'architecture ARM64, celle de notre RPI et est donc le seul choix disponible actuellement.

Il est tout de même à noter que pour ce type de projet les systèmes d'exploitation basés sur Arch ne sont pas la meilleure option due à leur système de mise à jour. La rotation des versions sur Arch étant très fréquente, elle pourrait entraîner des incompatibilités et des instabilités entre les modules et les outils nécessaires. Il reste néanmoins un très bon OS, à ne pas négliger dans le cas d'une utilisation plus manuelle que celle présentée dans ce projet.

Les développeurs de Parrot ont, quant à eux, partagé leur souhait de porter et d'adapter leur système d'exploitation sur les architectures ARM lors de ses prochaines versions afin de pouvoir être utilisé sur les nano ordinateurs. De plus, en comparaison à Kali, Parrot présente bien des avantages comme une consommation de RAM divisée de moitié, la possibilité d'être utilisé sans GPU intégré et enfin une surcouche qui favorise l'anonymat de l'utilisateur. Une alternative qui, ici aussi, n'est pas à

négliger, car cet OS *lightweight* permettra d'offrir de bien meilleures performances sur des machines au *hardware* limité.

5.1.3 Accès à distance

L'un des atouts principaux de ce projet est l'implémentation d'accès à distance. En effet, en plus de la procédure de scan automatisée, le RPI peut aussi servir de passerelle vers les réseaux où ils sont placés, chacun étant interconnecté par le réseau hébergé par le serveur. Ces derniers sont alors équipés des services SSH et VNC.

En premier lieu, SSH est un service, mais aussi un protocole qui offre la possibilité de partager des communications chiffrées entre deux machines. Il permet ainsi de prendre la main sur le RPI et d'effectuer des commandes en CLI. Pour se faire, un client est nécessaire. Bien évidemment n'importe quel terminal peut faire l'affaire, néanmoins pour ces tests c'est Terminus qui sera utilisé.

Terminus est un client SSH disponible sur toutes les plateformes entre autres Windows, Linux, Mac et sur mobile. Ce dernier contrairement à un terminal classique permet de mémoriser les sessions en retenant les informations comme l'IP, l'utilisateur, le mot de passe. De plus, une prise en charge des clés SSH est disponible ainsi que la création de groupe ce qui offre une meilleure gestion de l'environnement.

En deuxième lieu, et pour ne pas se limiter à une utilisation par command Line, c'est VNC qui sera implémenté. Ce dernier permet l'établissement de session sur le bureau à distance et donc avec interface graphique. Il est installé dans le cas où l'utilisation de SSH ne serait pas suffisante. Ici encore, un client est nécessaire est c'est VncViewer qui est utilisé. Tout comme Terminus pour SSH, celui-ci permet la prise en charge des sessions multiples soit via pc, soit sur mobile.

5.1.4 Émetteur / récepteur Wifi Moglor AC1200

Mis en évidence dans le point précédent, un aspect à prendre en compte dans ce projet est la volonté de faire primer la mobilité et la portée à laquelle les RPI pourront être atteints.

En effet, la puce wifi incorporée au RPI n'agit que sur une courte distance d'environ cinq à dix mètres et il est alors important de reconstruire la portée fournie par celle-ci. Dans un bâtiment de grande envergure tel qu'une école ou une entreprise, cette portée peut dès lors être améliorée grâce à l'utilisation d'antennes wifi.

Dans le cadre de ce projet, la portée d'émission nécessaire est d'une quarantaine de mètres, il existe cependant des antennes bien plus puissantes qui permettent d'augmenter la portée de plusieurs centaines de mètres.

Pour cela, l'utilisation d'une antenne wifi permet d'améliorer la portée de réception et l'émission du réseau hébergé par le serveur de plusieurs dizaines de mètres. Ainsi, un utilisateur placé à l'extérieur pourra sans problème garder la main sur chacun des clients. De même, des antennes pourraient aussi être installées sur les clients dans le but d'augmenter la distance séparant deux RPI.

5.2 Langages de programmation

La programmation est une activité indissociable du *Ethical hacking*. Pour cela différents langages de programmation existent et il est parfois compliqué de correctement choisir celui qui sera le mieux adapté à une tâche donnée. Chacun présente en effet un paradigme, autrement dit, une approche permettant de traiter un problème spécifique.

Parmi ces paradigmes, les plus connus sont la programmation de type orienté objet, impérative ou encore déclarative mais, en réalité, il en existe bien d'autres. En plus de définir le type de paradigme souhaité, prendre en compte les tendances permet d'aider quant à la sélection d'un bon langage de programmation.

En effet, les langages les plus populaires bénéficient régulièrement d'une grande implication de la part de la communauté, mettant ainsi à disposition nombre de tutoraux, d'exemples et d'explications quant au débugge des problèmes rencontrés. Communauté qui vient par la même occasion alimenter le langage de programmation en créant et partageant des modules et librairies en tout genre. Il est dès lors nécessaire de prendre en considération ces différents atouts.

Le langage de programmation est donc défini selon sa capacité à résoudre efficacement une tâche et non l'inverse.

Voici, ci-dessous [Figure 7 & Figure 8], une représentation des cotés de popularités des langages de programmation du site GitHub pour cette année 2020⁸.

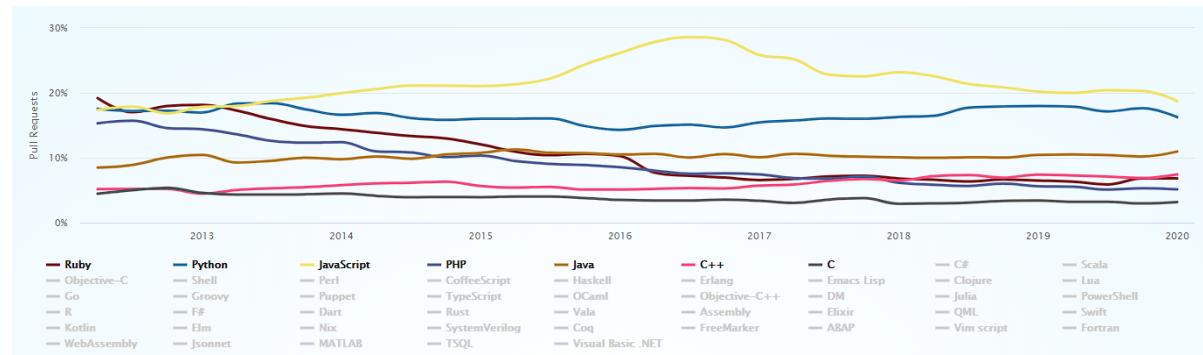


Figure 7 : Graphique des tendances en langage de programmation - GitHub 2020

⁸ https://madnight.github.io/github/#/pull_requests/2020/1

# Ranking	Programming Language	Percentage (Change)	Trend
1	JavaScript	18.703% (-1.406%)	
2	Python	16.238% (-1.654%)	
3	Java	10.938% (+0.538%)	
4	Go	9.005% (+0.978%)	
5	C++	7.423% (+0.040%)	
6	Ruby	6.812% (+0.342%)	
7	TypeScript	6.769% (+1.522%)	^
8	PHP	5.127% (-0.458%)	▼
9	C#	3.835% (+0.141%)	
10	C	3.181% (-0.203%)	
11	Scala	1.0470%	-

Figure 8 : Tableau des tendances GitHub 2020

5.2.1 Python

Conçu en 1991, python est un langage interprété multiparadigme de haut niveau axé sur la lisibilité, notamment au travers de sa syntaxe. Il est, depuis 2008 dans sa troisième version (désormais 3.7) et est, au fil des années, devenu un incontournable de la programmation tant par sa facilité à être abordé que par la multitude d'utilisations auquel il peut être rattaché.

Grâce à sa nature interprétée lui évitant d'être compilé, il permet dans le milieu de la cybersécurité d'être un réel atout pour l'automatisation de tâche. Une grande majorité des outils d'*hacking* disponibles sur le .NET et sur les systèmes d'exploitation axés sur ce domaine est d'ailleurs développée avec ce même langage.

De plus, grâce à sa communauté qui ne cesse de l'alimenter en module et en librairie, les possibilités qu'il offre ne cessent de croître.

Python est alors devenu un must hâve de la cybersécurité et probablement le langage le plus utilisé dans ce milieu.

5.2.2 JavaScript

En première position dans le tableau ci-dessus [Figure 8], JavaScript est actuellement le langage de programmation le plus populaire et le plus demandé, notamment grâce à ses fonctionnalités côté client.

De fait, il est aussi le meilleur langage de piratage d'applications web.

Il se trouve ainsi utilisé afin de concevoir les attaques de type XSS, dans la création d'*adware* ainsi que dans la manipulation de l'interface et des cookies ou encore des fonctionnalités *backend* et *frontend* d'un site web.

5.2.3 C & C++

Largement utilisé dans le milieu du développement et du *hacking*, l'un est un langage impératif structuré et l'autre est orienté objet. Le C et le C++ sont connus pour être des langages de bas niveau qui offrent ainsi une certaine proximité avec le système d'exploitation et le hardware.

Les Systèmes d'exploitation, comme Windows ou Unix, sont d'ailleurs conçus avec ce même langage (C).

De par cette nature *low-level*, ils sont tous deux largement utilisés dans le milieu de la cybersécurité pour leur affinité avec la gestion de la mémoire en RAM, la manipulation de processus et de DLL ainsi que dans l'écriture d'*exploit*, de *shell code* ou de virus.

De nombreux outils présents en entreprise sont d'ailleurs conçus dans ces langages, ce qui en fait de très bons outils de reverse engineering afin de contourner des sécurités systèmes.

5.2.4 PHP

PHP est un des langages de programmation web les plus utilisés, il permet aux côtés du HTML, du CSS et du JavaScript, de rendre les sites web dynamiques. Il est d'ailleurs utilisé par le célèbre CMS WordPress qui compose presque 30% des sites web. Dans le milieu de la sécurité informatique, une bonne connaissance du PHP permet de tirer parti de son utilisation côté serveur.

En effet, des scripts présents sur un serveur distant peuvent être manipulés dans le but de créer un accès à ceux-ci.

De plus, sur un serveur mal sécurisé, l'utilisation du PHP peut mener à la corruption du système sur lequel le site web est hébergé grâce à l'utilisation de fonctions permettant d'exécuter des commandes directement sur le système ou encore d'ouvrir des sessions vers une machine distante.

Un langage donc très présent dans le milieu du *hacking* web.

5.2.5 SQL

Désigné comme un langage déclaratif, SQL ne peut pas vraiment être qualifié de langage de programmation. Il est en revanche conçu pour interroger et communiquer avec une base de données, plus spécifiquement avec les SGBDR.

Grâce à celui-ci, une seule commande offre la possibilité de traiter tout un lot de données. Ainsi, utilisé au travers d'autres langages tels que python, C ou encore PHP, il peut aider à la gestion des données afin de concevoir des programmes ou des sites web dynamiques.

Il est cependant tristement connu pour les injections SQL, détournement du langage et de ses fonctions qui permettent d'exfiltrer des données, de contourner des systèmes d'authentification ou encore de servir de vecteur d'attaque à un serveur.

5.2.6 Conclusion

La rapide analyse de ces langages permet de définir dans quel contexte ils sont utilisés pour le développement et plus particulièrement vis-à-vis de la cybersécurité, mais aussi de mieux comprendre les avantages de chacun.

Ainsi, parmi les six présentés dans les points précédents, seuls le C et le C++ ne sont pas utilisés au cours de ce projet.

Le Python est utilisé dans la création du client, présenté au point 5.3 «Processus d'automatisation».

Les langages PHP et JavaScript quant à eux sont responsables de la partie front end et back end lors de la conception du site web, présenté au point 5.5 «Dashboard».

Finalement le SQL permettra la gestion et la manipulation de toutes les données relatives à ce projet. Il fera office de langage de communication entre les divers composants du projet, présenté au point 5.4 «Choix de la base de données».

Chacun aura donc son importance dans la conception de la solution détaillée dans ce document et sera utilisé pour les avantages qui ont été présentés au cours de leur analyse respective.

5.3 Processus d'automatisation

Le processus d'automatisation est au cœur de ce projet, il permet au module de s'exécuter en effectuant le moins d'interventions manuelles possible. Ce processus comporte l'exécution des scripts et l'utilisation des outils d'évaluation de vulnérabilité et d'énumération.

Afin d'effectuer cette procédure et d'héberger les outils, le système utilise une version du système d'exploitation Kali Linux conçu pour les systèmes embarqués. Kali est un système d'exploitation basé sur Debian Linux bien connu dans le milieu du pentesting, car il permet d'obtenir une quantité d'outils et de librairies non négligeable relative aux phases du cycle de vie d'un test d'intrusion [Erreur ! Nous n'avons pas trouvé la source du renvoi. Figure 2].

En ce qui concerne plus directement les scripts d'automatisation, ils sont écrits en Python. Analysé dans le chapitre précédent, il est celui qui convient le mieux à la réalisation de ce projet de par les modules qu'il propose ainsi que sa compatibilité multiplateforme.

Fonctionnalités	Détails
Reconnaitre son environnement	La première étape du processus d'automatisation est de trouver les informations relatives à l'environnement où l'appareil embarqué est déployé. Parmi les recherches sont attendues les informations par rapport au réseau et aux machines qui y sont connectées.
Scanner les hôtes	La deuxième étape consiste à récolter des informations propres à chacun des hôtes, comme leur OS et sa version, leur MAC adresse ainsi que des informations supplémentaires à partir des services présents.

	Une fois les hôtes et leurs services connus, un autre type de scan est alors exécuté afin de définir les vulnérabilités présentes sur les systèmes.
Tester les services	La dernière étape du processus consiste en une batterie de tests sur les services repérés afin d'effectuer des actions plus agressives. Ces tests visent à extraire un maximum d'informations critiques grâce à l'utilisation d'outils. Ces dernières sont par exemple des identifiants, des emails ou encore des politiques de sécurité.

5.4 Choix de la base de données

Tout comme le choix du langage de programmation, celui de la base de données peut paraître compliqué au premier abord. Cependant, définir les besoins et les objectifs relatifs au projet auquel elle sera associée permet de définir correctement lequel des services disponibles sera le plus adéquat.

Il existe alors deux types distincts de bases de données, le premier basé sur l'utilisation du SQL, l'autre non, que l'on appelle alors NoSQL et qui commence à prendre de l'ampleur ces dernières années [Figure 10].

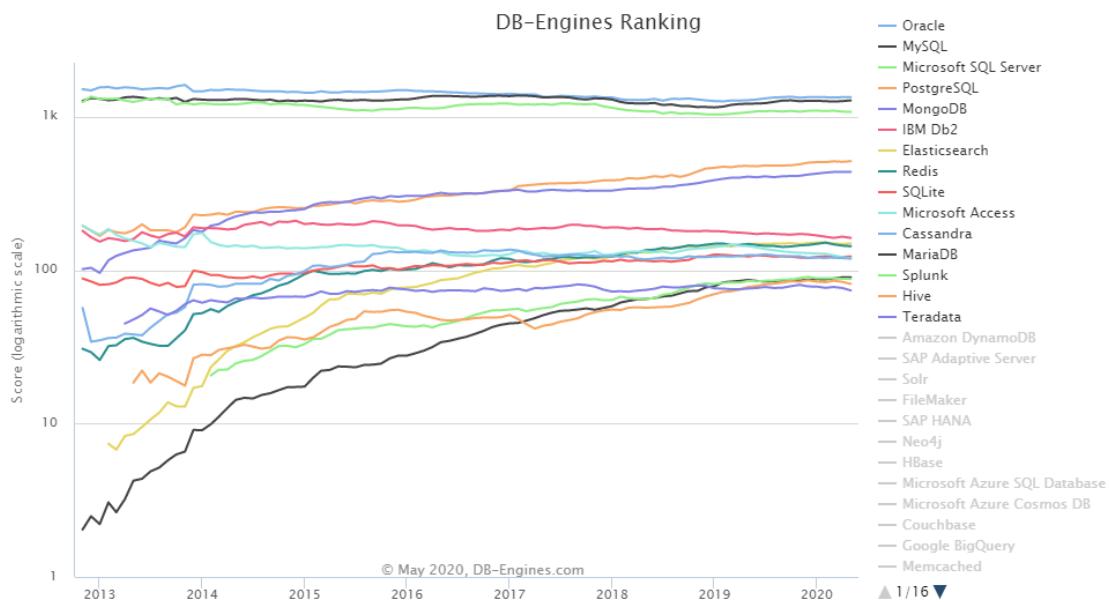


Figure 9 : Tendances - base de données 2020

SQL, comme expliqué précédemment, est directement lié aux bases de données relationnelles et permet la manipulation des données stockées par le biais de requêtes. Ce type de BD se base principalement sur l'organisation en table des données qui y sont stockées et sur les interactions et les liaisons entre ces différentes tables.

Les bases de données NoSQL se basent, contrairement à notre première, sur l'utilisation de données non structurées et dont l'organisation est dynamique. Ces données sont ensuite triées et structurées à la volée directement dans la BD. De plus, les types de données traités par ces BD ne sont plus majoritairement textuels et numériques, mais peuvent notamment comprendre des documents, graphiques, ou des objets de données comme le JSON ou le XML.

Au cours de ce projet, les données utilisées sont uniquement des données structurées et dont l'organisation a été pensée, c'est pourquoi l'utilisation d'une base de données en NoSQL est ici écartée. Cependant, leur utilisation trouve sens dans des projets axés sur le big data et le stockage de logs notamment dans les SIEM.

L'article de High Scalability⁹ sur les tendances en base de données montre d'ailleurs qu'en 2019, plus de 44% des entreprises avaient opté pour une utilisation multi-BD composée à plus de 75% d'une utilisation commune de SQL et de NoSQL. Chacun de ces deux types de bases de données ayant leurs avantages selon la situation.

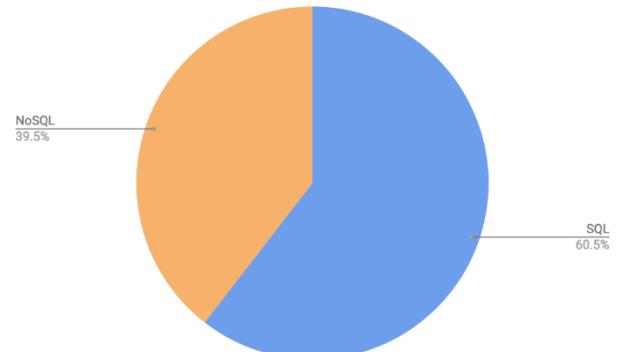


Figure 10 : Répartition NoSQL et SQL

⁹ <http://highscalability.com/blog/2019/3/6/2019-database-trends-sql-vs-nosql-top-databases-single-vs-mu.html>

5.4.1 Comparatif des SGBDR

Le type de base de données désormais défini, c'est maintenant le service de SGBDR qui va devoir être choisi afin de définir lequel sera le plus apte à héberger le projet.

Parmi ceux-ci on retrouve principalement les quatre mis en évidence sur le graphique ci-dessous [Figure 11].

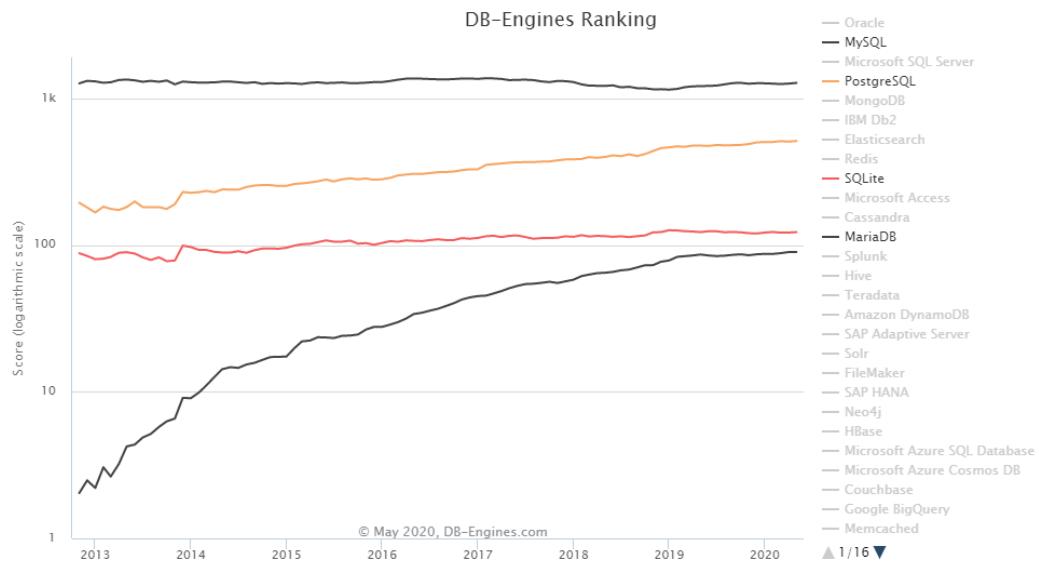


Figure 11 : Tendances SGBDR 2020

Les SGBDR comme Oracle, Microsoft SQL server et IBM Db2 sont des logiciels propriétaires, payants ou encore non multiplateformes et ne conviennent donc pas à ce projet. Les SGBDR restants sont MySQL, PostgreSQL, SQLite et MariaDB.

5.4.1.1 SQLite

SQLite est un SGBDR open source embarqué dont les principaux avantages sont sa nature *lightweight* et sa portabilité. Son utilisation repose sur l'utilisation d'un seul et même fichier contenant toutes les données. Contrairement aux bases de données traditionnelles, les SGBD embarqués sont directement interrogés en chargeant le fichier de BD. De par cette nature, SQLite présente alors plusieurs avantages ainsi que plusieurs inconvénients.

Parmi ses avantages, celui à mettre en évidence est sa simplicité à être implémentée. En effet, son utilisation ne reposant pas sur l'utilisation d'un service permet à l'utilisateur de l'implémenter sans devoir effectuer la moindre configuration. Cependant, ce fonctionnement implique aussi quelques gros désavantages, notamment en termes de sécurité, SQLite n'offre à l'utilisateur aucune possibilité de définir des permissions d'accès aux données et est plus sensible que les BD traditionnelles à la corruption de données.

5.4.1.2 MySQL & MariaDB

Comme représenté dans le graphique ci-dessus [Figure 11], MySQL est le SGBDR open source le plus populaire actuellement, sans compter MariaDB, un *fork* conçu par d'anciens développeurs de MySQL après son rachat par Oracle, auquel il peut être associé. Celui-ci est conçu et reconnu pour sa rapidité et sa fiabilité et est accompagné d'une foulée d'extensions et des librairies lui permettant de s'adapter plus que correctement à un large nombre de situations et de faciliter son utilisation.

Son approche de la sécurité est un avantage pour MySQL. En effet, le service vient directement accompagné d'un script qui prend en charge la configuration du service afin de restreindre au mieux les accès non autorisés à celui-ci. De plus, dû à sa popularité, le SGBDR est accompagné d'une documentation très élaborée qui offre à l'utilisateur un certain confort lors de la configuration du service. En revanche, suite aux différents rachats du projet, le développement de celui-ci a fortement été affecté ce qui n'a pas plu à la communauté. Il reste cependant et malgré ce ralentissement le SGBDR le plus populaire utilisé aujourd'hui.

5.4.1.3 PostgreSQL

PostgreSQL est un SGBDR open source conçu pour être hautement performant, fiable et connu pour sa flexibilité. Contrairement aux deux premiers présentés ci-dessus, celui-ci est orienté objet ce qui le qualifie de SGBDRO. Grâce à cette nature orientée objet, PostgreSQL propose alors des fonctionnalités comme l'héritage ou la surcharge de fonction ainsi que la création de nouveaux types de données et de nouvelles fonctions directement par l'utilisateur.

La force de ce SGBDR réside dans son extensibilité et dans sa capacité à charger dynamiquement ses fonctions, qu'elles soient fournies ou conçues par l'utilisateur. Il est de plus celui se conformant le mieux aux standards SQL parmi ceux analysés.

Cependant, cette nature extensible le rend aussi plus complexe à implémenter et plus gourmand en ressource que ses concurrents, il convient alors moins bien aux configurations plus basiques ou de petite envergure.

5.4.2 Choix du SGBDR

Après avoir analysé les tendances des services de base de données, c'est Maria DB qui a été choisie pour cette solution. Un des SGBD à la popularité croissante et un des quatre éléments de la *stack LAMP*¹⁰, elle-même très appréciée des développeurs. Un autre SGBD pourrait le remplacer selon les préférences, mais l'intérêt de celui-ci résulte dans sa facilité à être configuré et sa compatibilité avec les autres éléments de la *stack LAMP*, utilisés eux aussi dans ce projet. Il est de plus installé par défaut comme SGBD sur de nombreux systèmes Linux et ce dernier propose toutes les fonctionnalités nécessaires à la solution développée.

Afin de faciliter la gestion de la base de données, MySQL Workbench est combiné à la BD. En effet MariaDB étant un fork de MySQL ce qui permet leur utilisation commune. En termes de possibilités, le logiciel permet ici la création d'un schéma relationnel et donc des liaisons et des interactions entre les différentes tables de notre base de données. Une fois ce schéma créé, il peut être exporté sous

¹⁰ <https://www.ibm.com/cloud/learn/lamp-stack-explained>

forme de requêtes SQL qui peuvent alors être interprétées par le SGBDR afin de créer la base de données sur le serveur distant. Ces requêtes doivent, selon le service et la version utilisés être adaptés afin de convenir à la syntaxe instaurée par le SGBDR.

5.5 Dashboard

Le *dashboard* est le dernier des composants primaires et représente la partie visible de ce projet. Comme expliqué au point 2.1.1, le site web est hébergé sur le nano-ordinateur et offre un rendu clair et abordable aux utilisateurs de la plateforme.

Fonctionnalités	Détails
Offrir un résumé	Le <i>dashboard</i> doit présenter de manière dynamique les informations récoltées lors des précédents tests, de préférence sous forme de schémas et de graphiques. Il permet aussi à l'utilisateur d'explorer en détail tous les rapports générés durant les scans par les outils utilisés.

5.5.1 Serveur web

Deux autres éléments de la *stack LAMP* sont ici utilisés.

Le premier est Apache2 qui est le service qui héberge le *frontend* et donc la partie graphique de ce projet. Tout comme le SGBD, il est facilement configurable et se voit être un des services web les plus populaires.

Lors de la conception d'un site web, l'utilisation de langages comme le HTML, le CSS et le JavaScript est inévitable. Il existe cependant depuis quelques années des outils permettant une conception plus efficace de ceux-ci.

C'est par exemple le cas de WordPress, le CMS le plus connu au monde, qui a permis la création de plus d'un tiers des pages présentes sur le web grâce à sa simplicité d'utilisation. Néanmoins d'autres outils, dont l'objectif est de réduire l'utilisation brute du HTML et du CSS, existent et c'est le cas de Bootstrap.

Ce *framework* open source permet ainsi une création rapide et simplifiée de sites web grâce aux nombreuses librairies qu'il met à disposition. Parmi celles-ci se trouvent alors tous les éléments frontends de base d'un site web par exemple les boutons, formulaires, barre de navigation, menu et graphiques conçus en CSS et JavaScript. L'API Bootstrap offre de plus une documentation complète et de nombreux templates afin de simplifier au maximum l'utilisation de ce dernier.

Finalement l'utilisation du PHP viendra compléter la liste des langages utilisés en permettant la connexion et l'interaction avec la base de données.

6 Implémentation

6.1 Configuration de l'environnement

La première étape de ce projet consiste à rendre l'environnement de travail, ici le RPI, apte à héberger les outils et services nécessaires à la procédure de scan. À cette fin, c'est le système d'exploitation, les accès à distance ainsi que les services comme le web ou la base de données qui sont installés en premier lieu.

6.1.1 Installation de la Raspberry

Une des problématiques abordées précédemment et concernant le RPI était le choix du système d'exploitation. Après analyse des choix disponibles et selon les critères requis pour ce projet, c'est l'OS Kali qui a été désigné comme celui qui correspond le mieux aux attentes.

L'iso peut alors être téléchargé sur le site officiel, il en existe plus d'une dizaine disponible pour Kali, mais celui qui nous intéresse est celui conçu spécialement pour les architectures ARM et en 64 Bits. Un second programme sera aussi nécessaire afin de graver l'image de l'os sur un support. Il existe différents programmes à cet effet comme Rufus, Imgburn, Daemon Tools ou encore Etcher qui est celui utilisé dans ce cas, néanmoins chacun des autres est tout autant fonctionnel. En tant que support, une carte SD de 32Gb est fournie avec le package Labists¹¹ utilisé au cours de ce projet. Il est d'ailleurs conseillé de ne pas utiliser une carte de taille inférieure à celle-ci dépendamment de la charge d'outils à installer.

Une fois la gravure terminée, il suffit d'insérer la carte dans le RPI afin qu'il démarre automatiquement sur notre système d'exploitation fraîchement installé. La procédure d'installation de notre système d'exploitation Kali Linux est similaire à la version par défaut¹².

Une des premières mesures à prendre ensuite est la création d'un utilisateur non privilégié.

En effet, sur Kali il n'existe par défaut qu'un seul utilisateur étant celui d'administration de la machine et son utilisation permanente est fortement déconseillée. Afin de gérer au mieux les permissions liées à notre nouvel utilisateur, le service sudo peut être installé afin de n'autoriser qu'une gamme d'outils limité et nécessaire à ce projet.

De même les différents outils et services installés sur la machine doivent être reconfigurés pour n'être accessibles que sur base de ces permissions. L'utilisation de SSH étant par exemple autorisée en root par défaut sur Kali. Les configurations des services d'accès à distance doivent donc être revues afin d'éviter toute infiltration non autorisée sur la machine.

¹¹ <https://labists.com/>

¹² <https://www.kali.org/docs/installation/>

Le premier type d'accès utilisé au cours de ce projet est SSH.

En plus de la mesure à prendre quant à l'autorisation du login en root par défaut, une autre est de modifier le port par défaut afin de rendre la détection du service plus compliquée. Dans l'objectif d'augmenter une fois de plus la sécurité du service SSH, l'intégration d'une authentification par clé publique et l'installation de fail2ban permet d'éviter toute attaque de type brute force. Une fois le service SSH correctement configuré, il sera accessible par n'importe quel client SSH.

Le second type d'accès à distance est celui proposé par le service VNC qui propose la création de sessions vers bureau virtuel. Plusieurs services VNC sont disponibles, néanmoins des critères spécifiques sont requis dans la réalisation de ce projet.

Premièrement la capacité de rendre le service *headless*. En effet par défaut ce qui permet à VNC de partager le bureau réel et non virtuel, c'est sa connexion à un moniteur. Cependant, la solution imaginée ici doit être la plus indépendante possible et donc, ne pas dépendre de ce moniteur.

Deuxièmement, le service VNC choisi doit être disponible avec un client autant sur mobile que sur pc. Tout comme SSH, la connexion à courte portée est un des objectifs visés par cette solution. Parmi les services qui sont disponibles seuls, un répond à ces critères et c'est TightVNC, service qui est d'ailleurs installé par défaut sur Kali.

Un problème survient cependant dû au GUI par défaut de la distribution. Il ne permet pas un affichage correct du bureau, car incompatible. Pour cela un autre GUI, cette fois compatible, peut être installé tel que XFCE choisi ici pour son faible cout en performance.

Un autre des points à aborder est celui de l'accès au réseau.

L'exécution de la procédure de scan va engendrer un certain taux de communications sur le réseau et il est alors important de limiter au maximum ce trafic. Pour cela un point d'accès est créé sur la machine serveur afin d'établir une communication isolée entre tous les clients. Ce dernier peut être créé soit grâce à un outil appelable depuis le terminal, soit depuis l'interface graphique et permet de définir un réseau protégé visible ou non.

De plus, l'accès créé sera de type hotspot, car il permet de communiquer avec le réseau principal où le serveur est connecté.

Grâce à celui-ci, si un VPN est installé sur le RPI ou s'il est connecté à un serveur de C&C, il peut alors former une sorte de passerelle vers ce réseau. Cependant le gros problème auquel nous allons ici être confrontés est celui de la portée.

La dernière problématique concerne plus directement l'automatisation des scripts.

Le RPI étant désormais accessible par divers accès à distance et sans connexion à un moniteur, un auto-login peut être configuré afin d'accéder directement à l'environnement de notre utilisateur.

De plus, des tâches peuvent être exécutées au démarrage ou encore de manière répétitive grâce à l'utilisation du service cron ce qui permet de rendre le RPI opérationnel dès son alimentation.

Alimentation par prise qui peut d'ailleurs être évitée grâce à l'utilisation d'une batterie externe, ce qui améliore la mobilité et la discréetion de la solution.

6.2 Conception de la base de données

Cette partie abordera l'installation, la configuration et la conception de la base de données ainsi que les composants qui y sont liés. Les points abordés seront donc l'installation du service et sa configuration, la création d'utilisateurs et la gestion des permissions associées à ceux-ci, les accès et finalement la conception de la BD elle-même.

6.2.1 Installation du service de base de données et du SGBD

Une fois le système d'exploitation et ses sources mises à jour, l'installation des services mariadb-server et mariadb-client se fait par le biais du package manager sous Linux, ses deux services sont aussi disponibles sous Windows. Une fois ce dernier installé le binaire mariadb-secure-installation peut être exécuté dans le but de sécuriser rapidement et efficacement le service de base de données. Ce binaire est en réalité un lien symbolique vers mysql_secure_installation, mais a été renommé depuis les dernières versions de MariaDB.

```
user@Kali# mariadb-secure-installation
```

Ce binaire prend en charge différentes configurations plus que nécessaires face aux paramètres par défaut du service.

Il permet dans un premier temps de définir un mot de passe pour l'utilisateur root ainsi que la suppression de l'utilisateur anonyme qui offre une connexion sans même avoir d'utilisateur. Deux configurations donc de premier ordre permettant de réduire les accès non désirés à la base de données.

En deuxième lieu, la table de tests ainsi que les priviléges qui y sont associés sont supprimés, limitant une fois des plus le champ d'opérations non voulues.

Ensuite, les accès à distances pour l'utilisateur root sont désactivés, ne l'autorisant donc qu'en local. Utilisateur root qui n'est d'ailleurs accessible qu'avec son mot de passe fraîchement défini et si le service est lancé par un utilisateur privilégié ou par le biais de la commande sudo.

Finalement, la table des priviléges du SGBD est rechargée laissant la possibilité d'utiliser le service de manière bien plus sécurisée.

```
All done! If you've completed all of the above steps, your MariaDB installation should now be secure.
```

6.2.2 Chiffrement du trafic

L'implémentation d'un chiffrement permet tout d'abord d'éviter que le trafic ne soit intercepté et analysé durant l'exécution des scans ainsi que de valider l'identité des clients communiquant avec la BD.

Pour cela, Openssl, un outil de chiffrement et de déchiffrement disponible sous Linux, permet la création de trois certificats, l'un pour l'autorité de certification (AC), l'autre pour le serveur et le troisième pour les clients.

Afin de garantir l'intégrité, la confidentialité et l'authenticité des communications, le chiffrement doit alors respecter les recommandations actuelles. Les clés associées à ces certificats sont alors générées en RSA 2048 Bits. De plus, un délai d'expiration peut être défini afin de nécessiter le renouvellement du certificat client.

L'utilisation d'Openssl permet ainsi la création, la signature et la vérification des certificats et des clés qui y sont associés.

Une fois cette étape terminée, la base de données doit ensuite être configurée afin de pouvoir reconnaître et utiliser ces certificats et ainsi permettre le chiffrement de tout le trafic en TLS. Pour cela il suffit de spécifier le chemin de ces fichiers dans le fichier de configuration lié au SGBD utilisé.

Cette configuration peut être vérifiée grâce à la commande.

Afin de vérifier le chiffrement du trafic, un *sniffer* comme Wire Shark ou encore tcpdump sous Linux peut être utilisé.

6.2.3 Configurations supplémentaires et sécurité du service

Afin de mieux sécuriser les accès à la base de données, d'autres paramètres peuvent être modifiés dans le fichier de configuration propre à la BD.

Tout d'abord, un de ces paramètres offre la possibilité d'isoler les connexions à la base de données. Ainsi en utilisant l'option bind-address, définie par défaut en localhost, un réseau unique peut être ciblé afin de n'autoriser que le trafic provenant de celui-ci.

Ensuite, une option permet de désactiver le chargement des fichiers locaux. Celle-ci empêche alors l'utilisation de la fonction LOAD DATA INFILE qui pourrait être utilisée par un utilisateur mal intentionné dans le but de nuire à l'intégrité de la base de données.

Une autre permet ensuite de re diriger les logs générés par la base de données afin de les déplacer dans un dossier à l'abri des regards et uniquement visible par un utilisateur privilégié.

Finalement, depuis le SGBD, une commande permet de redéfinir le nom de l'administrateur afin de le rendre moins prévisible.

Ces différentes mesures sont mises en place afin de limiter et d'isoler au mieux les accès aux services et ainsi éviter toute fuite de données ainsi que la compromission de la BD.

6.2.4 Conception et structure

Une fois le service installé et configuré, la prochaine étape consiste à concevoir un modèle relationnel pour la future base de données.

Ce modèle relationnel a pour objectif mettre en évidence les relations existantes entre les différentes tables qui composent notre schéma.

Pour ce faire c'est MySQL Workbench, le logiciel de gestion et d'administration conçu pour les serveurs MySQL, qui est utilisé. Celui-ci permet entre autres d'effectuer l'entièreté des actions possibles de gestion sur la base de données ainsi que la gestion des rôles et utilisateurs. De plus, il permet de créer et de visualiser les bases de données liées au serveur.

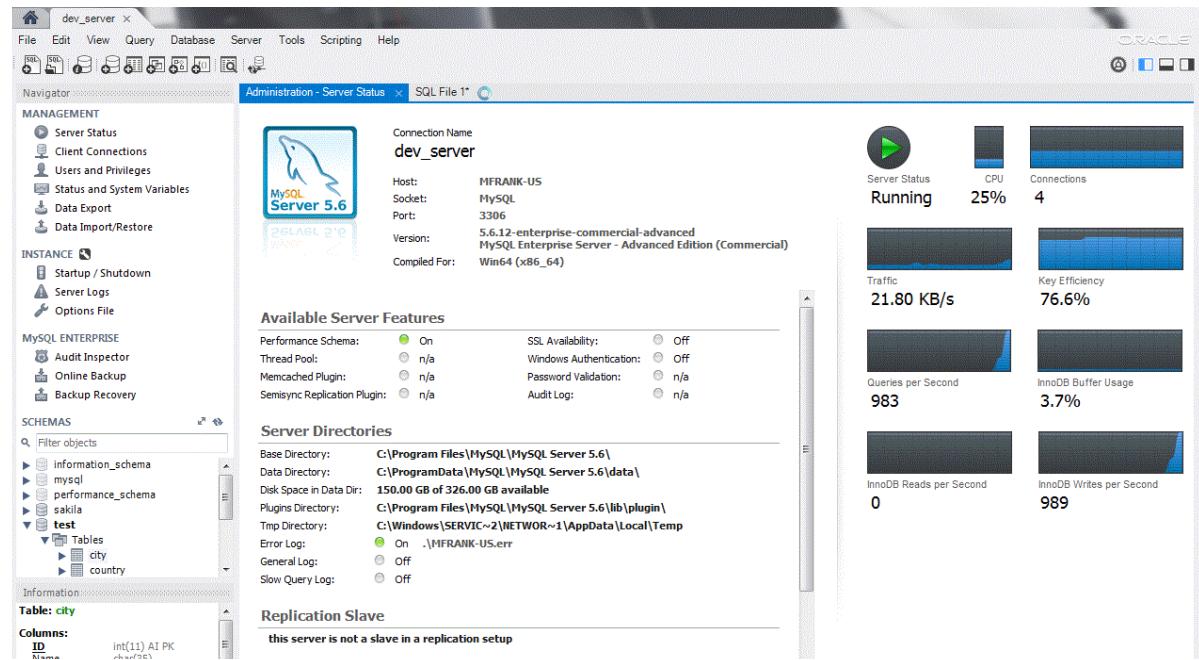


Figure 12 : Interface MySQL workbench

Le modèle conçu pour notre base de données peut être divisé en trois parties, chacune responsable d'un type d'informations spécifique retenue pendant le scan.

La première partie permet alors de mettre en évidence les liaisons entre les clients et les scans qui sont effectués.

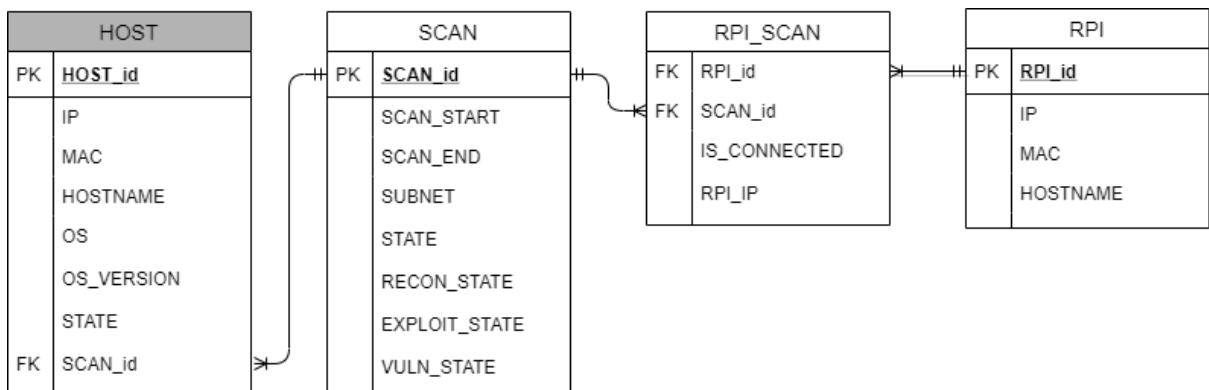


Figure 13 : Modèle ER : Partie client

La table RPI permet alors de récupérer les données qui identifient le client comme son IP, sa MAC adresse et son hostname.

La Table SCAN contient les timestamps qui permettent de dater le scan, en plus du réseau ciblé par celui-ci et finalement de garder un état de chacune des phases effectuées pendant la procédure.

La table RPI_SCAN qui fait la jonction entre les deux tables précédentes, permet de définir quels sont les clients actuellement en charge d'un scan spécifique et de fournir l'adresse IP qui leur est attribuée dans ce réseau.

Finalement, la table HOST, la plus importante de notre modèle, garde les informations principales permettant d'identifier chacun des hôtes scannés. Ces informations sont composées de ses identifiants ainsi que de son système d'exploitation et sa version s'ils ont été trouvés. Enfin, un état est gardé pour chaque hôte permettant de définir si la procédure de scan a été exécutée correctement ou si elle a rencontré une erreur.

La seconde permet de répartir les données récoltées sur les hôtes et est responsable de la reconnaissance et de la détection des vulnérabilités.

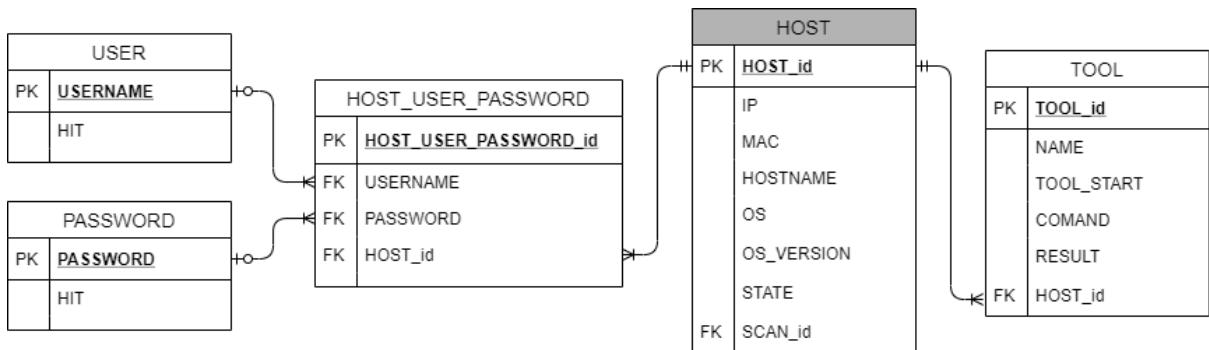


Figure 14 : Modèle ER : Partie hôte

La table HOST_SERVICE permet de mapper et d'enregistrer un historique de chacun des services trouvés sur les hôtes scannés ainsi que la version et le port de ceux-ci.

La table VULN est composée de l'ensemble des vulnérabilités connues et testées à ce jour les attributs repris dans ce projet sont leur sévérité, complexité, score de base et leur description. Ces attributs ont été extraits des *feeds* mis à disposition par le NIST, l'institut national des standards et de la technologie. *Feeds* qui contiennent toutes les vulnérabilités référencées depuis 1999.

Remarque : Bien d'autres attributs sont disponibles avec les *feeds* et selon l'avancement du projet ils pourraient être récupérés et ajoutés en base de données afin d'analyser les vulnérabilités rencontrées sous plusieurs aspects.

```
"baseMetricV2" : {  
    "cvssV2" : {  
        "version" : "2.0",  
        "vectorString" : "AV:N/AC:M/Au:N/C:C/I:C/A:C",  
        "accessVector" : "NETWORK",  
        "accessComplexity" : "MEDIUM",  
        "authentication" : "NONE",  
        "confidentialityImpact" : "COMPLETE",  
        "integrityImpact" : "COMPLETE",  
        "availabilityImpact" : "COMPLETE",  
        "baseScore" : 9.3  
    },  
    "severity" : "HIGH",  
    "exploitabilityScore" : 8.6,  
    "impactScore" : 10.0,  
    "acInsufInfo" : false,  
    "obtainAllPrivilege" : false,  
    "obtainUserPrivilege" : false,  
    "obtainOtherPrivilege" : false,  
    "userInteractionRequired" : true  
}
```

Figure 15 : Structure des feeds au format JSON

La table HOST_SERVICE_VULN ne sert ici que de table de jonction entre les deux tables précédentes afin de mapper chaque vulnérabilité au service où elle a été repérée.

La table SERVICE quant à elle permet de construire un dictionnaire des services rencontrés au cours de scans précédents tout en gardant le nombre de fois qu'ils ont été analysés. Ainsi lors des futurs scans, les services peuvent être analysés par ordre décroissant d'apparition afin de commencer par les plus courants.

Finalement, la troisième partie est responsable de la partie exploitation. Elle permet de stocker les résultats des outils qui ont été utilisés sur chaque host ainsi que leur commande et un timestamp lors de l'exécution de celui-ci.

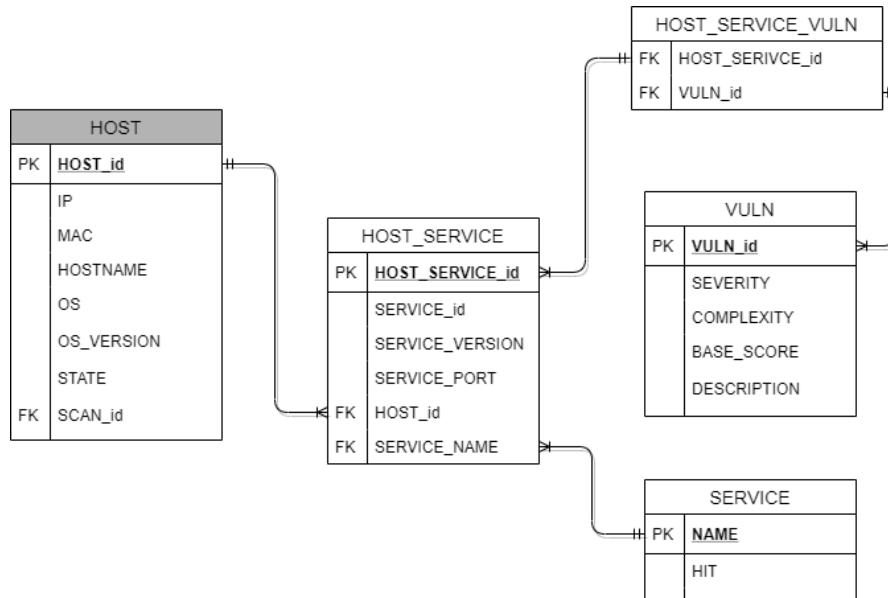


Figure 16 : Modèle ER : Partie service

De plus, lors de l'exécution des outils, des informations comme les identifiants découverts sur les machines sont récupérés afin d'alimenter un dictionnaire d'identifiant. Sur chacun de ces identifiants, un nombre de hits est conservé afin de pouvoir lors des prochaines phases de scan utiliser ce dictionnaire en favorisant les pairs d'identifiants les plus courants.

En joignant ces trois parties, le modèle relationnel complet peut ainsi être créé sur MYSQL Workbench et être exporté sous forme de requêtes SQL. Selon la version de MariaDB utilisé, les requêtes générées par MYSQL Workbench doivent être modifiées afin de correspondre à la syntaxe actuelle du SGBD. Cet ensemble de requêtes SQL lui-même exporté directement sur notre serveur distant permet la création de la base de données.

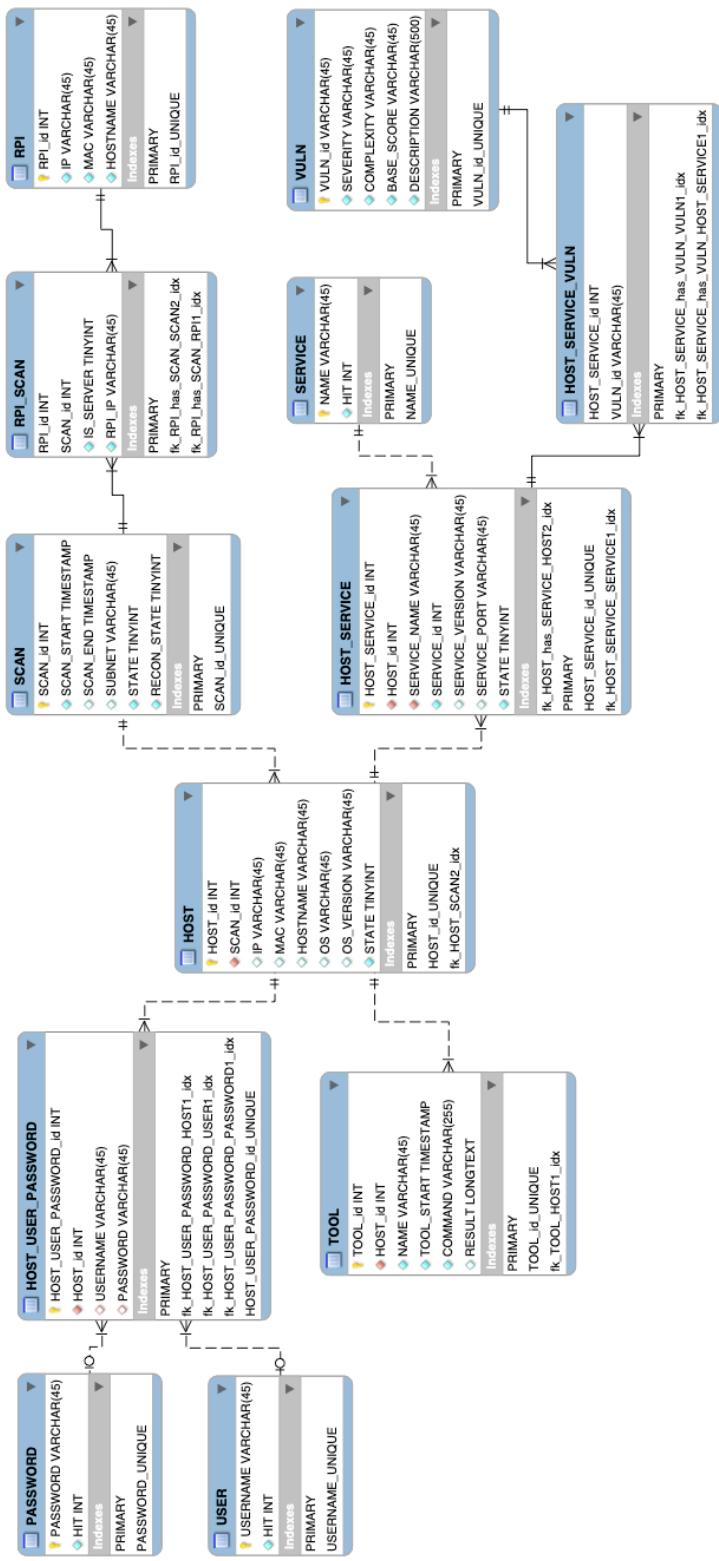


Figure 17 : Modèle ER complet

6.3 Procédure d'automatisation

La solution imaginée fonctionne sur base d'appel de scripts automatisés, chacun composé de fonctions ou de sous modules propre à la tâche qui leur est assignée. Il est important de rappeler que chaque partie du scan est fortement liée à la base de données qui permet le stockage et la gestion des données et informations récoltées durant l'exécution des scripts. C'est aussi cette dernière qui va permettre l'interaction entre les phases et les différentes parties du projet.

La procédure de scan est divisée en trois parties, respectivement les phases de reconnaissance, de scanning, et d'exploitation. Ces dernières sont appelées via un manager principal qui permet l'exécution complète du code et donc de toutes les phases.

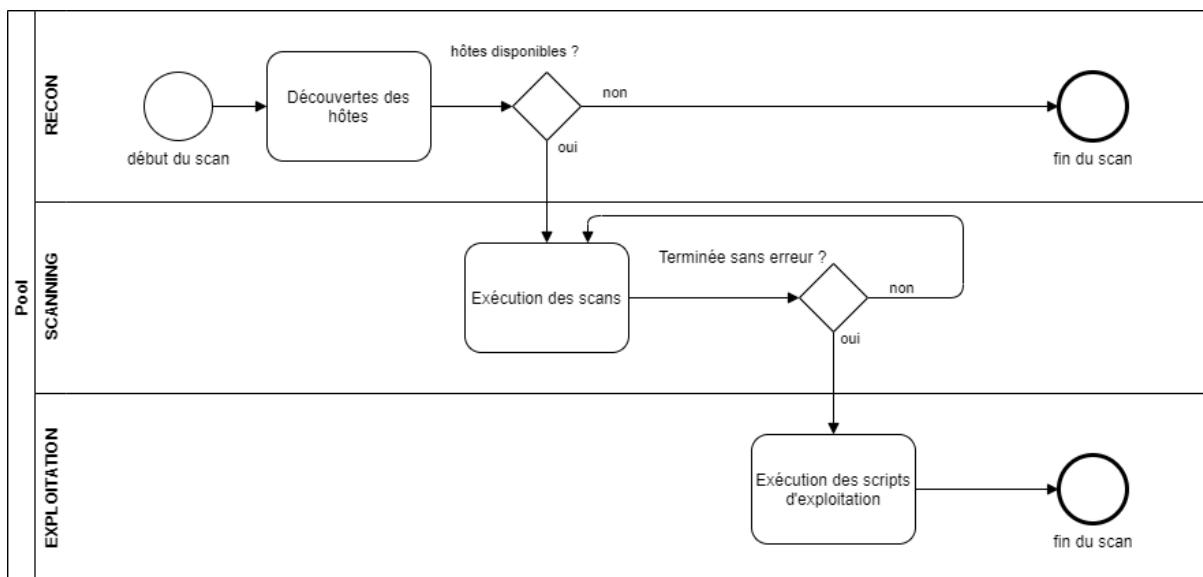


Figure 18 : Modèle BPMN des trois phases

La première phase, qui est celle de reconnaissance s'effectue de la manière suivante. Tout d'abord, les informations qui permettent d'identifier la machine qui exécute le scan ainsi que son environnement sont récoltées. Grâce à ces dernières, un scan est créé et introduit en base de données. L'IP, le MAC adresse et le nom d'hôte du RPI sont alors enregistrés et liés à ce dernier. C'est aussi cette première partie du code qui va permettre aux RPI de se synchroniser et de déterminer si un nouveau scan doit être créé ou si un scan est déjà en cours dans l'environnement actuel.

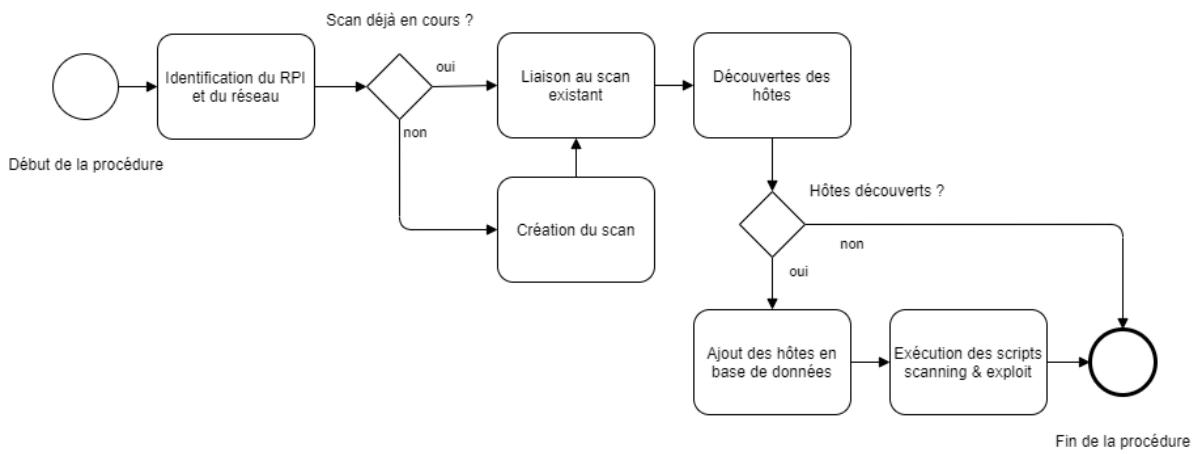


Figure 19 : Modèle BPMN de la phase 1 : Recon

Ensuite vient la seconde phase, celle de scan qui se concentre sur l'analyse des systèmes. Les informations qui permettent de mieux identifier ceux-ci seront récoltées afin définir leur utilisation au sein de l'environnement testé. Les informations attendues sont alors le type de système (selon la MAC adresse), le dernier redémarrage de la machine, son système d'exploitation et sa version ainsi que le *listing* de tous les services actifs. Ce sont ces données qui vont permettre de définir la démarche suivie lors de la prochaine phase. En effet, cette démarche sera différente si la cible présente un OS Windows ou Linux, de même que les services présents sur cette dernière.

De plus, un scan de vulnérabilité est effectué afin de déterminer les failles possiblement présentes. Les services précédemment listés vont alors être scannés plus en détail afin de connaître leur version. Cette version, une fois connue, permet de faire la relation avec la NVD¹³ mise à disposition par le NIST¹⁴ qui offre la totalité des vulnérabilités connues et testées par le CVSS, systèmes de *scoring* présenté ci-dessus.

Ces CVE, vulnérabilités testées disponibles sous forme de fichiers, doivent être extraites et ajoutées dans la base de données utilisée lors de ces tests. Chacune des vulnérabilités présumées pourra alors être liée en BD à l'hôte sur lequel elle a été découverte. En plus de permettre l'identification de la CVE, des attributs comme la complexité et la sévérité ont été retenus afin d'apporter une compréhension directe des conséquences que ces dernières pourraient avoir.

¹³ <https://nvd.nist.gov/vuln/data-feeds>

¹⁴ <https://www.nist.gov/>

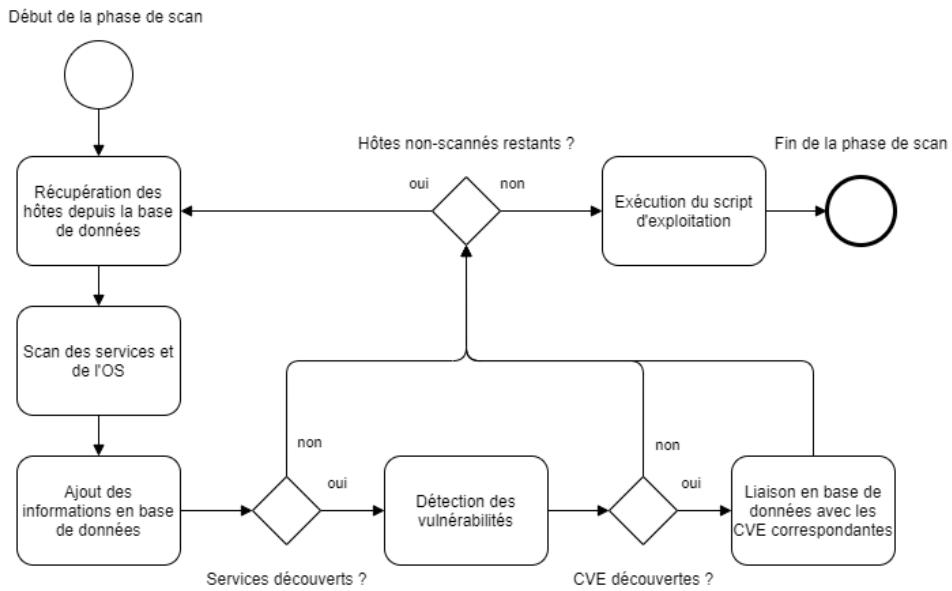


Figure 20 : Modèle BPMN de la phase 2 : Scanning

Finalement, c'est lors de la phase d'exploitation que les différents services vont être activement testés. Le script, comme expliqué plus haut, est conçu de manière à prendre en charge les hôtes un par un et de charger leurs informations depuis la base de données afin de définir la procédure à suivre. Chaque procédure est donc composée d'une suite de modules, eux-mêmes propre à un service spécifique et composés d'une suite de commandes parfois dépendantes les unes des autres, parfois non. Ces commandes sont une suite d'appel à des outils divers choisis pour les informations critiques qui peuvent résulter de leur exécution. C'est d'ailleurs ici que l'automatisation prend tout son sens, car chaque module d'exploitation peut être modifié afin de fournir des résultats propres au type d'analyse désiré.

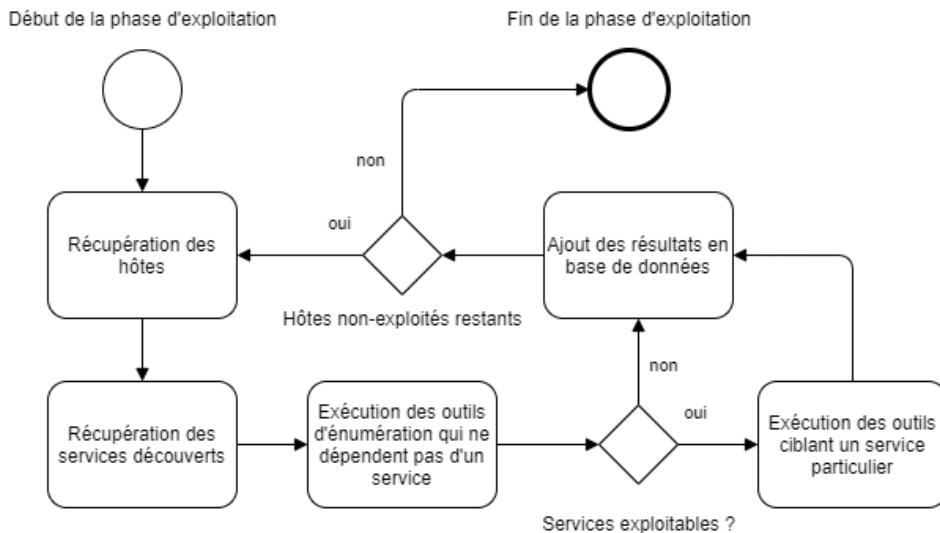


Figure 21 : Modèle BPMN de la phase 3 : Exploitation

6.4 Conception du Dashboard

Ce chapitre est consacré à la conception du *dashboard*, site web dont l'objectif est de faire office de rapport aux tests effectués. Celui-ci doit par la même occasion être le plus intuitif possible et permettre au mieux à une personne sans réelles compétences dans le domaine de déterminer si son infrastructure présente un risque.

Pour ce faire, le premier point consiste en la création d'une maquette qui permettra de définir le positionnement de chacune des données afin de fournir une ligne directrice lors de la création de ce dernier.

Ensuite, le résultat obtenu sera présenté et analysé afin de vérifier si l'objectif du *dashboard* a été atteint.

6.4.1 Développement de la maquette

Afin de concevoir un *dashboard* qui offrira une vision claire et interprétable de la situation, une maquette doit donc être réalisée. Celle-ci permet entre autres de réfléchir au positionnement des données ainsi qu'à la mise en évidence de celles-ci. Pour cela, la structure choisie respecte celle instaurée dans la base de données.

La première page accessible est celle dont l'objectif est de présenter à l'utilisateur les derniers scans effectués et des statistiques globales ainsi que les informations des scans en cours.

De plus, la solution développée pouvant être effectuée à l'aide de plusieurs agents et dans plusieurs réseaux en simultané, une liste des RPI permet de présenter les informations qui leur sont propres ainsi que les réseaux dans lesquels ils se trouvent et le scan auquel ils sont actuellement liés.

Grâce à cette première page, l'utilisateur pourra alors sélectionner un scan afin d'aller analyser les informations spécifiques à celui-ci.

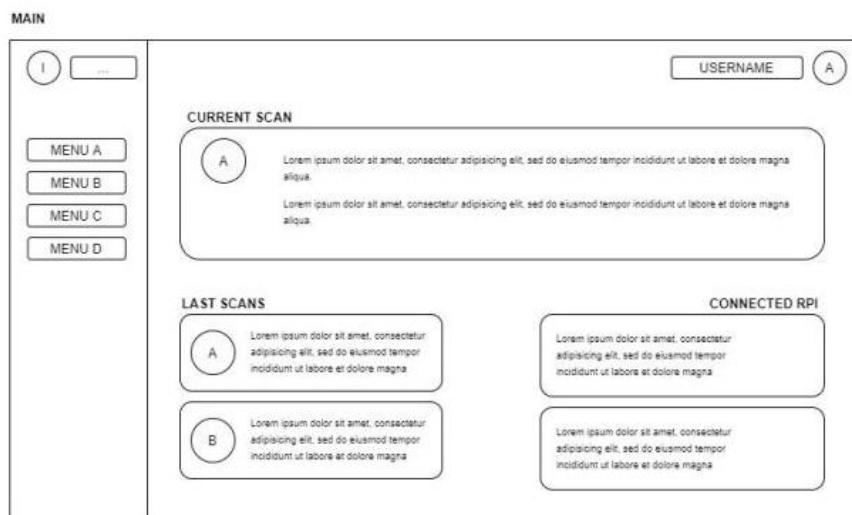


Figure 22 : Maquette site web main page

La seconde page a pour but d'afficher les informations propres à un scan particulier qui aura été sélectionné par l'utilisateur.

Sur cette page figurera la liste des hôtes découverts ainsi que des graphiques afin d'afficher un état global de la situation relativ au nombre de vulnérabilités trouvées et à leur criticité.

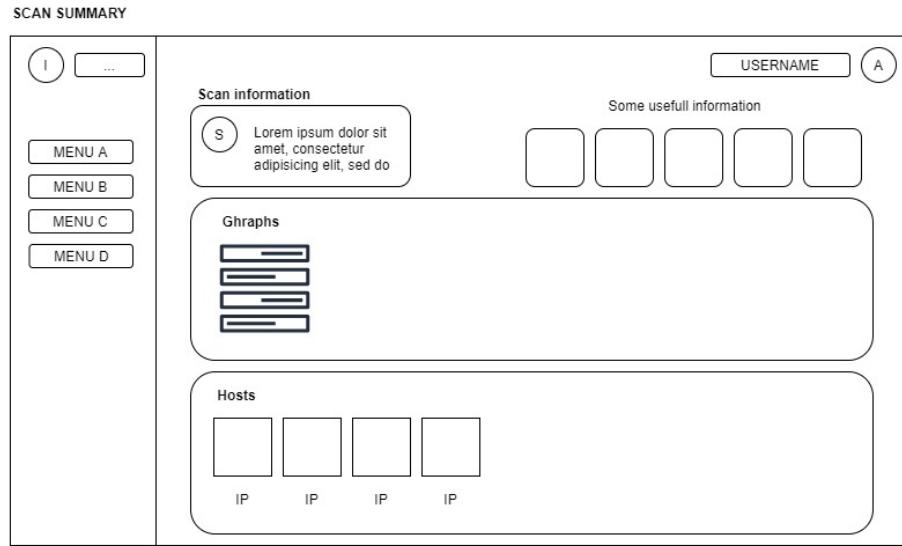


Figure 23 : Maquette site web scan page

La page apportant le plus de détails est celle des hôtes.

Celle-ci doit alors mettre en évidence, par des graphiques et des tableaux, les vulnérabilités découvertes sur cet hôte ainsi que les informations qui lui sont associées. La liste des utilisateurs et des services découverts sur ce système devra aussi y figurer.

Finalement, la liste des outils utilisés ainsi que les résultats de chacun de ceux-ci seront affichés afin de fournir un maximum d'informations sur ce qui a été effectué pendant le scan.

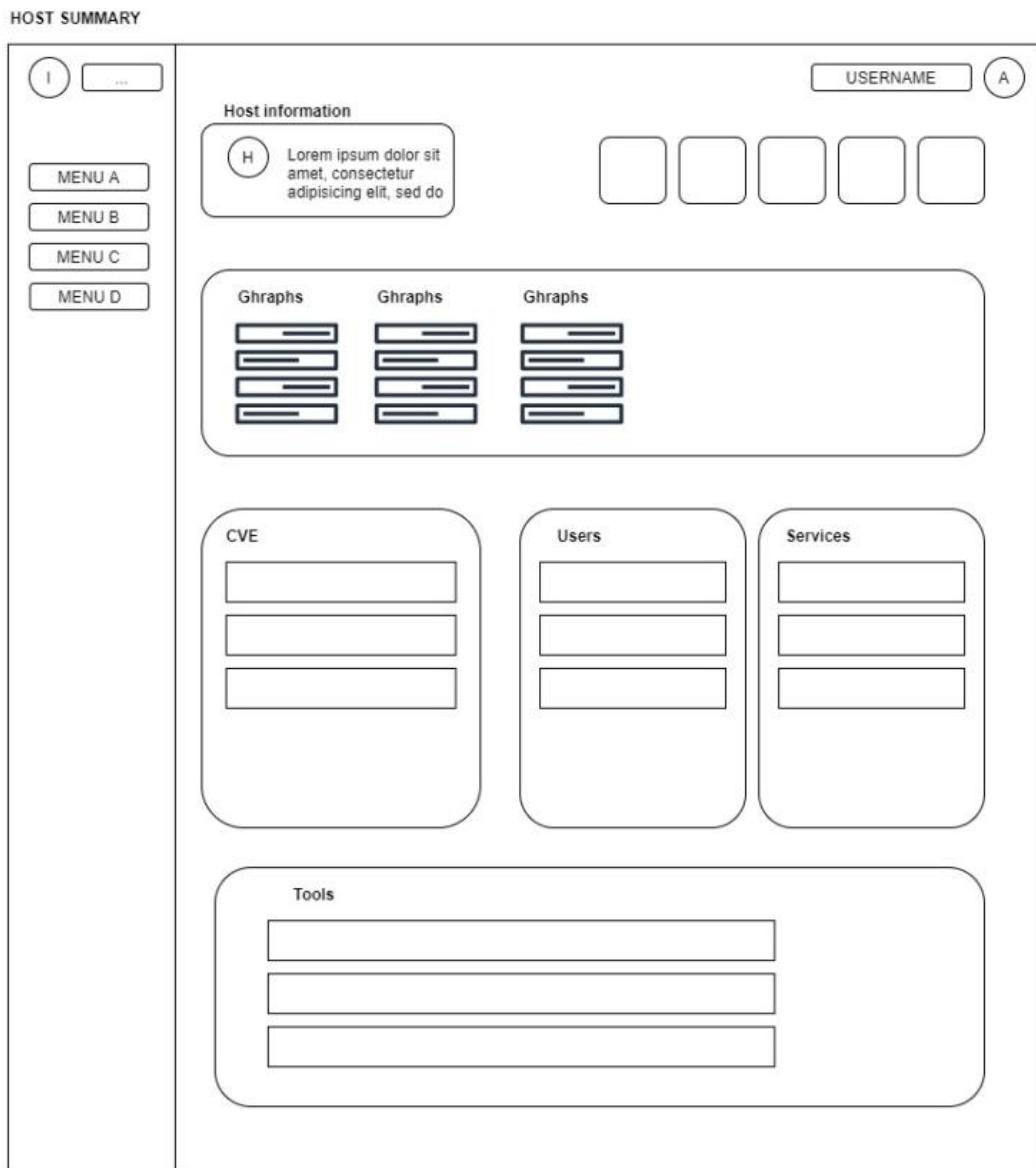


Figure 24 : Maquette site web host page

6.4.2 Analyse du Dashboard développé

Désormais réalisé, analysons si le *dashboard* permet de répondre aux objectifs qui lui ont été fixés.

En premier lieu, la structure des pages correspond à celles pensées lors de la réalisation de la maquette et chacune des informations a pu y être affichée de manière claire et lisible ce qui offre une prise en main rapide du site. Les informations de chaque scan peuvent être parcourues ainsi que celles de chacun des hôtes qui leur sont associés.

Au travers des différentes pages, les graphiques conçus grâce aux codes JavaScript fournis par l'API Bootstrap offrent au *dashboard* une très bonne lisibilité concernant le nombre de vulnérabilités et leur criticité ainsi que sur les actions qui ont été effectuées sur le système en question.

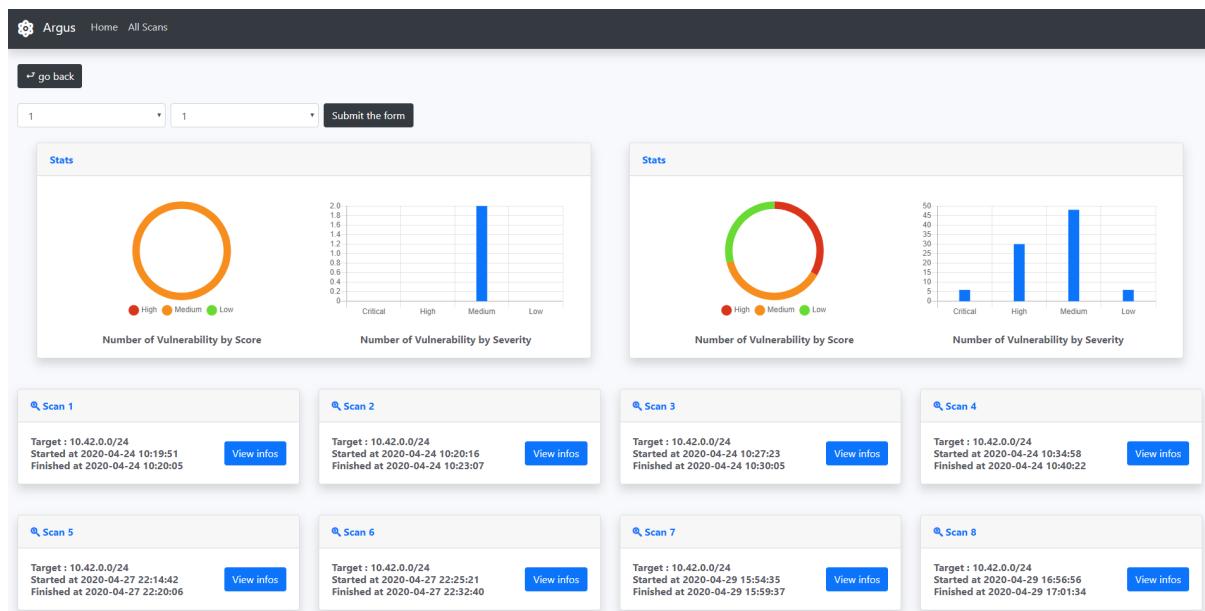


Figure 25 : Page de comparaison de scan

De plus, des fonctionnalités supplémentaires non pensées initialement ont été rajoutées au cours du développement.

Certaines plus impactantes que les autres, la première est l'ajout des barres de progression qui permettent de fournir un état d'avancement des scans en cours. La seconde est l'ajout d'une page qui offre une vue sur tous les scans effectués et qui permet soit de s'y rendre, soit de pouvoir sélectionner un scan afin d'effectuer un comparatif avec un scan antérieur.

D'autres encore sont présentes comme des raccourcis, icônes ou codes couleur, mais ceux-ci permettent plutôt une meilleure utilisation du site que de réelles fonctionnalités.

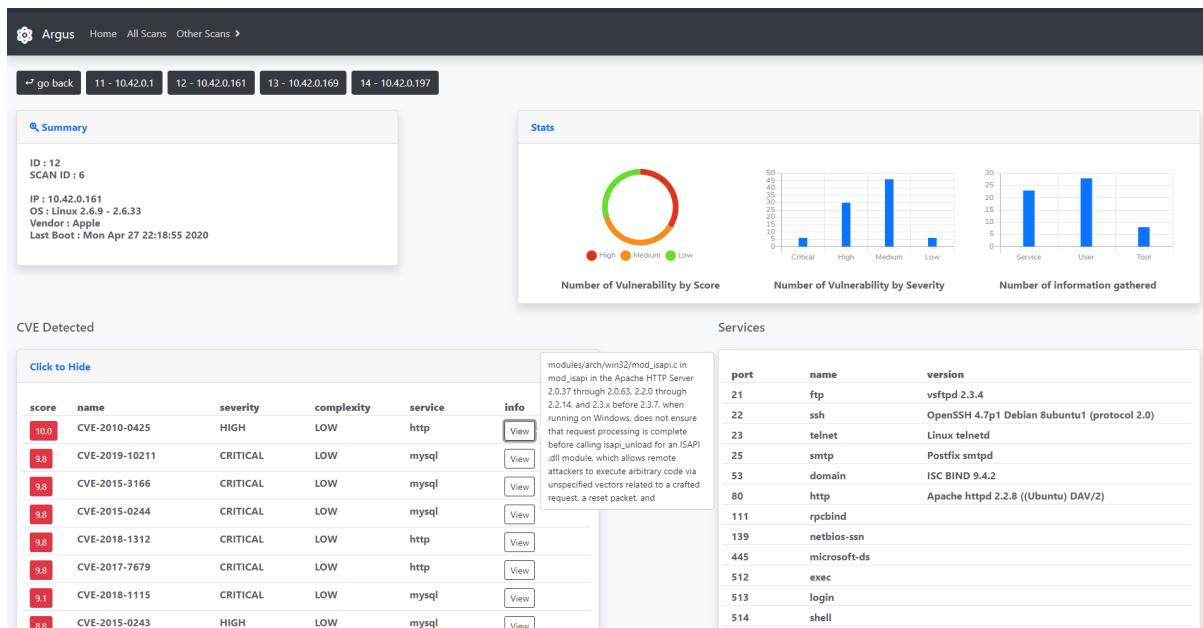


Figure 26 : Page hôte

Malgré la difficulté de rendre des résultats d'analyse compréhensibles pour tous, le *dashboard* permet pour autant de correctement les mettre en évidence et de les rendre facilement accessibles. Il fournit ainsi un compte rendu fiable et précis quant aux vulnérabilités, utilisateurs et services présents sur les systèmes pour chacun des hôtes présents sur le réseau.

7 Analyse rétrospective

Ce chapitre a pour objectifs de mettre en évidence les améliorations que pourrait subir le projet ainsi que les difficultés et les points forts rencontrés au cours de ce stage et pendant la conception de ce projet.

7.1 Améliorations possibles

Bien que ce projet touche à sa fin, la solution n'en est pas pour autant finalisée, certaines fonctionnalités explorées au niveau théorique et pratique au cours du développement de celui-ci n'ont pas pu aboutir et d'autres idées en revanche se sont rajoutées tout au long du projet.

La première que doit subir le projet est une amélioration globale de ses performances et un nettoyage de ses codes. Afin d'obtenir un résultat après ces 15 semaines, le projet a été pensé de manière fonctionnelle et nécessite donc une optimisation pour améliorer ses performances. Ainsi, différentes parties du client peuvent être largement optimisées notamment par l'ajout de *multithreading* afin d'augmenter la rapidité d'exécution des scans, ou encore la réécriture de certains modules afin de les rendre ici aussi plus performants et malléables. Le site web lui aussi doit subir des améliorations afin de respecter les principes de programmation web et devenir complètement *clean-code*.

La seconde cible plutôt les fonctionnalités du projet. En effet, en ce qui concerne le client, la majorité des outils qui ont été implémentés ne vise que les services les plus couramment détectés et utilisés, cependant au cours des tests la base de données contenant les services a commencé à se remplir de beaucoup d'autres. De plus, si le nombre d'outils vient à considérablement augmenter, une fonctionnalité permettant de sélectionner uniquement certains d'entre eux deviendra absolument nécessaire afin de garder la solution modulable et performante. Quant au *dashboard*, avec l'ajout de nouveaux outils et donc de nouvelles données récoltées, il devra être mis à jour constamment afin d'afficher et de classer ces dites informations.

Troisièmement, les performances de celui-ci peuvent aussi être améliorées grâce aux futures technologies. Raspberry a par exemple annoncé une version de son nano-ordinateur avec une capacité de mémoire de 8Gb contre seulement la moitié pour la version utilisée au cours de ce projet. Bien que le système d'exploitation Kali mette à disposition une large gamme d'outils, il n'en reste pas moins un système d'exploitation à utilisation personnelle et qui n'est pas destiné à ce genre d'intégration dans un projet. L'OS Parrot et son portage sur les architectures ARM qui a été brièvement cité au point 5.1.2 «Choix du système d'exploitation» permettrait quant à lui d'apporter une couche de sécurité bien supérieure à la solution grâce à une meilleure gestion des permissions par défaut ainsi que par son intérêt poussé pour l'anonymisation.

8 Conclusion

Afin de conclure la réalisation de ce projet, il est nécessaire de passer en revue les différentes étapes accomplies qui ont permis la création de cette solution de détection de vulnérabilités.

Tout d'abord, une phase de documentation a été réalisée afin de construire une analyse des différents domaines relatifs au projet. Chacun a donc été analysé dans son fonctionnement et sa nature afin d'en extraire les composants et les technologies qui pourraient ensuite servir à construire ce projet.

Ensuite, sur base de cette analyse, les fonctionnalités de chaque composant ont pu être définies et associées aux outils permettant leur intégration au sein de la solution. Que ce soit le service de base de données, des langages utilisés ou les autres services, tous ont été sélectionnés avec soin par rapport aux objectifs du projet.

La solution doit cependant encore être améliorée au niveau de ses performances et peut intégrer d'autres outils permettant une plus grande précision dans les résultats obtenus et une augmentation de la prise en charge des vecteurs d'attaque.

C'est donc, non pas facilement, mais en abordant des thématiques et des technologies nouvelles que l'objectif fixé au cours de ce TFE qui était de concevoir une plateforme facilement déployable, discrète, lightweight et abordable a pu être accompli. Il démontre, de plus, la faisabilité de la problématique de base qui était de réaliser cette plateforme afin d'aider les PME à aborder le monde de la sécurité informatique.

9 Webographie

- Anderson, K. (2019, mars 6). *2019 Database Trends – SQL Vs. NoSQL, Top Databases, Single Vs. Multiple Database Use*. From High Scalability:
<http://highscalability.com/blog/2019/3/6/2019-database-trends-sql-vs-nosql-top-databases-single-vs-mu.html>
- Education, I. C. (2019, mai 9). *LAMP Stack*. From IBM Cloud:
<https://www.ibm.com/cloud/learn/lamp-stack-explained>
- FORS R&D HENALLUX. (2020, mars 8). From fors-ing-henallux: <https://www.fors-ing-henallux.be/>
- Galiana, D. (2019, juillet 18). *CyberSécurité : Les 49 Statistiques Marquantes*. From Wimi:
<https://www.wimi-teamwork.com/fr/blog/chiffres-statistiques-marquants-cybersecurite/>
- GitHut 2.0 . (2020). From madnight github.io:
https://madnight.github.io/githut/#/pull_requests/2020/1
- Midena, M. (2019, juin 28). *Cyberattaques : Un Gouffre Financier Pour Les PME*. From Forbes:
<https://www.forbes.fr/technologie/cyberattaques-un-gouffre-financier-pour-les-pme/?cn-reloaded=1&cn-reloaded=1>
- NetExplorer. (2019, décembre 30). *CYBERSÉCURITÉ : 2019 EN CHIFFRES*. From NetExplorer:
<https://www.netexplorer.fr/blog/cyber-securite-2019-en-chiffres>
- Osborne, C. (2019, octobre 9). *Cyberattaques : 66 % des PME ciblées au cours de l'année écoulée*. From zdnet: <https://www.zdnet.fr/actualites/cyberattaques-66-des-pme-ciblees-au-cours-de-l-annee-ecoulee-39891887.htm>
- Pentest Methodologies. (2016, février 25). From verifyit: <https://www.verifyit.nl/wp/?p=175054>
- re4son. (2020, Mars 24). *Kali Installation*. From Kali: <https://www.kali.org/docs/installation/>
- Tenable. (2019, avril 7). *Tenable Scan Strategy*. From Tenable:
https://docs.tenable.com/other/nessus/Tenable_ProServ_Scan_Strategy_Guide.pdf

10 Annexes

10.1 Rapport de stage

10.1.1 Présentation de l'entreprise

Voir chapitre Présentation de l'entreprise

10.1.2 Présentation du projet

Voir chapitre Présentation du projet

10.1.3 Rapports journaliers

10.1.3.1 Période du 3 février au 29 février

Ces premières semaines sont destinées à la bonne intégration au sein du centre FoRS, le centre de recherche et développement de l'Henallux ainsi qu'à définir les objectifs du stage et à faire un premier état de l'art.

La première étape de mon projet consiste en une phase de documentation qui sert à définir correctement les objectifs, composants et procédures nécessaires à la conception de celui-ci.

Au bout de mes recherches, j'ai donc pu définir quatre axes majeurs. Le processus d'automatisation, l'implémentation d'une base de données (BD), les technologies et services relatifs au Raspberry (RPI) et finalement un Dashboard qui offrira une vue sur le déroulement des tests.

10.1.3.1.1 Recherches et analyses

Ce premier mois est destiné à la documentation, la liste des recherches et des réflexions quant à celle-ci fut longue.

Dans un premier temps, j'ai donc établi une liste des solutions déjà existantes, relatives de près ou de loin à mon sujet. Le milieu du *pentesting* est très vaste et présente déjà bon nombre de solutions plus créatives et efficaces les unes que les autres. En effet, l'automatisation est fortement liée aux tests d'intrusion, il existe des programmes tels que Maltego pour la phase d'OSINT ([open-source intelligence](#)) ou encore Armitage et Cobalt Strike pour celle d'exploitation. Il existe aussi énormément de programmes open source disponibles sur GitHub ou encore des programmes propriétaires comme PenTera 3.0.

Cependant, aucune solution ambitieuse qui repose principalement sur l'utilisation de Raspberry ne semble exister. La majorité d'entre elles se focalise sur l'installation de système d'exploitation comme Kali ou encore de craquage de wifi qui tire majoritairement parti de la mobilité du nano-ordinateur.

Mon deuxième axe de recherche consiste à définir les composants et outils nécessaires. L'environnement fourni avec le RPI est un système d'exploitation Linux Debian destiné à une

utilisation basique. Afin d'accéder directement à un environnement plus adéquat à mon stage, l'installation d'un OS tel que Kali est nécessaire. Ce dernier offre d'ailleurs une multitude de programmes et de librairies préinstallés.

En ce qui concerne l'écriture du processus d'automatisation, l'utilisation du Python semble être la plus adéquate. En effet, ce langage figure parmi les tendances actuelles en termes de développement et la majeure partie des outils déjà présents ainsi que ceux disponibles sur GitHub sont d'ailleurs développés avec ce même langage. De plus, il permet d'utiliser nombre de modules qui permettent l'interaction avec les futurs composants du projet. Par exemple la base de données, l'utilisation de fichiers au format JSON et CSV ou encore d'exécuter des commandes par le système d'exploitation.

Mes recherches se sont ensuite concentrées sur l'utilisation de la base de données, le traitement des données et leur affichage. J'ai donc effectué différentes recherches sur la conception théorique de celles-ci ainsi que sur les outils et modules qui me permettent de communiquer avec cette dernière.

Par chance les personnes présentes au DTM sont de grands habitués de ces concepts et représentent une grande aide dans la conception de ma propre base de données. De par leur nature, ces BD doivent être conçues de la manière la plus simple et évolutive possible afin de fournir les meilleures performances.

Finalement, c'est la partie concernant l'interface web qui est la plus simple à traiter. En effet, l'utilisation d'un Template (gratuit et disponible en open source sur GitHub) ainsi qu'un Dashboard précédemment développé lors du cours de sécurité des applications permettent d'offrir un compte rendu sur les différents scans. Les fonctionnalités de celui-ci doivent donc être la gestion des sessions, un système d'authentification et des interactions avec la base de données grâce au PHP. La majorité du travail ici se voit être plus pratique que théorique.

10.1.3.1.2 Notions découvertes

Au cours de mes recherches, j'ai découvert une solution nommée PenTera, développée par PcySys dont le but est l'automatisation des tests d'intrusion. PenTera est donc une plateforme dont les différentes phases vont être automatisées. Cette plateforme propose un rendu en direct des vulnérabilités découvertes sur les différents hôtes ainsi que de démontrer par des corrélations entre les événements d'où proviennent ces failles.

La majorité des autres notions avaient déjà été introduites lors de mon cursus. Par exemple, l'utilisation du Python, MySQL et de l'OS Linux ou encore des langages web. Cependant, le fait de les utiliser conjointement rend la chose bien plus ludique. Et cela introduit nombre de modules à utiliser. Par exemple pour permettre les transferts entre le script d'automatisation et la base de données, ou encore l'exécution de commandes par le système d'exploitation comme Nmap.

Ensuite, lors de l'écriture du script en Python j'ai la nécessité d'extraire des informations qui proviennent par exemple d'un scan Nmap. Ce fut la parfaite occasion d'apprendre à utiliser les expressions régulières (REGEX) qui repèrent un paterné donné indépendamment de la donnée entrée (input).

Finalement, lors de cette dernière semaine, je me suis concentré sur les fonctionnalités du Raspberry. La première est la possibilité de fournir un accès à distance grâce à la puce wifi intégrée.

Pour cela, le système d'exploitation Kali installé permet la création d'un point d'accès protégé par une clé WPA2. Ce point d'accès peut ensuite être utilisé afin d'obtenir une connexion en SSH sur la RPI. La deuxième fonctionnalité imaginée est de fournir en plus de l'accès SSH, une connexion au bureau à distance. Pour cela il existe plusieurs protocoles, comme RDP (Remote Desktop Protocol) ou VNC (Virtual Network Computing). Là où RDP permet l'accès à distance en créant une nouvelle session propre à chaque connexion, VNC partage une seule et même session. Dans le cas présent j'ai décidé d'opter pour VNC, car il permet de visualiser ce qui est réellement exécuté sur la machine. Plusieurs types de serveurs sont alors disponibles comme tightVNC, realVNC ou encore X11VNC. Chacun a donc dû être installé et testé afin de définir celui qui conviendrait le mieux. X11 par exemple, malgré sa stabilité, nécessite une connectivité en HDMI au Raspberry. Connectivité qui peut être émulée par un accessoire branché directement sur la RPI, mais qui dans le cas présent ne convient pas aux objectifs fixés. RealVNC lui n'est pas accessible depuis l'application mobile vncViewer et a donc été écarté. Finalement c'est tightVNC après quelques modifications dues à des dépendances liées au GUI, qui a permis l'accès à distance à la machine sans moniteur (Headless).

10.1.3.1.3 Tâches réalisées

Parallèlement aux recherches, j'ai la nécessité d'essayer les différents outils et modules sélectionnés. D'une part pour comprendre leur fonctionnement et d'autre part pour vérifier leur compatibilité. Afin de prendre en main ceux-ci, la conception d'un environnement de test est inévitable. Pour ce faire, j'ai configuré une machine Linux Debian avec des ressources similaires à celles d'un Raspberry. Pour la partie automatisation, l'environnement étant orienté Linux, seule l'installation de python3 est nécessaire. Enfin, le serveur web est hébergé sur un serveur apache2 et la gestion de la base de données est réalisée via MySQL - Maria DB et phpMyAdmin.

En ce qui concerne la base de données, j'ai premièrement défini les différentes classes qui seraient nécessaires lors du processus de récolte des données sur les hôtes. Ensuite, sur VisualParadigm j'ai pu créer un schéma d'entité relationnel qui peut ensuite être converti en commande SQL. Ces commandes SQL peuvent directement être utilisées pour générer ou recréer la base de données. Quant au script d'automatisation, la "première phase" touche à sa fin. La machine est maintenant capable de reconnaître son environnement (IP / Mask / subnet). Le code doit cependant être amélioré. Elle repère ensuite les différentes machines du même sous réseau, avant de passer à un scan plus profond qui permet de repérer leur système d'exploitation, mac adresse, version des services et leur domaine. De plus, le script permet d'établir une liste des vulnérabilités (CVE) exploitables en fonction de la version des services. La liste des CVE a été fournie par le « National Institute of Standards and Technology » (NIST) et téléchargée au format JSON avant d'être importée dans la base de données.

```

1 import nmap
2 import re
3 import os
4 import sys
5 import subprocess
6 import ipaddress
7 import mysql.connector
8 from subprocess import PIPE, run
9
10 import tools as tools
11
12 > db = mysql.connector.connect( ...
13
14 cursor = db.cursor()
15
16 > def get_network(): ...
17
18
19 > def get_hosts_from_network(host_info): ...
20
21 > def parse_nmap_service(host, scan_report, host_info): ...
22
23 > def parse_nmap_cve(host, scan_report, host_info): ...
24
25 > def parse_nmap_info(host, scan_report, host_info): ...
26
27 > def get_cve_from_hosts(hosts, host_info): ...
28
29 > def main(): ...
30
31 > if __name__ == "__main__":

```

Figure 27 : Structure du code

10.1.3.1.4 Problèmes rencontrés

Lors de la conception d'un projet, rien n'est fonctionnel du premier coup. En effet, chaque composant de celui-ci nécessite son taux de recherche et des tests.

Les premiers problèmes rencontrés sont principalement dus à des dépendances du système d'exploitation et à des erreurs de compatibilité.

Pour l'automatisation, la plus grosse difficulté réside dans le fait que le script Nmap qui permet la détection des vulnérabilités effectue des recherches sur un serveur externe. Ce script nécessite donc une connexion en WAN pour fonctionner, en plus de surcharger le réseau.

De plus, le manque de connaissances théoriques et pratiques en matière de base de données m'a fortement ralenti lors de la création de celle-ci. Alexandre Marchal de l'équipe du FoRS m'a donc accompagné lors de sa conception en me réexpliquant certains concepts.

```
5432/tcp open postgresql PostgreSQL DB 8.3.0 - 8.3.7
| vulners:
|   cpe:/a:postgresql:postgresql:8.3:
|     CVE-2016-7048 9.3 https://vulners.com/cve/CVE-2016-7048
|     CVE-2019-10211 7.5 https://vulners.com/cve/CVE-2019-10211
|     CVE-2015-3166 7.5 https://vulners.com/cve/CVE-2015-3166
|     CVE-2015-0244 7.5 https://vulners.com/cve/CVE-2015-0244
|     CVE-2017-14798 6.9 https://vulners.com/cve/CVE-2017-14798
|     CVE-2015-0243 6.5 https://vulners.com/cve/CVE-2015-0243
|     CVE-2015-0242 6.5 https://vulners.com/cve/CVE-2015-0242
|     CVE-2015-0241 6.5 https://vulners.com/cve/CVE-2015-0241
|     CVE-2018-1115 6.4 https://vulners.com/cve/CVE-2018-1115
|     CVE-2015-3167 5.0 https://vulners.com/cve/CVE-2015-3167
|     CVE-2012-2143 4.3 https://vulners.com/cve/CVE-2012-2143
|     CVE-2014-8161 4.0 https://vulners.com/cve/CVE-2014-8161
|     CVE-2010-0733 3.5 https://vulners.com/cve/CVE-2010-0733
|     CVE-2019-10210 1.9 https://vulners.com/cve/CVE-2019-10210
5900/tcp open vnc VNC (protocol 3.3)
6000/tcp open X11 (access denied)
6667/tcp open irc UnrealIRCd
```

Figure 28 : Script NMAP de détection de CVE

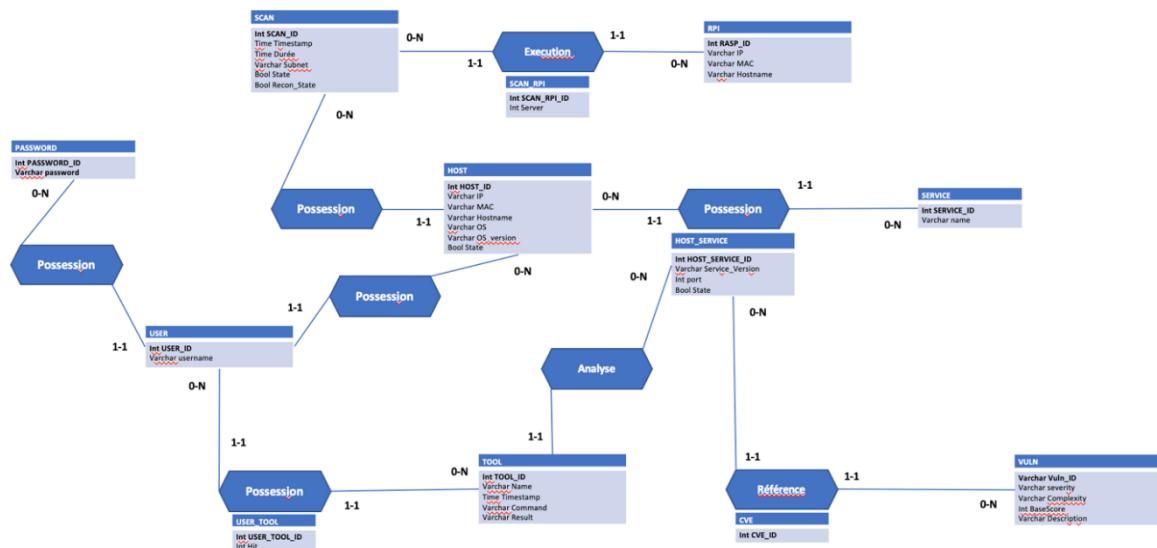


Figure 29 : Structure temporaire de la BD

En ce qui concerne la configuration du serveur VNC, dû aux différents services à disposition et aux nombreuses dépendances relatives autant au système d'exploitation qu'au GUI, la documentation propre à leur configuration est souvent partielle ou inexacte et m'a plusieurs fois mené à la réinstallation de l'interface graphique ou du système. De plus, chaque service comporte un grand nombre d'options qui mènent à des incompatibilités avec le moniteur et le GUI. L'installation d'un autre GUI que celui proposé par défaut sur Kali m'a permis de patcher ce premier.

10.1.3.1.5 Références

National Vulnerability Database : https://nvd.nist.gov/vuln/data-feeds#JSON_FEED

Script Vuln NMAP : <https://securitytrails.com/blog/nmap-vulnerability-scan>

MySQL Connector : <https://www.a2hosting.com/kb/developer-corner/mysql/connecting-to-mysql-using-python>

Linux Exploits : <https://gtfobins.github.io/>

Regex : <https://www.rexegg.com/regex-quickstart.html>

Databases : <https://www.a2hosting.com/kb/developer-corner/mysql/managing-mysql-databases-and-users-from-the-command-line>

Securing Raspberry : <https://www.raspberrypi.org/documentation/configuration/security.md>

Brute Forcing : <https://redteamtutorials.com/2018/10/25/hydra-brute-force-techniques/>

Exploits : <https://redteamtutorials.com/2018/10/24/msfvenom-cheatsheet/>

Bootstrap Dashboard : <https://github.com/BlackrockDigital/startbootstrap-sb-admin-2>

Pentesting Bible : <https://github.com/blaCCkHatHacEEkr/PENTESTING-BIBLE/tree/master/1-part-100-article>

Livre – Pentesting with raspberry :

https://books.google.be/books?id=YZ_cDgAAQBAJ&pg=PA247&lpg=PA247&dq=solution+pentest+raspberry&source=bl&ots=WCcLYoxjzZ&sig=ACfU3U0xWtlqEDDeBlbsapP9sbLDuu6inw&hl=fr&sa=X&ved=2ahUKEwjo4vWuod3nAhVJRUIHZ0XC1MQ6AEwCXoECAoQAQ#v=onepage&q&f=false

Pentesting Guide : http://virgil-cj.blogspot.com/2018/02/enumeration-is-key_6.html

PcySys YouTube : <https://www.youtube.com/channel/UCo6aPJnsyvoUkDHdNHQoEXQ/videos>

PcySys Site : <https://www.pcysys.com/>

10.1.3.2 Période du 30 février au 13 mars

10.1.3.2.1 Recherches et analyses

Ma phase de prise en main terminée, je me suis concentré sur la recherche des programmes et modules qui pourraient permettre d'améliorer le projet. Pour cela, le cours de base de données d'informatique de gestion de 2^e année m'a permis de mieux comprendre la gestion des BD.

Ensuite, j'ai cherché à optimiser mon code afin de le rendre plus lisible et performant. J'ai aussi essayé de trouver des alternatives qui mèneraient à une gestion plus efficace du processus d'automatisation. Par exemple, le Framework Metasploit propose de générer des scripts sous le format « .RC ». Ces scripts, combinés à celui écrit en python, pourraient dès lors être utilisés lors de la phase d'énumération et d'exploitation des services.

Finalement, l'utilisation d'Openvas (Framework d'évaluation de vulnérabilité) au travers de Metasploit semblait être une bonne alternative au problème précédemment rencontré lors de la détection de vulnérabilité.

10.1.3.2.2 Notions découvertes

Tout d'abord, le cours de base de données m'a permis de mieux comprendre l'enjeu que représente la conception d'une BD pour ensuite l'optimiser au mieux. Le schéma précédemment conçu a donc été revu dans sa globalité afin d'éviter toute redondance et d'optimiser la gestion des données envoyées et reçues à cette dernière pendant le processus d'automatisation.

Ensuite, des recherches plus poussées sur les modules utilisés lors du processus d'automatisation ainsi que l'implémentation de programmation orientée objet (POO) m'ont permis de réduire le code et sa durée d'exécution.

L'utilisation d'Openvas grâce à Metasploit m'était en revanche complètement inconnue. En effet, le Framework permet de charger ce dernier sous forme d'un module. Des commandes permettent alors de définir les hôtes à scanner et d'ensuite générer un rapport sous plusieurs formats.

```
16
17 > db = mysql.connector.connect(-
23   cursor = db.cursor()
24
25   main_interface = 'eth0'
26   self_interface = 'wlan0'
27
28   lock = Lock()
29
30 class Host:
31
32 >   def __init__(self, host_id, ip, mac, hostname, os, os_version, state, scan_id):-
41
42 >   def get_service(self, lock):-
78
79 >   def cve_scan(self, service_list):-
90
91 >   def parse_cve(self, scan_report):-
117
118 >   def get_RPI_info(): # Return IP / self IP / Netmask / Subnet-
164
165 >   def get_hosts(subnet, scan_id, rpi_id):-
179
180 >   def multiproc(object_list):-
184
185 >   def create_hosts_instance(scan_id):-
202
203 >   def main():-
213
214   if __name__ == "__main__":
215     # execute only if run as a script
216     main()
```

Figure 30 : Structure du code

10.1.3.2.3 Tâches réalisées

Suite aux conseils avisés d'Alexandre Marchal, j'ai décidé d'abandonner VisualParadigm pour MySQL Workbench. Ce programme permet, tout comme le premier, de concevoir un schéma d'entité relationnelle avant de l'importer dans la base de données. Cependant, celui-ci est plus simple d'utilisation et permet de définir avec plus d'efficacité les interactions entre les différentes tables.

Voici ci-dessous le nouveau schéma, optimisé et bien plus lisible que le précédent :

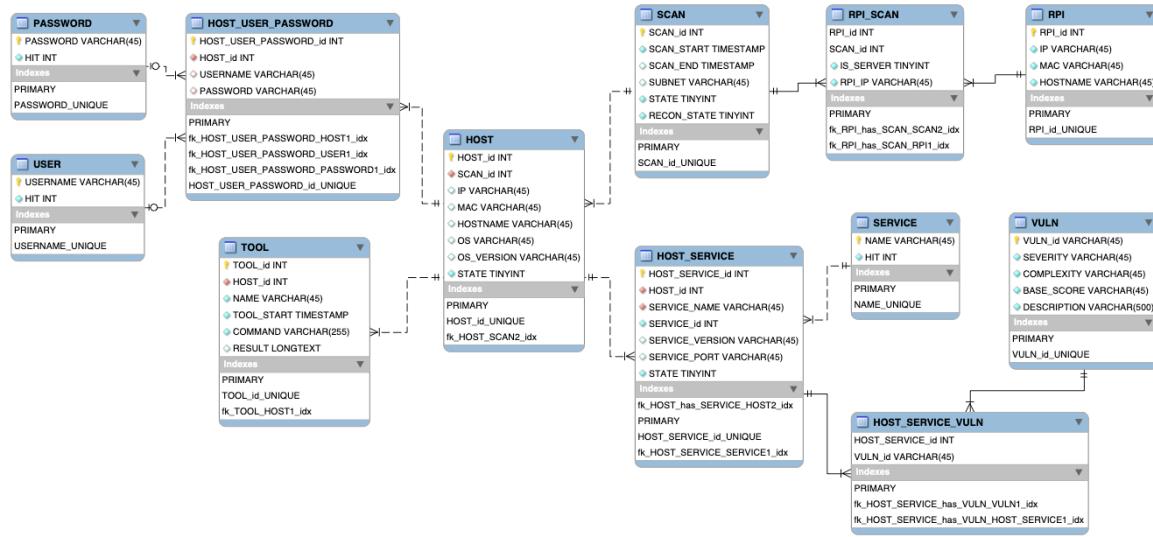


Figure 31 : Schéma ER - MySQL Workbench

Des flags de contrôle sont dès lors intégrés à certaines tables, ce qui permet de répartir les tâches à effectuer entre les RPI pendant la procédure de scan. Par exemple, une fois les hôtes repérés, ces flags permettent de définir si les actions propres à chaque système sont soit en cours, soit finalisées ou si une erreur a été rencontrée.

En ce qui concerne le code de reconnaissance, sa longueur a été diminuée de près d'un tiers grâce à la POO et toute la partie de récupération des informations du RPI et des hôtes a été simplifiée. De plus, l'utilisation du regex qui était auparavant omniprésente dans le script est désormais utilisée uniquement lorsqu'aucune autre possibilité n'est envisageable.

Pendant la phase d'optimisation, j'ai tenté d'implémenter du multithreading afin de réduire le temps d'exécution des scans Nmap, cependant ces derniers entrent régulièrement en conflit ce qui empêche le processus de se terminer. De plus, le taux d'information généré lors de ces scans sature la base de données. Il semble que cela soit un problème courant à l'utilisation du module Nmap.

Bien que le module de multithreading ne puisse être implémenté lors de la phase de scan, son utilisation sera certainement d'une grande aide pendant la phase d'exploitation.

Après ces optimisations, le scan dont la durée était d'environ 15 minutes (sur un environnement de test conséquent) s'effectue maintenant en moins de 6 minutes.

10.1.3.2.4 Problèmes rencontrés

Afin de régler le problème rencontré lors de l'implémentation du multithreading, j'ai eu l'idée d'intégrer des « locks ». Ces locks permettent de verrouiller temporairement l'accès à une ressource. Malgré cela, le problème n'a pas pu être résolu. Une autre solution possible à ce dernier est l'écriture d'un module de scan. Pour cela, le module Scapy, dont l'utilisation a été vue lors du cours de développement de 2^e année, est conçu à cet effet et semble être une bonne alternative au module Nmap actuel.

En ce qui concerne l'utilisation d'Openvas, le principal problème repose sur la façon dont le module définit les ID. Le module travaille de la façon suivante. Premièrement, une cible doit être définie, avant d'être ajoutée à une liste. Ensuite, il faut créer une tâche liée à un ou plusieurs cibles qui, une fois réalisée, peut être exportée sous la forme d'un rapport.

Cependant, afin de lier ces éléments, Openvas utilise des ID générés aléatoirement dont le format est une chaîne de caractère en hexadécimal, et qui ne peuvent donc pas être récupérés pendant l'exécution, ce qui empêche l'utilisation du module. Ces ID étaient auparavant sous forme numérique, mais plus depuis les dernières versions.

```
Object#timeout is deprecated, use Timeout.timeout instead.
[+] OpenVAS connection successful
msf > openvas_target_create "Metasploitable2" 10.0.2.17 "Linux Target"
/usr/share/metasploit-framework/vendor/bundle/ruby/2.5.0/gems/openvas-omp-0.6.0/lib/openvas/objects.rb:10: Object#timeout is deprecated, use Timeout.timeout instead.
[*]
/usr/share/metasploit-framework/vendor/bundle/ruby/2.5.0/gems/openvas-omp-0.6.0/lib/openvas/objects.rb:10: Object#timeout is deprecated, use Timeout.timeout instead.
[+] OpenVAS list of targets

ID          Name
comment      -
--          -----
3cf3cb85-85a1-44a8-a62a-571b1a63c4a3  Target for immediate scan of IP 10.0.2.17
f5f28e89-557e-453c-a946-4d52fcf234f6  Metasploitable2
linux Target

msf > openvas_config_list
/usr/share/metasploit-framework/vendor/bundle/ruby/2.5.0/gems/openvas-omp-0.6.0/lib/openvas/objects.rb:10: Object#timeout is deprecated, use Timeout.timeout instead.
[+] OpenVAS list of configs

ID          Name
--          -----
085569ce-73ed-11df-83c3-002264764cea  empty
2d3f051c-55ba-11e3-bf43-406186ea4fc5  Host Discovery
698f691e-7489-11df-9d8c-002264764cea  Full and fast ultimate
708f25c4-7489-11df-8094-002264764cea  Full and very deep
74db13d6-7489-11df-91b9-002264764cea  Full and very deep ultimate
8715c877-47a0-438d-98a3-27c7a6ab2196  Discovery
bbca7412-a950-11e3-9109-406186ea4fc5  System Discovery
daba56c8-73ec-11df-a475-002264764cea  Full and fast
```

Figure 32 : Création d'un scan openvas

10.1.3.2.5 Références

Python-nmap: <https://pypi.org/project/python-nmap/>

GitHub cheat sheet: <https://github.com/SecuProject/Pentest-Cheat-Sheet>

Pentest Tools Cheat Sheet: <https://highon.coffee/blog/penetration-testing-tools-cheat-sheet/>

Openvas official API : <https://docs.greenbone.net/GSM-Manual/gos-3.1/en/omp.html>

Openvas-lib: <https://pypi.org/project/openvas-lib/>

Openvas-automation: <https://github.com/cehkunal/Openvas-Automation>

10.1.3.3 Période du 14 mars au 30 mars

Avant d'expliquer les différentes étapes à travers lesquelles je suis passé durant ces deux dernières semaines, il est important de rappeler que des mesures de confinement ont été prises suite au coronavirus en Belgique.

Le travail qui suit a alors été réalisé à domicile en accord avec la direction de l'IESN et celle du DTM de marche, ce qui me permet ainsi de continuer mon stage par télétravail.

Les points majeurs explorés pendant ces deux semaines sont :

- La mise en place du modèle Client - Server
- La phase de scan avancé
- La phase d'exploitation et d'énumération

10.1.3.3.1 Modèle Client – Server

Avant que le confinement ne débute, Alexandre Marchal m'avait confié 2 nouveaux RPI à combiner au premier pour mes tests. Le deuxième RPI a donc été configuré similairement au précédent, mais dans une moindre mesure. En effet celui-ci nécessite que Python3 et ses modules ainsi que le serveur VNC pour l'accès à distance. Une fois configuré, l'accès à tous les RPI peut donc être effectué depuis un ordinateur fixe ou un portable. J'ai par la même occasion vérifié les performances fournies par le point d'accès wifi. La puce interne permet une connectivité allant de 5 à 10 mètres sans gros obstacle. L'obtention d'une antenne pourra ensuite permettre d'augmenter cette portée jusqu'à plusieurs dizaines de mètres. Une portée augmentée qui sera un énorme atout lors des tests du produit.

Ensuite, la procédure d'automatisation a été mise à jour afin de permettre le partage des tâches entre les RPI. L'utilisation des flags a été intégrée afin de permettre aux RPI de travailler conjointement sur le même scan. Les hôtes sont alors marqués d'un nombre entier compris entre 0 et 2 respectivement si le scan n'a pas encore été effectué, s'il s'est terminé avec succès ou si une erreur a été rencontrée.

Lors de ces modifications, des problèmes de synchronisation avec la base de données ont été rencontrés. Ces problèmes ont ainsi permis de mettre en avant des anomalies dans la structure de la BD qui ont ensuite été corrigées.

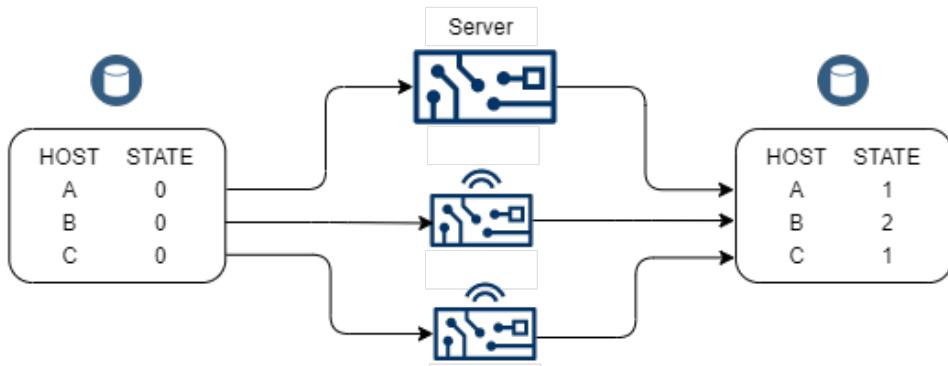


Figure 33: Modèle client - server

10.1.3.3.2 Phase de scan

Dans le but d'améliorer le scan de détection de vulnérabilité, qui se fait sous forme de requête à un serveur web, je m'étais précédemment intéressé à Openvas. Cependant, ce dernier présente une implémentation et une utilisation d'ID complexe qui rend son utilisation impossible. Il existe néanmoins deux autres API en python qui permettent d'interagir avec le Framework Openvas, mais l'utilisation des ID reste, ici aussi, un problème. Le scan de vulnérabilité est donc temporairement mis de côté et est actuellement implémenté avec le module Vulners de NMAP.

De plus, le script en Python a été segmenté afin d'externaliser les fonctions de scan et de les améliorer. Des informations supplémentaires sont dès lors récupérées, comme le dernier redémarrage de la machine ou encore le type de système selon la MAC adresse.

Les informations relatives à l'OS et sa version, avant récupérées lors du scan des vulnérabilités, sont désormais récoltées lors de ce scan. Ce qui permet au scan de vulnérabilité d'être allégé et de s'effectuer plus rapidement.

10.1.3.3.3 Phase d'exploitation et d'énumération

Afin de fournir une meilleure vision sur les informations qui pourraient fuiter lors d'une attaque, le scan va désormais effectuer une analyse avancée des services présents sur les machines. Cette analyse consiste en une suite de commandes propres à chaque service, parfois dépendantes, parfois indépendantes d'autres services. Ces services étant désormais référencés en base de données grâce au scan précédent, le script d'exploitation va pouvoir récupérer les informations relatives à chaque hôte afin d'exécuter uniquement les scans nécessaires.

Une suite de scripts propre aux protocoles fréquemment rencontrés a donc été élaborée, autant pour les systèmes d'exploitation Linux que Windows.

Cette suite provisoire ciblera alors les protocoles :

- FTP
- SSH
- SMTP
- DNS
- HTTP
- HTTPS
- LDAP
- Kerberos
- RPC
- Samba

```
root@spartan:/home/code2# python3 exploit.py
Doing NBT name scan for addresses from 192.168.0.0/24
IP address      NetBIOS Name    Server     User          MAC address
-----
192.168.0.28    DESKTOP-4FIE609  <server>  <unknown>    04:92:26:d9:ea:76
192.168.0.21    STRRR           <server>  <unknown>    ac:bc:32:88:42:3f
192.168.0.31    METASPLOITABLE  <server>  METASPLOITABLE  00:80:00:00:00:00
192.168.0.10    VOO-USB         <server>  VOO-USB       00:80:00:00:00:00
192.168.0.19    ROLAND-ASUS   <server>  <unknown>    24:8a:64:86:2f:19

--- FTP ---
Hydra v8.8 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-03-29 12:43:46
[DATA] max 16 tasks per 1 server, overall 16 tasks, 66 login tries, -5 tries per task
[DATA] attacking ftp://192.168.0.31:21/
[21][ftp] host: 192.168.0.31  login: anonymous  password: anonymous
[21][ftp] host: 192.168.0.31  login: ftp  password: bliuRR3
[21][ftp] host: 192.168.0.31  login: ftp  password: ftp
1 of 1 target successfully completed, 3 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-03-29 12:44:02

--- SSH ---
Hydra v8.8 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-03-29 12:44:02
[DATA] max 16 tasks per 1 server, overall 16 tasks, 131 login tries, ~9 tries per task
[DATA] attacking ssh://192.168.0.31:22/
1 of 1 target completed, 0 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-03-29 12:44:05

--- SMTP ---
Starting smtp-user-enum v1.2 ( http://pentestmonkey.net/tools/smtp-user-enum )

-----
|           Scan Information           |
-----

Mode ..... VRFY
Worker Processes ..... 5
Usernames file ..... /usr/share/seclists/Usernames/cirt-default-usernames.txt
Target count ..... 1
Username count ..... 825
Target TCP port ..... 25
Query timeout ..... 5 secs
Target domain .....
```

Figure 34: Script d'exploitation

En plus de ceux-ci, d'autres scripts d'énumération plus généraux (ex. enum4linux) viennent se rajouter dans le but de fournir des informations supplémentaires ou pour compléter celles déjà récoltées. Celles qui seront récoltées, dans un premier temps, sont celles qui pourraient amener l'infrastructure à être compromise si elles venaient à fuiter.

Parmi celles-ci se trouvent alors les informations relatives :

- aux machines présentes sur le réseau, leur nom d'hôte, utilisateurs, mac adresse, et système d'exploitation
- aux noms d'utilisateurs sur le système
- aux enregistrements DNS
- aux répertoires du serveur web ainsi que les failles présumées
- aux politiques de mots de passe
- aux répertoires partagés sur le réseau
- à la structure du réseau
- aux noms de domaine (si les machines se trouvent dans un AD)

Une fois ces dernières récupérées, les données qui concernent les utilisateurs et leur mot de passe sont synchronisées en base de données. Les RPI pourront ainsi utiliser le dictionnaire nouvellement créé et propre à l'entreprise afin d'effectuer des attaques de type brute force.

10.1.3.3.4 Référence

Python-Nmap: <https://pypi.org/project/python-nmap/>

GitHub cheat sheet: <https://github.com/SecuProject/Pentest-Cheat-Sheet>

Pentest Tools Cheat Sheet: <https://highon.coffee/blog/penetration-testing-tools-cheat-sheet/>

Openvas official API : <https://docs.greenbone.net/GSM-Manual/gos-3.1/en/omp.html>

Openvas-lib : <https://pypi.org/project/openvas-lib/>

Openvas-automation : <https://github.com/cehkunal/Openvas-Automation>

10.1.3.4 Période du 1er avril au 17 avril

10.1.3.4.1 Mise en commun des différentes phases de la solution

Les outils et les scripts qui étaient jusqu'à présent exécutés individuellement, forment désormais un tout. En effet après avoir complété l'implémentation des flags de contrôle qui permettent aux scans de s'effectuer de manière optimale, les trois phases ont pu être implémentées l'une à la suite de l'autre.

Lors de l'exécution de la première phase, les multiples RPI se lient au scan en base de données et commencent la reconnaissance du réseau, un flag RECON_STATE est alors défini sur la valeur 1 qui signifie qu'il est en cours. Les informations ciblées sont ensuite récoltées et envoyées en BD afin d'être stockées et, une fois la reconnaissance finie, le flag est défini à 3.

Les hôtes du réseau désormais connus, c'est à la phase de *vulnerability assessment* d'être exécutée. Cette phase garde un système de flag identique à la première et permet la récolte des informations comme les versions des services et les CVE rencontrées. Ces CVE sont ensuite comparées et liées à celles contenues en base de données. Ces dernières ont été importées depuis les sources mises à disposition par le NIST comme expliqué dans les rapports précédents. L'import de ces vulnérabilités avait d'ailleurs été problématique au début du projet, et il semble que ces problèmes soient survenus à cause de caractères mal encodés dans les descriptions des CVE ainsi qu'à la taille de celles-ci. Le problème a donc été résolu grâce à l'utilisation de fonction d'encodage. Chacune des descriptions est dès lors codée en base64 afin d'éliminer les caractères non désirés. Suite à ça, la totalité des vulnérabilités a pu être importée contre seulement 70% les fois précédentes.

The screenshot shows the MySQL Workbench interface with the 'VULN' table selected. The table contains 127155 rows, with the first few rows visible in the preview pane:

	VULN_ID	SEVERITY	COMPLEXITY	BASE_SCORE	DESCRIPTION
1	CVE-1999-0001	MEDIUM	LOW	5.0	aXBfaW5wdXQuYyBpbjBCU0QIZGVyaXIZCBUQ1AvSVAgwTwbG...
108	CVE-1999-0002	HIGH	LOW	10.0	QnVmZmVyl097ZXJmbG93GlueE5GLuyBlb3VuGQgZ2iZXMgc...
132	CVE-1999-0003	HIGH	LOW	10.0	RXH1YzV0ZS1bj21YW5cybchyBy2500l1ZpYSBdWZmXlgb3...
144	CVE-1999-0004	MEDIUM	LOW	5.0	TUINRSBidWZmZXlgb3ZcmZsb3cgaV4gZW1haWwgY2xpZW50c...
156	CVE-1999-0005	HIGH	LOW	10.0	QXJiaRyyXX51GNvbWVhbQzXHh3V3daVuIhZpYsTUFQ1o...
168	CVE-1999-0006	HIGH	LOW	10.0	QnVmZmVyl092ZXJmbG93GlueFPUjCBzzXJzJlGJc2VklKG...
180	CVE-1999-0007	MEDIUM	LOW	5.0	SW5mb3JYKRp24g2JujsBTU0vZTWSjchwdGVkhINC3npb2...
202	CVE-1999-0008	HIGH	LOW	10.0	QnVmZmVyl092ZXJmbG93Glue5JUysfGlufN1tdzIJuwy...
204	CVE-1999-0009	HIGH	LOW	10.0	SW52ZXJzZ5BxdWVyeSBidWZmZXlgb3ZcmZsb3cgaW4gQkIOR...
216	CVE-1999-0010	MEDIUM	LOW	5.0	RGVuawFsiGmifNlcnZpY2UgdnsbmvYYUjbG0leSBpbBCSU...
228	CVE-1999-0011	HIGH	LOW	10.0	RGVuawFsiGmifNlcnZpY2UgdnsbmvYYUjbG0leawVziGlUE...
240	CVE-1999-0012	MEDIUM	LOW	5.0	U29tZSB3ZlgC2VdmVycb1bmRlcBNwNy3NzNqQz2luZG...
250	CVE-1999-0013	HIGH	LOW	7.5	U3RvbGVigNyZWRlbnpYwzIGzy20gU1NIGNsaWwdHMgdm...
251	CVE-1999-0014	HIGH	LOW	7.2	VW5hdXRob3JpemVkiHByaXZpbGVnZWQgYWNjZXNzIgyljGRibm...
252	CVE-1999-0015	MEDIUM	LOW	5.0	VGhcmRybAgSVAgZGVuaWFsiGmifNlcnZpY2Uu...
253	CVE-1999-0016	MEDIUM	LOW	5.0	TGFiuZCBJUCBKZw5pYWwgb2Ygc2VydmiJzS4=
254	CVE-1999-0017	HIGH	LOW	7.5	RIRQHNCnZlcmMgY2UlGFsfob931GFulGF0dGfJa2VylHRvIG...
255	CVE-1999-0018	HIGH	LOW	10.0	QnVmZmVyl092ZXJmbG93GlueHN0YXRkiGsfob93Yybzb30H...
256	CVE-1999-0019	MEDIUM	LOW	5.0	RGVszXRiIgNyZWF0ZSBhIGZpbGUgdmlhJwYy5zdGf0ZC...

Figure 35 : Table des CVE importées dans la BD

Quant à la troisième phase, celle d'exploitation, un système de normalisation des inputs a été créé afin d'interpréter correctement chacun des résultats renvoyés par les outils. Grâce à celui-ci, chacun des modules d'exploitation peut être facilement amélioré ou modifié. Chaque module est en effet composé d'un appel à un outil installé sur le système, le script récupère ensuite l'output et la stocke avec la commande et le timestamp dans un tableau qui est ensuite renvoyé à la fonction principale. Ce fonctionnement permet ainsi d'ajouter une multitude d'outils au projet sans avoir à modifier la structure. Il suffit d'introduire la commande à effectuer dans un script et d'annoncer la condition de son appel dans le script principal.

10.1.3.4.2 Développement du Dashboard

L'objectif de ce Dashboard est la représentation rapide et compréhensible de l'avancement d'un scan et de son résultat. Une maquette des différentes pages web a ainsi été réalisée afin de réfléchir au positionnement et à l'ordre des informations à afficher. Sur chacune des pages, on retrouve dès lors, sous forme de boîte, les informations relatives aux scans et aux hôtes découverts durant la procédure. La quasi-totalité des informations récoltées précédemment et stockées en base de données y est affichée de manière dynamique.

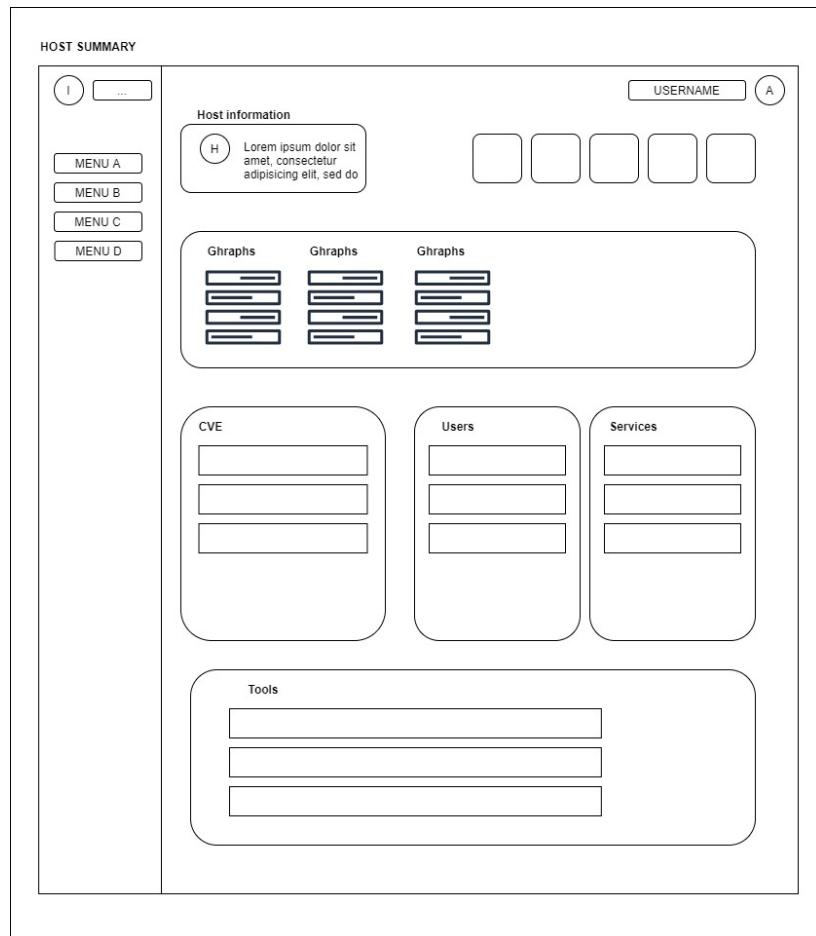


Figure 36 : Maquette du Dashboard

Pour cela, c'est PHP et le module MySQLi qui sont utilisés afin de communiquer avec la base de données. L'utilisateur utilisé ici est identique à celui des scripts, cependant afin de sécuriser au mieux la base de données, un utilisateur doté uniquement du droit en lecture sera créé. De manière similaire au module python Mysql-connector, MySQLi permet de définir une requête SQL avec des variables intégrées, d'établir la connexion avec la BD et d'ensuite exécuter la requête. Le résultat reçu sous forme de tableau est ensuite formaté afin de pouvoir l'afficher correctement sur la page web.

De plus, afin de faciliter son développement, le site est conçu avec Bootstrap, une librairie de composant en HTML, CSS et JavaScript qui permet au site d'être responsive et dynamique. Celui-ci permet ici de construire notre page web sous forme de blocs en définissant des lignes et des colonnes, de sorte à pouvoir disposer les informations de manière similaire aux cases d'un tableau.

La première page propose alors un compte rendu des derniers scans effectués et des clients connectés. Les informations relatives à ceux-ci, telles que les timestamps de début et de fin de scan, le réseau ciblé et les données qui permettent d'identifier les clients y sont alors présentés. Outre cela, une barre de progrès permet de représenter l'avancement des scans en cours.

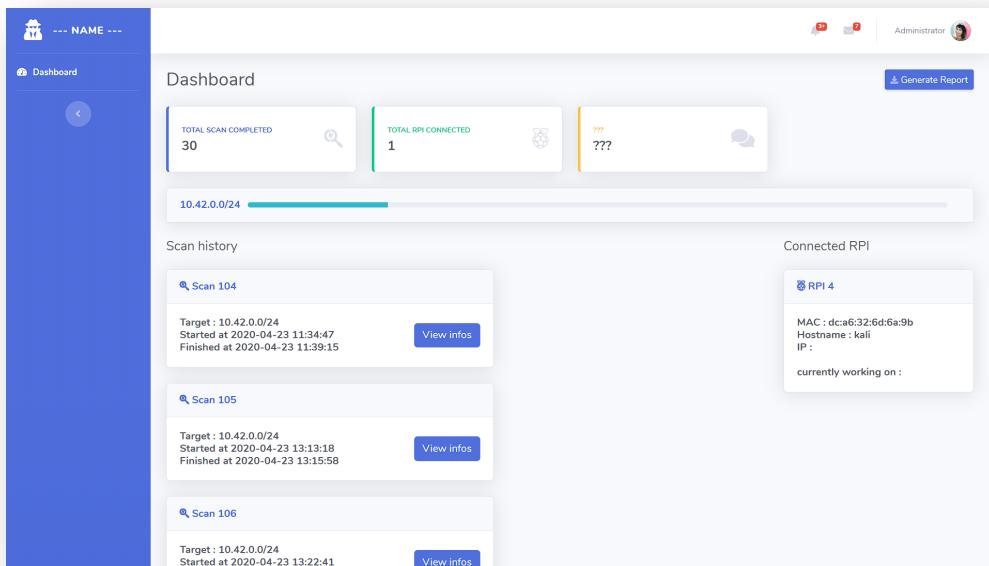


Figure 37 : Dashboard index

La seconde page est celle qui vise à apporter les informations relatives à chaque scan effectué. En plus des données relatives au scan, une boîte permet de représenter les hôtes découverts, et d'autres, plus petites, affichent le taux de vulnérabilité et le nombre total d'hôtes. Par la suite, un ou plusieurs graphiques viendront représenter le taux de vulnérabilité par hôte ainsi que leur criticité selon différents critères.

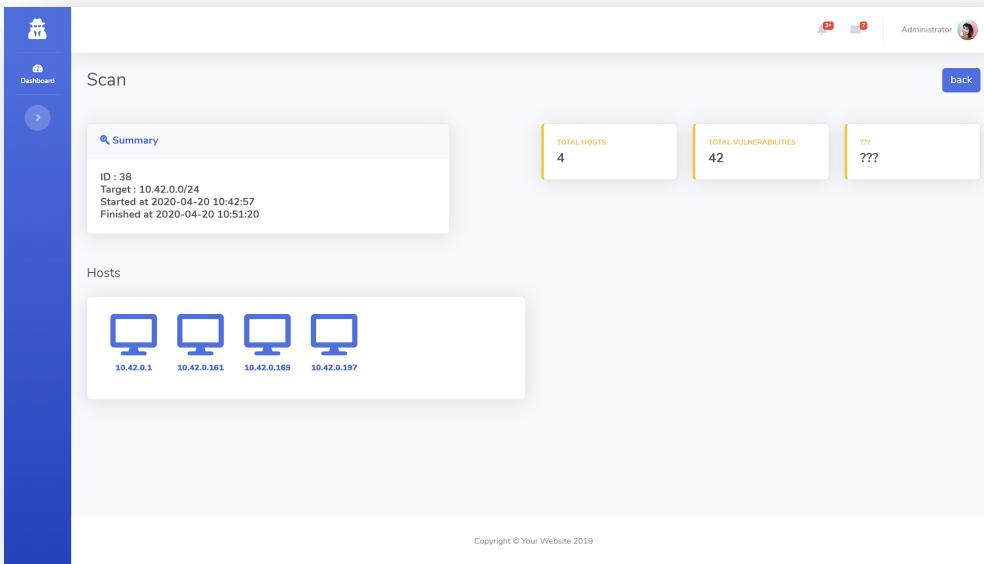


Figure 38 : Dashboard Partie Scan

La dernière page développée pour le moment est celle qui concerne directement chaque hôte scanné. Les informations relatives à son système d'exploitation, vendeur, IP, MAC et dernier redémarrage sont affichées en tant que résumé. Ensuite, plusieurs tableaux viennent présenter les CVE détectées sur chaque système, les utilisateurs découverts ainsi que chacun des services et leurs versions. Ici aussi, des graphiques viendront représenter la situation de l'hôte vis-à-vis des vulnérabilités qui auront été découvertes. Finalement, c'est le résultat de chacun des outils d'exploitation testés qui est affiché avec sa commande et son timestamp.

The screenshot shows the 'Host' section of the dashboard. The left sidebar has an icon for Host. The main area has a header 'Host' and a 'Summary' card. The summary card displays:

- ID : 272
- SCAN ID : 133
- IP : 10.42.0.161
- OS : Linux 2.6.9 - 2.6.33
- Vendor : Apple
- Last Boot : Thu Dec 13 13:04:44 2018

Below the summary are three tables: 'CVE Detected', 'Users', and 'Services'. The 'CVE Detected' table lists several CVE IDs. The 'Users' table lists system users like root, daemon, bin, games, lp, mail, man, news, nobody, and nobody. The 'Services' table lists various ports and their corresponding services and versions.

Port	Service	Version
21	ftp	vsftpd 2.3.4
22	ssh	OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23	telnet	Linux telnetd
25	smtp	Postfix smtpd
53	domain	ISC BIND 9.4.2
80	http	Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111	rpcbind	
139	netbios-ssn	
445	microsoft-ds	
512	exec	

Figure 39 : Dashboard Partie Hôte

10.1.3.4.3 Références

Bootstrap : <https://getbootstrap.com/>

Template Dashboard : <https://startbootstrap.com/themes/sb-admin-2/>

PHP MySQLi : <https://www.php.net/manual/fr/book.mysql.php>

10.1.3.5 Période du 18 avril au 22 mai (Fin du stage)

10.1.3.5.1 Amélioration et ajout de fonctionnalités au Dashboard

Au cours des trois dernières semaines, le projet a subi des tests ainsi que des améliorations afin de vérifier son fonctionnement et de l'améliorer au possible.

Ainsi, le *dashboard* a subi une refonte et différentes fonctionnalités y ont été rajoutées comme l'ajout d'une page permettant la comparaison entre plusieurs scans ou encore l'affichage de plusieurs graphiques afin d'augmenter la lisibilité des informations et de la vulnérabilité présentes sur les hôtes.

Finalement le projet a été testé plusieurs sur plusieurs réseaux afin de vérifier son bon fonctionnement. Quelque problème subsiste toujours, notamment lors du scan de certains services d'équipement réseau plus spécifique comme des CPL ou des imprimantes.

10.1.3.5.2 Rédaction de la documentation et du TFE

La fin de mon stage fut ensuite consacrée à l'écriture de mon TFE ainsi qu'à la rédaction de la documentation de mon projet.

Pour la documentation, une mise en commun des différents documents écrits au cours du stage a permis de construire un document pour chaque axe du projet.

Parmi ces trois documents :

- Le premier qui explique et détaille les commandes à exécuter afin de réaliser l'installation du système d'exploitation, des services de base de données et des accès à distance.
- Le second détaille la structure des codes rédigés où se trouvent les différentes parties du code ainsi que des annotations afin d'en expliquer le raisonnement.
- Le troisième reprend les différentes pages conçues pour le *dashboard*. Chacune des pages y est alors détaillée et la structure de la page ainsi que les informations qu'on y retrouve sont décrites.

L'écriture du TFE, quant à elle, fut plus complexe dû au manque d'habitude quant à la rédaction d'un tel rapport et de pratique dans cette activité.

10.1.4 Programme ON STAGE

Ci – joint, les résultats du programme ON STAGE qu'il nous a été demandé de compléter afin de mettre en évidence nos soft skills. Ce programme a été complété une fois avant le début du stage, une fois en fin de stage et finalement une fois par notre maître de stage.

	Évaluation (sur 4)	
	Test 1	Test 2
Idées et opportunités		
1. Déceler des opportunités	3.75	2.50
2. Avoir de la créativité	2.80	2.60
3. Stimuler sa vision	3.67	2.33
4. Valoriser les idées	3.50	2.50
5. Penser de manière éthique et durable	2.50	2.00
Ressources		
6. Conscience de soi et auto-efficacité	2.75	1.50
7. Motivation et persévérance	3.20	2.00
8. Mobiliser les ressources	3.00	2.50
9. Culture financière et économique	1.75	3.00
10. Mobiliser les autres	3.75	2.25
En action		
11. Prendre de l'initiative	4.00	3.00
12. Planifier et gérer	3.83	3.00
13. Gérer l'incertitude, l'ambiguïté et les risques	3.67	2.00
14. Travailler avec les autres	3.67	3.33
15. Apprendre de ses expériences	4.00	2.33

Figure 40 : Résultats ON STAGE personnels

	Évaluation	
	Test 1	Maitre
Idées et opportunités		
1. Déceler des opportunités	2.50	3.33
2. Avoir de la créativité	2.60	3.25
3. Stimuler sa vision	2.33	3.00
4. Valoriser les idées	2.50	3.00
5. Penser de manière éthique et durable	2.00	3.33
Ressources		
6. Conscience de soi et auto-efficacité	1.50	3.50
7. Motivation et persévérance	2.00	3.60
8. Mobiliser les ressources	2.50	N/A
9. Culture financière et économique	0.75	N/A
10. Mobiliser les autres	2.25	N/A
En action		
11. Prendre de l'initiative	3.00	3.50
12. Planifier et gérer	3.00	3.50
13. Gérer l'incertitude, l'ambiguïté et les risques	2.00	N/A
14. Travailler avec les autres	3.33	N/A
15. Apprendre de ses expériences	2.33	4.00

Figure 41 : Résultats ON STAGE avec le maître de stage

10.1.5 Organisation du travail (Gantt)

En début de stage, il est demandé de réaliser un diagramme de Gantt afin d'organiser notre travail pour les quinze semaines de stage.

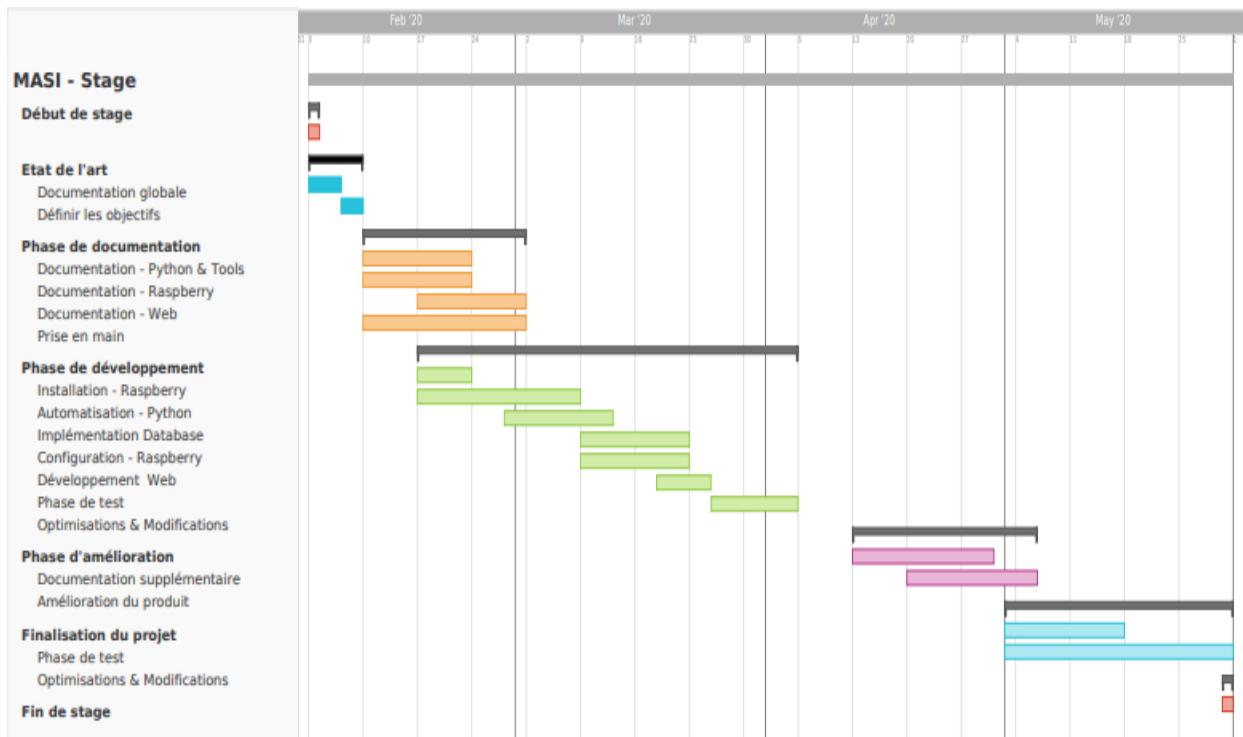


Figure 16 : Diagramme de Gantt

Le diagramme ci-dessus [figure 3] reprend la description des cinq phases que je prévois pour le déroulement du stage :

1. Un état de l'art des solutions similaires existantes
2. La réalisation d'une recherche de documentations sur les technologies déployées
3. Le développement du cœur principal du projet
4. L'amélioration du produit
5. La réalisation d'une période de tests et d'optimisation

Cet outil, largement utilisé dans la gestion de projet, offre une meilleure visibilité des tâches accomplies et de celles qui doivent l'être. Il permet de définir la durée envisagée de chacune ainsi que les interactions entre celles-ci. Il offre ainsi une vision claire de l'agencement du projet et de l'avancement de celui-ci tout au long de son déroulement.

10.1.5.1 Gantt Post stage

Au cours du stage, la réalisation d'un second diagramme de Gantt a été réalisée, dans le but de mettre en évidence les différences avec le premier et de vérifier si nos estimations de temps étaient correctes.

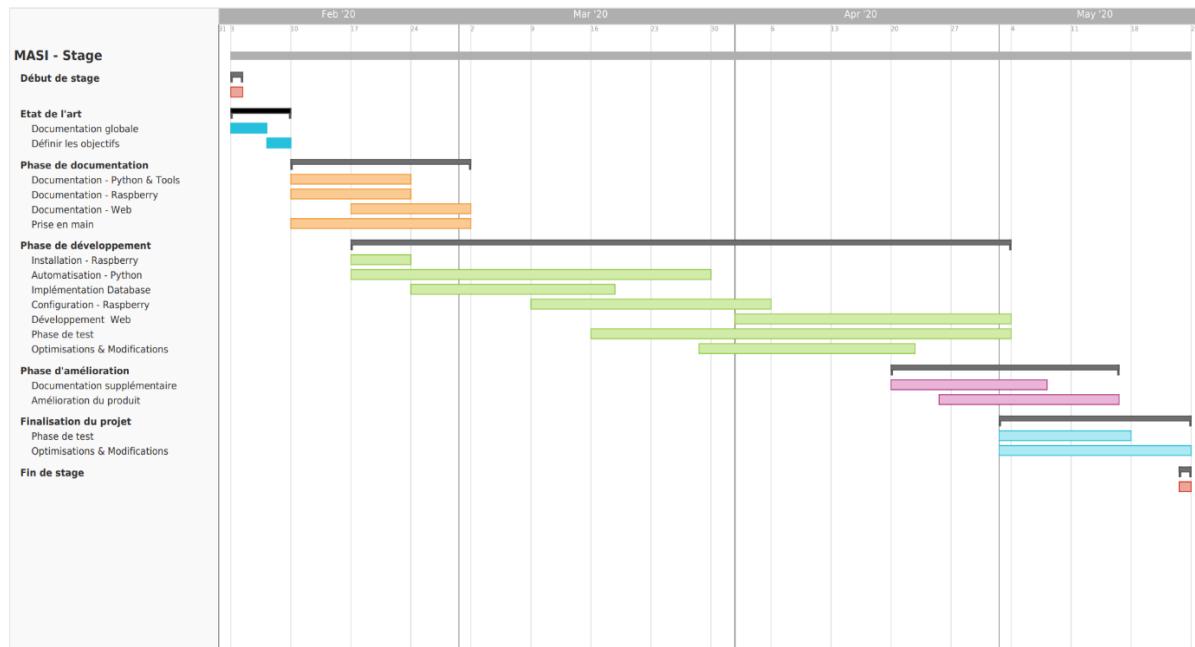


Figure 17 : Diagramme de Gantt post-stage

Ainsi, une fois le stage terminé, on peut voir sur ce diagramme que c'est principalement les tâches de la section développement qui ont subi des modifications, les autres phases quant à elles ont été correctement respectées.

Le diagramme permet alors de remarquer que les tâches qui ont été en quelque sorte sous-estimées étaient celles correspondant au développement du processus d'automatisation, de la conception de la base de données et celle du développement web.

En effet, pour chacune de ces trois phases, plusieurs versions ont été réalisées afin d'affiner le travail réalisé. Le processus d'automatisation a été optimisé en termes de performances, mais aussi en termes d'exécution d'outils. La conception de la base de données a, quant à elle, été revue plusieurs fois afin de s'assurer des informations à garder et de la façon dont allaient interagir les données entre les différentes tables. Le dashboard, lui, a subi une refonte graphique et fonctionnelle afin de rendre son utilisation plus abordable et lisible.

10.1.6 Evaluation du stage

10.1.6.1 Difficultés rencontrées

La première difficulté, et probablement la plus importante, fut l'apprentissage d'un ensemble d'outils et celui de compétences encore non explorées et pourtant indissociables de la cybersécurité.

Ainsi, au cours du projet plusieurs outils et *frameworks* ont été abordés afin d'être introduits dans le projet. Certains ont demandé beaucoup de temps et de documentation, mais finalement ne convenaient pas à la solution, parfois simplement à cause d'un détail ou une fonctionnalité incompatible. Parmi ces incompatibilités sont comprises celles purement techniques comme la lenteur d'exécution du Metasploit Framework qui est inévitable ou encore l'intégration de l'API OpenVAS qui a été abandonnée à cause de son utilisation d'ID générées aléatoirement.

Ensuite, malgré les connaissances apprises au cours de mon cursus, notamment en matière de base de données et de programmation, une documentation et des cours supplémentaires ont été nécessaires afin d'apporter plus de compréhension aux sujets abordés. Cet apprentissage plus approfondi a en revanche démontré l'intrication entre les différentes compétences développées au cours de ce stage.

Finalement, la conception d'un projet ne reposant sur aucune base préexistante a été un défi à relever. Il n'est en effet pas évident de réfléchir à chaque aspect à aborder et à explorer de manière objective et équilibrée, de juger et d'analyser quelles sont les technologies les plus aptes à être intégrées dans ce projet et de réfléchir en permanence à la possibilité d'ajouter des fonctionnalités qui permettront à l'outil d'évoluer.

10.1.6.2 Points forts

Les points forts quant à eux rejoignent en quelque sorte les difficultés abordées ci-dessus. Ce stage a ainsi permis de mettre en application plusieurs compétences apprises au cours de mon cursus, mais aussi d'en approfondir certaines qui m'ont permis d'aborder le projet sous plusieurs angles et de le faire aboutir. Les compétences mises en commun concernent les cours de principe de programmation et de développement, ceux de développement web et l'utilisation de la stack LAMP ainsi que les cours de base de données, de sécurité offensive et tous ceux ayant pour but l'apprentissage du système d'exploitation Linux. Ce projet a donc été une opportunité pour apprendre à concevoir des interactions entre toutes ces compétences.

Mais celui-ci a aussi mis en évidence le fait que malgré que beaucoup de technologies aient été abordées au cours de mon cursus, il en reste encore bien d'autres et que celles-ci ont chacune leurs avantages comme leurs inconvénients, leurs compatibilités et incompatibilités et quelles doivent être analysées attentivement, car les faire communiquer et interagir entre elles représente souvent un défi.

10.1.6.3 Relation avec l'équipe du FoRS et déroulement du stage

Dès le début du stage, l'entièreté de l'équipe fut très accueillante, et pendant les premières semaines du stage le fonctionnement était le suivant, nous travaillions entre stagiaires sur nos projets respectifs et l'équipe du FoRS était à notre disposition afin de répondre à nos interrogations.

Chaque vendredi, un débriefing en compagnie de l'équipe était organisé afin de discuter des tâches effectuées au cours de la semaine et de vérifier le bon déroulement du projet.

Ensuite, dus à la situation de confinement amenée par le Covid-19, des moyens de communication ont été mis en place entre l'équipe et les stagiaires afin de poursuivre le stage par télétravail. Ceux-ci ont permis d'effectuer quotidiennement des rapports, nous permettant de garder un contact avec l'équipe et d'avoir un suivi sur l'avancement de nos projets respectifs. Cette période a permis de développer des compétences comme l'autonomie et la gestion d'un emploi du temps.

Le stage dans son ensemble s'est donc très bien déroulé. La bonne ambiance et le professionnalisme dégagé par l'ensemble des membres et par les stagiaires se sont ensuite retrouvés lors de nos conversations qui ont permis plus d'une fois de répondre à nos interrogations.

10.1.7 Conclusion

Afin de conclure ces quinze semaines, ce stage a permis de mobiliser l'ensemble de mes compétences comme celles de la gestion et de la configuration d'un système d'exploitation, du développement et de l'analyse de processus d'automatisation, mais également d'en développer et d'en perfectionner de nouvelles comme la conception d'un site web, l'utilisation des langages de programmation et bien évidemment la gestion d'un projet.

Mais plus important, il m'a offert la possibilité d'effectuer un premier pas dans le monde professionnel aux côtés d'une équipe jeune et enthousiaste. Une équipe qui pousse, à travers les projets qu'ils entreprennent, à faire preuve d'imagination et d'autonomie, et ce, sans cesser de développer et d'améliorer leurs connaissances et compétences dans le milieu technique.

Je suis heureux d'avoir pu réaliser ce stage qui m'a permis d'en apprendre plus sur moi-même et pour les moments agréables de partage et d'apprentissage.

10.2 Documentation

10.2.1 Documentation de l'architecture

Installation du système

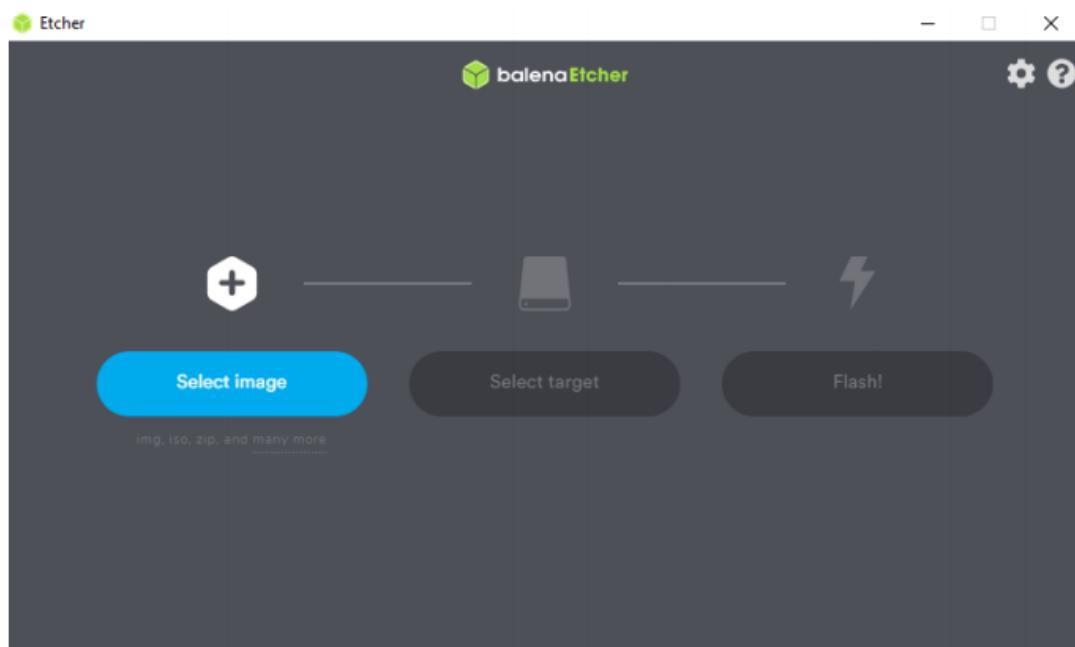
Téléchargement de l'OS

Download - <https://www.offensive-security.com/kali-linux-arm-images/>

Version utilisée : Kali Linux 2020.1a rpi3-nexmon-64
Dernière version en date : Kali Linux 2020.2

Flash de la carte sd

Download - <https://www.balena.io/etcher/>



Configuration du système

Paramètres de langage et du clavier

```
setxkbmap be
dpkg-reconfigure tzdata
dpkg-reconfigure locales
```

Création de l'utilisateur non privilégié

```
adduser <NOM>
```

Configuration de l'auto-login

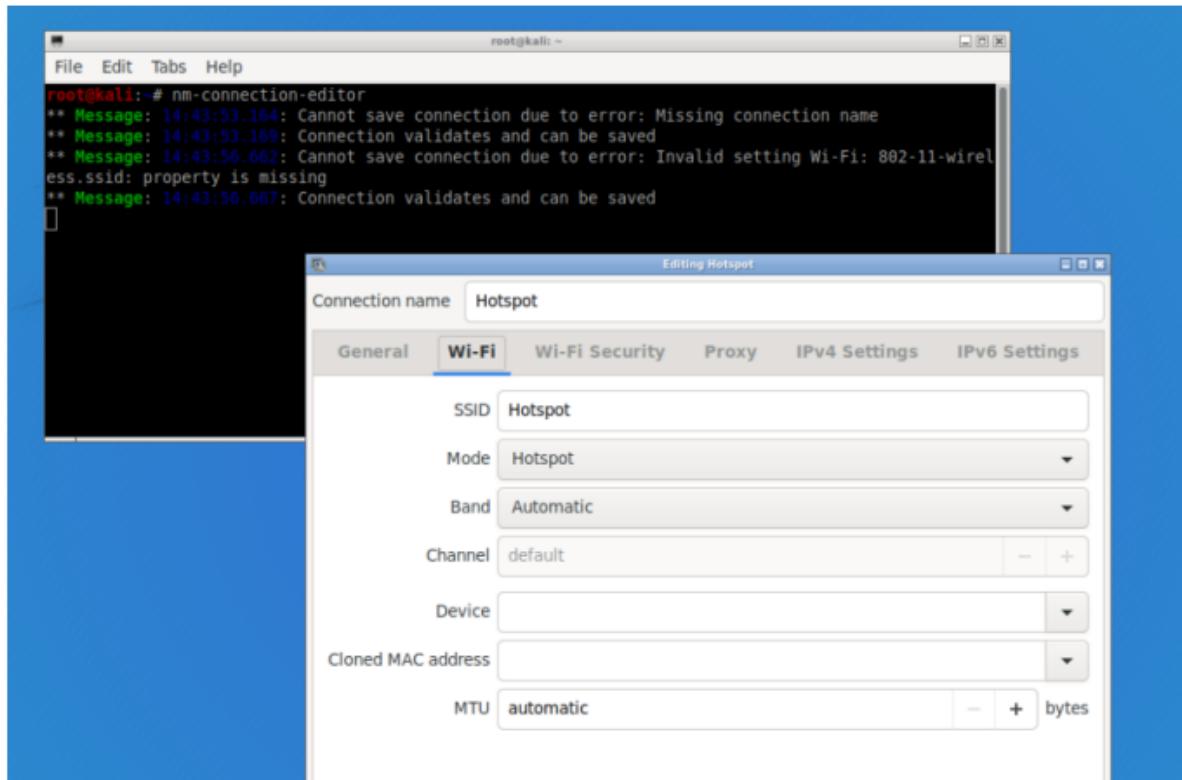
```
nano /etc/lightdm/lightdm.conf
```

```
uncomment  
autologin-user=<NOM>  
autologin-timeout=0
```

[LightDM config - https://wiki.ubuntu.com/LightDM](https://wiki.ubuntu.com/LightDM)

Création du réseau hotspot

```
nm-connection-editor
```



[man page - nm-connection-editor](#)

Setup des accès à distance

Config SSH

```
root@kali:~# ssh -V
OpenSSH_8.1p1 Debian-1, OpenSSL 1.1.1d 10 Sep 2019
```

```
sudo nano /etc/ssh/sshd_config

PermitRootLogin no
Port 2222
ListenAddress <NETWORK>
ClientAliveInterval 360
ClientAliveCountMax 0
PermitEmptyPasswords no
MaxAuthTries 3
```

```
apt-get install fail2ban

nano /etc/fail2ban/jail.local

# [sshd]
enabled = true

# "bantime" is the number of seconds that a host is banned.
bantime  = <NUMBER_IN_SECONDS>

# A host is banned if it has generated "maxretry" during the last
"findtime"
# seconds.
findtime = <NUMBER_IN_SECONDS>
maxretry = 3
```

Config - fail2ban

Download Terminus Client - <https://terminus.com/>



Config VNC

```
sudo apt-get install tightvncserver
sudo apt-get install lxde
```

```
cd /home/user/.vnc
nano xstartup

+ /usr/bin/startlxde
```

Grey Screen comes on connecting to VNC server - Youtube

```
vncserver :1 -geometry 1920x1080
netstat -tulpn ( debug )
```

Download VNCviewer Client - <https://www.realvnc.com/en/connect/download/viewer/>



Installation et configuration des services

Installation

```
apt-get install mariadb-server mariadb-client  
apt-get install mysql-server  
apt-get install apache2  
apt-get install phpmyadmin  
  
systemctl enable apache2 mariadb phpmyadmin
```

```
root@kali:~# mysql --version  
mysql Ver 15.1 Distrib 10.3.22-MariaDB, for debian-linux-gnu (aarch64)  
using readline 5.2
```

Apache/2.4.41

```
root@kali:/etc/ssh# php --version  
PHP 7.3.15-3 (cli) (built: Feb 23 2020 07:15:44) ( NTS )  
Copyright (c) 1997-2018 The PHP Group  
Zend Engine v3.3.15, Copyright (c) 1998-2018 Zend Technologies  
    with Zend OPcache v7.3.15-3, Copyright (c) 1999-2018, by Zend  
Technologies
```

Configuration de la base de données

```
mariadb-secure-installation ( commande pour la sécurisation du service )
```

```
MariaDB [(none)]> CREATE DATABASE <DB_NAME>;  
MariaDB [(none)]> CREATE USER '<USER>'@'%' IDENTIFIED BY 'password';  
MariaDB [(none)]> GRANT ALL PRIVILEGES ON <DB_NAME>.* TO '<USER>'@'%';  
MariaDB [(none)]> FLUSH PRIVILEGES;  
  
[CHECK]  
MariaDB [(none)]> SELECT user, host FROM mysql.user;
```

```
nano /etc/mysql/mariadb.conf.d/50-server.cnf

port 30306 ( custom port )
bind-address = <NETWORK> ( ici 10.42.0.1 )
```

[Création des certificats]

```
$ mkdir /etc/mysql/ssl & cd /etc/mysql/ssl

$ sudo openssl genrsa 2048 > ca-key.pem
$ sudo openssl req -new -x509 -nodes -days 365000 -key ca-key.pem -out ca-cert.pem
$ sudo openssl req -newkey rsa:2048 -days 365000 -nodes -keyout server-key.pem -out server-req.pem
$ sudo openssl rsa -in server-key.pem -out server-key.pem
$ sudo openssl x509 -req -in server-req.pem -days 365000 -CA ca-cert.pem -CAkey ca-key.pem -set_serial 01 -out server-cert.pem
$ sudo openssl req -newkey rsa:2048 -days 365000 -nodes -keyout client-key.pem -out client-req.pem
$ sudo openssl rsa -in client-key.pem -out client-key.pem

$ openssl verify -CAfile ca-cert.pem server-cert.pem client-cert.pem
```

[SERVER]

```
$ sudo vi /etc/mysql/mariadb.conf.d/50-server.cnf
    ssl-ca=/etc/mysql/ssl/ca-cert.pem
    ssl-cert=/etc/mysql/ssl/server-cert.pem
    ssl-key=/etc/mysql/ssl/server-key.pem
    tls_version = TLSv1.2

chown mysql:root /etc/mysql/ssl/
systemctl restart mysql
```

L'envoi des fichiers peut être effectué par FTP ou grâce à la commande suivante (dans le dir contenant les fichiers) :
python3 -m http.server

[CLIENT]

```
$ sudo vi /etc/mysql/mariadb.conf.d/50-mysql-clients.cnf
  ssl-ca=/etc/mysql/ssl/ca-cert.pem
  ssl-cert=/etc/mysql/ssl/client-cert.pem
  ssl-key=/etc/mysql/ssl/client-key.pem
```

[Verification]

```
$ mysql -u root -h 192.168.0.30
MariaDB [(none)]> SHOW VARIABLES LIKE '%ssl%'; ( have_openssl & have_ssl
enabled )
MariaDB [(none)]> status;
```

SSL/TLS config

Installation des outils

Python & Pip

```
root@kali:~# python3 --version
Python 3.7.7
```

```
root@kali:~# pip3 --version
pip 18.1 from /usr/lib/python3/dist-packages/pip (python 3.7)
```

requirements.txt

```
pip3 install nmap
pip3 install mysql-connector-python
pip3 install getmac
pip3 install netifaces
pip3 install python-nmap
```

Installation des outils

Recursive gobuster - github

```
git clone https://github.com/epi052/recursive-gobuster.git
```

Kerbrute - github

```
git clone https://github.com/ropnop/kerbrute.git  
sudo apt-get install golang  
go build  
alias kerbrute=<DIR>/kerbrute"
```

Smtp-user-enum - github

```
git clone https://github.com/pentestmonkey/smtp-user-enum.git
```

Enum4Linux - github

```
git clone https://github.com/portcullislabs/enum4linux.git
```

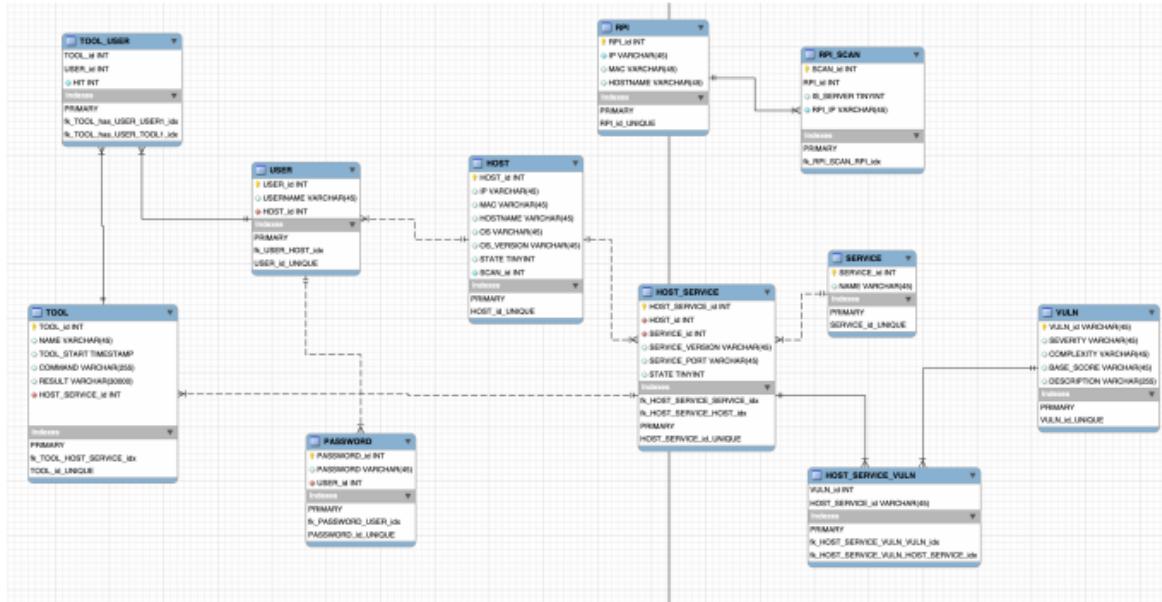
Autres outils (déjà présent sur Kali)

```
dig  
nikto  
hydra  
ldapsearch  
nbtscan  
rpcclient  
nmblookup  
smbclient
```

Penetration testing cheat sheet

Conception de la base de données

Download MySQL workbench - <https://www.mysql.com/products/workbench/>



```
-- phpMyAdmin SQL Dump
-- version 4.9.2deb1
-- https://www.phpmyadmin.net/
--
-- Hôte : localhost:3306
-- Généré le : mer. 20 mai 2020 à 16:11
-- Version du serveur : 10.3.22-MariaDB-1
-- Version de PHP : 7.3.15-3

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Base de données : `STAGE`
--
```

```

-- 
-- Structure de la table `HOST` 
-- 

CREATE TABLE `HOST` (
    `HOST_id` int(11) NOT NULL,
    `SCAN_id` int(11) NOT NULL,
    `IP` varchar(45) DEFAULT NULL,
    `MAC` varchar(100) DEFAULT NULL,
    `HOSTNAME` varchar(50) DEFAULT NULL,
    `OS` varchar(100) DEFAULT NULL,
    `VENDOR` varchar(100) DEFAULT NULL,
    `LAST_BOOT` varchar(100) DEFAULT NULL,
    `STATE` tinyint(4) NOT NULL DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

----- 

-- 
-- Structure de la table `HOST_SERVICE` 
-- 

CREATE TABLE `HOST_SERVICE` (
    `HOST_SERVICE_id` int(11) NOT NULL,
    `HOST_id` int(11) NOT NULL,
    `SERVICE_NAME` varchar(45) NOT NULL,
    `SERVICE_VERSION` varchar(45) DEFAULT NULL,
    `SERVICE_PORT` varchar(45) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

----- 

-- 
-- Structure de la table `HOST_SERVICE_VULN` 
-- 

CREATE TABLE `HOST_SERVICE_VULN` (
    `HOST_SERVICE_id` int(11) NOT NULL,
    `VULN_id` varchar(45) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

```

```

-- 
-- Structure de la table `PASSWORD` 

-- 

CREATE TABLE `PASSWORD` (
  `PASSWORD` varchar(45) NOT NULL,
  `HIT` int(11) NOT NULL DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- 
-- Structure de la table `RPI` 

-- 

CREATE TABLE `RPI` (
  `RPI_id` int(11) NOT NULL,
  `IP` varchar(45) NOT NULL,
  `MAC` varchar(45) NOT NULL,
  `HOSTNAME` varchar(45) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- 
-- Structure de la table `RPI_SCAN` 

-- 

CREATE TABLE `RPI_SCAN` (
  `RPI_id` int(11) NOT NULL,
  `SCAN_id` int(11) NOT NULL,
  `IS_CONNECTED` tinyint(4) NOT NULL DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- 
-- Structure de la table `SCAN` 

-- 

CREATE TABLE `SCAN` (
  `SCAN_id` int(11) NOT NULL,
  `SCAN_START` timestamp NULL DEFAULT NULL,
  `SCAN_END` timestamp NULL DEFAULT NULL,
  `SUBNET` varchar(45) DEFAULT NULL,
  `STATE` tinyint(4) NOT NULL DEFAULT 0,
  `RECON_STATE` tinyint(4) NOT NULL DEFAULT 0,
  `EXPLOIT_STATE` tinyint(4) NOT NULL DEFAULT 0,
  `VULN_STATE` tinyint(4) NOT NULL DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

```

```

-- 
-- Structure de la table `SERVICE` 

CREATE TABLE `SERVICE` (
    `NAME` varchar(45) NOT NULL,
    `HIT` int(11) NOT NULL DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-----


-- 
-- Structure de la table `TOOL` 

CREATE TABLE `TOOL` (
    `TOOL_id` int(11) NOT NULL,
    `HOST_id` int(11) NOT NULL,
    `NAME` varchar(100) NOT NULL,
    `COMMAND` varchar(500) NOT NULL,
    `RESULT` text DEFAULT NULL,
    `TOOL_START` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-----


-- 
-- Structure de la table `USER` 

CREATE TABLE `USER` (
    `USERNAME` varchar(45) NOT NULL,
    `HIT` int(11) NOT NULL DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-----


-- 
-- Structure de la table `VULN` 

CREATE TABLE `VULN` (
    `VULN_id` varchar(45) NOT NULL,
    `SEVERITY` varchar(45) NOT NULL DEFAULT 'UNKNOWN',
    `COMPLEXITY` varchar(45) NOT NULL DEFAULT 'UNKNOWN',
    `BASE_SCORE` varchar(45) NOT NULL DEFAULT 'UNKNOWN',
    `DESCRIPTION` varchar(10000) NOT NULL DEFAULT 'UNKNOWN'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

```

```

-- Index pour les tables déchargées
-- 

-- Index pour la table `HOST`
-- 
ALTER TABLE `HOST`
    ADD PRIMARY KEY (`HOST_id`),
    ADD KEY `fk_HOST_SCAN2` (`SCAN_id`);

-- Index pour la table `HOST_SERVICE`
-- 
ALTER TABLE `HOST_SERVICE`
    ADD PRIMARY KEY (`HOST_SERVICE_id`),
    ADD KEY `fk_HOST_has_SERVICE_HOST2` (`HOST_id`),
    ADD KEY `fk_HOST_SERVICE_SERVICE1` (`SERVICE_NAME`);

-- Index pour la table `HOST_SERVICE_VULN`
-- 
ALTER TABLE `HOST_SERVICE_VULN`
    ADD PRIMARY KEY (`HOST_SERVICE_id`, `VULN_id`),
    ADD KEY `fk_HOST_SERVICE_has_VULN_VULN1` (`VULN_id`);

-- Index pour la table `HOST_USER_PASSWORD`
-- 
ALTER TABLE `HOST_USER_PASSWORD`
    ADD PRIMARY KEY (`HOST_USER_PASSWORD_id`),
    ADD KEY `fk_HOST_USER_PASSWORD_HOST1` (`HOST_id`),
    ADD KEY `fk_HOST_USER_PASSWORD_USER1` (`USERNAME`),
    ADD KEY `fk_HOST_USER_PASSWORD_PASSWORD1` (`PASSWORD`);

-- Index pour la table `PASSWORD`
-- 
ALTER TABLE `PASSWORD`
    ADD PRIMARY KEY (`PASSWORD`);

-- Index pour la table `RPI`
-- 
ALTER TABLE `RPI`
    ADD PRIMARY KEY (`RPI_id`);

-- Index pour la table `RPI_SCAN`
-- 
ALTER TABLE `RPI_SCAN`
    ADD PRIMARY KEY (`RPI_id`, `SCAN_id`),
    ADD KEY `fk_RPI_has_SCAN_SCAN2` (`SCAN_id`);

```

```

-- Index pour la table `SCAN`
ALTER TABLE `SCAN`
ADD PRIMARY KEY (`SCAN_id`);

-- Index pour la table `SERVICE`
ALTER TABLE `SERVICE`
ADD PRIMARY KEY (`NAME`);

-- Index pour la table `TOOL`
ALTER TABLE `TOOL`
ADD PRIMARY KEY (`TOOL_id`),
ADD KEY `fk_TOOL_HOST1` (`HOST_id`);

-- Index pour la table `USER`
ALTER TABLE `USER`
ADD PRIMARY KEY (`USERNAME`);

-- Index pour la table `VULN`
ALTER TABLE `VULN`
ADD PRIMARY KEY (`VULN_id`);

-- AUTO_INCREMENT pour les tables déchargées

-- AUTO_INCREMENT pour la table `HOST`
ALTER TABLE `HOST`
MODIFY `HOST_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=101;

-- AUTO_INCREMENT pour la table `HOST_SERVICE`
ALTER TABLE `HOST_SERVICE`
MODIFY `HOST_SERVICE_id` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=419;

-- AUTO_INCREMENT pour la table `HOST_USER_PASSWORD`
ALTER TABLE `HOST_USER_PASSWORD`
MODIFY `HOST_USER_PASSWORD_id` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=169;

```

```

-- AUTO_INCREMENT pour la table `RPI`
--
ALTER TABLE `RPI`
    MODIFY `RPI_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

-- AUTO_INCREMENT pour la table `SCAN`
--
ALTER TABLE `SCAN`
    MODIFY `SCAN_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=29;

-- AUTO_INCREMENT pour la table `TOOL`
--
ALTER TABLE `TOOL`
    MODIFY `TOOL_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=357;

-- Contraintes pour les tables déchargées
--

-- Contraintes pour la table `HOST`
--
ALTER TABLE `HOST`
    ADD CONSTRAINT `fk_HOST_SCAN2` FOREIGN KEY (`SCAN_id`) REFERENCES `SCAN`(`SCAN_id`)
        ON DELETE NO ACTION ON UPDATE NO ACTION;

-- Contraintes pour la table `HOST_SERVICE`
--
ALTER TABLE `HOST_SERVICE`
    ADD CONSTRAINT `fk_HOST_SERVICE_SERVICE1` FOREIGN KEY (`SERVICE_NAME`)
        REFERENCES `SERVICE`(`NAME`) ON DELETE NO ACTION ON UPDATE NO ACTION,
    ADD CONSTRAINT `fk_HOST_has_SERVICE_HOST2` FOREIGN KEY (`HOST_id`)
        REFERENCES `HOST`(`HOST_id`) ON DELETE NO ACTION ON UPDATE NO ACTION;

-- Contraintes pour la table `HOST_SERVICE_VULN`
--
ALTER TABLE `HOST_SERVICE_VULN`
    ADD CONSTRAINT `fk_HOST_SERVICE_has_VULN_HOST_SERVICE1` FOREIGN KEY
        (`HOST_SERVICE_id`) REFERENCES `HOST_SERVICE`(`HOST_SERVICE_id`)
        ON DELETE NO ACTION ON UPDATE NO ACTION,
    ADD CONSTRAINT `fk_HOST_SERVICE_has_VULN_VULN1` FOREIGN KEY (`VULN_id`)
        REFERENCES `VULN`(`VULN_id`) ON DELETE NO ACTION ON UPDATE NO ACTION;

```

```

-- Constraintes pour la table `HOST_USER_PASSWORD`
--
ALTER TABLE `HOST_USER_PASSWORD`
    ADD CONSTRAINT `fk_HOST_USER_PASSWORD_HOST1` FOREIGN KEY (`HOST_id`)
REFERENCES `HOST` (`HOST_id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
    ADD CONSTRAINT `fk_HOST_USER_PASSWORD_PASSWORD1` FOREIGN KEY
(`PASSWORD`) REFERENCES `PASSWORD` (`PASSWORD`) ON DELETE NO ACTION ON
UPDATE NO ACTION,
    ADD CONSTRAINT `fk_HOST_USER_PASSWORD_USER1` FOREIGN KEY (`USERNAME`)
REFERENCES `USER` (`USERNAME`) ON DELETE NO ACTION ON UPDATE NO ACTION;

-- Constraintes pour la table `RPI_SCAN`
--
ALTER TABLE `RPI_SCAN`
    ADD CONSTRAINT `fk_RPI_has_SCAN_RPI1` FOREIGN KEY (`RPI_id`) REFERENCES
`RPI` (`RPI_id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
    ADD CONSTRAINT `fk_RPI_has_SCAN_SCAN2` FOREIGN KEY (`SCAN_id`)
REFERENCES `SCAN` (`SCAN_id`) ON DELETE NO ACTION ON UPDATE NO ACTION;

-- Constraintes pour la table `TOOL`
--
ALTER TABLE `TOOL`
    ADD CONSTRAINT `fk_TOOL_HOST1` FOREIGN KEY (`HOST_id`) REFERENCES `HOST`
(`HOST_id`) ON DELETE NO ACTION ON UPDATE NO ACTION;
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

10.2.2 Documentation du code

Serv.py

Structure

```
import netifaces as ni
import ipaddress
import nmap
import re
import mysql.connector
import getmac
from subprocess import PIPE, run
import socket
import sys
import scanning
import exploit.exploit as exp

db = mysql.connector.connect(
    host="10.42.0.1",
    user="bob",
    passwd="bob",
    database="STAGE"
)
cursor = db.cursor()

main_interface = 'wlan0'
self_interface = 'eth0'

> def get_hosts(subnet, scan_id, rpi_id):...
> def check_scan(addr, RPIaddr, subnet, hostname, mac):...
> def get_RPI_info(): # Return IP / self IP / Netmask / Subnet...
> def main():...

if __name__ == "__main__":
    # execute only if run as a script
    main()
```

main()

```
def main():

    #1 Récupère les informations du RPI / client
    addr, RPIaddr, subnet, hostname, mac = get_RPI_info()

    #2 Assigne le client à un scan ou le crée s'il n'existe pas
    scan_id, rpi_id, link_id = check_scan(addr, RPIaddr, subnet, hostname,
                                           mac)

    #3 Exécute le script de scanning
    scanning.main(scan_id)
```

```

#4 Exécute le script d'exploitation
exp.main(scan_id, subnet)

#5 Définit le scan à l'état fini
scan_update = "UPDATE SCAN SET STATE = '%s', SCAN_END =
CURRENT_TIMESTAMP WHERE SCAN_id ='%s';"
rpi_scan_update = "UPDATE RPI_SCAN SET IS_CONNECTED = '%s' WHERE
RPI_id = '%s' AND SCAN_id ='%s';"
try:
    cursor.execute(scan_update % ('3', scan_id))
    cursor.execute(rpi_scan_update % ('0', rpi_id, scan_id))
    db.commit()
except:
    print("Database Error On Scan Update")
    sys.exit(0)

db.close()

```

get_RPI_info()

```

def get_RPI_info(): # Return IP / self IP / Netmask / Subnet

    try :
        # Récupère IP & MASK de l'interface externe
        ni.ifaddresses(main_interface)
        addr = str(ni.ifaddresses(main_interface)[ni.AF_INET][0]['addr'])
        netmask = str(ni.ifaddresses(main_interface)[ni.AF_INET][0]
['netmask'])

        # Récupère le sous réseau
        addr_mask = addr + '/' + netmask
        subnet = str(ipaddress.ip_network(addr_mask, strict=False))
    except:
        print("no lan connection !!!")
        sys.exit(0)

    # Récupère IP & MASK de l'interface interne
    ni.ifaddresses(self_interface)
    RPIaddr = str(ni.ifaddresses(self_interface)[ni.AF_INET][0]['addr'])
    RPInetmask = str(ni.ifaddresses(self_interface)[ni.AF_INET][0]
['netmask'])

    # Récupère HOSTNAME
    hostname = str(socket.gethostname())

    # Récupère MAC adresse
    mac = str(getmac.get_mac_address())

    return addr, RPIaddr, subnet, hostname, mac

```

Check_scan()

```
def check_scan(addr, RPIaddr, subnet, hostname, mac):

    #1 Vérifie si un scan est déjà en cours dans le même sous-réseau
    scan_select = "SELECT SCAN_id FROM SCAN WHERE SCAN_END IS NULL AND
STATE = 1 AND SUBNET = '%s';"
    try:
        ...
    except:
        ...
    row = cursor.fetchone()

    #2 Création d'un scan s'il n'en existe pas déjà un
    if not row:
        print("Scan Creation !")
    ...
    else :
        scan_id = row[0]

    #3 Ajout du RPI / client en base de données
    rpi_insert = "INSERT INTO RPI (IP, MAC, HOSTNAME) SELECT '%s', '%s',
'%s' FROM DUAL WHERE NOT EXISTS( \
        SELECT 1 FROM RPI WHERE IP = '%s' AND MAC = '%s' AND HOSTNAME =
'%s') LIMIT 1;"
    rpi_select = "SELECT RPI_id FROM RPI WHERE IP = '%s' AND MAC = '%s'
AND HOSTNAME = '%s';"
    try:
        ...
    except:
        ...
    link_select = "SELECT RPI_id, SCAN_id FROM RPI_SCAN WHERE RPI_id =
'%s' AND SCAN_id = '%s';"
    try:
        ...
    except:
        ...
    row = cursor.fetchone()

    #4 Création du lien entre le scan et le client
    if not row:
        print("Link Creation")
        link_insert = "INSERT INTO RPI_SCAN (RPI_id, SCAN_id,
IS_CONNECTED) VALUES ('%s', '%s', '1');"
        try:
            ...
        except:
            ...
    #5 Récupération des hôtes sur le réseau
    get_hosts(subnet, scan_id, rpi_id)
```

```

    else :
        link_id = row[0]

    return scan_id, rpi_id, link_id

```

Get_hosts()

```

def get_hosts(subnet, scan_id, rpi_id):

    #1 Utilisation du module NMAP pour scanner le réseau ( simple scan
    ping )
    scanner = nmap.PortScanner()
    scanner.scan(hosts=subnet, arguments=' -sP')
    hosts = scanner.all_hosts()

    #2 Ajout des hôtes dans la base de données
    for host in hosts:
        host_insert = "INSERT INTO HOST (IP, SCAN_id) SELECT '%s', '%s'
FROM DUAL WHERE NOT EXISTS( \
            SELECT 1 FROM HOST WHERE IP = '%s' AND SCAN_id='%s') LIMIT 1;"
        try:
            cursor.execute(host_insert % (str(host), scan_id, str(host),
scan_id))
            db.commit()
        except:
            db.rollback()
            print("Database Error On Host Creation")
            sys.exit(0)

```

Scanning.py

Structure

```
import nmap
import re
import sys
import mysql.connector
from subprocess import PIPE, run
from datetime import datetime
import base64

db = mysql.connector.connect(...
cursor = db.cursor()

class Host:
    def __init__(self, host_id, scan_id, ip, mac, hostname, os, vendor, last_boot, state):...
    def get_service(self):...
    def cve_scan(self, service_list):...
    def parse_cve(self, report):...

    def create_hosts_instance(host):...
    def check_hosts(scan_id, start_state, process_state, error_state):...
    def run_scan(scan_id):...
    def main(scan_id):...
    if __name__ == "__main__":
        # execute only if run as a script
        main()
```

run_scan() & check_hosts()

```
#1 Exécute la fonction check_hosts et définit un flag de contrôle à chacun
#2 Exécute la fonction Create_hosts_instance pour chaque hôte découvert
sur le réseau
```

create_host_instance()

```
def create_hosts_instance(host):
    #1 Création d'un objet "host"
    host = Host(host[0], host[1], host[2], host[3], host[4], host[5],
    host[6], host[7], host[8])

    #2 Fonctions d'analyse de l'hôte
    service_list = host.get_service()
    report = host.cve_scan(service_list)
    host.parse_cve(report)

    return host
```

```

get_service()

def get_service(self):
    service_list = []

    #1 Utilise le module NMAP afin de scanner chacun des hôtes
    scanner = nmap.PortScanner()
    scanner.scan(self.ip, arguments='--sS -O --host-timeout 100')

    #2 Récupération des infos ( uptime, os, os version, vendor, mac
    address )
    ...

    host_update = "UPDATE HOST SET MAC = '%s', OS = '%s', VENDOR = '%s',
LAST_BOOT = '%s' WHERE HOST_id = '%s';"

    try:
        ...
    except:
        ...

    #3 Scan et ajoute en BD chacun des ports découverts sur l'hôte
    for proto in scanner[self.ip].all_protocols():
        for port in scanner[self.ip][proto].keys():
            name = scanner[self.ip][proto][port]['name']
            service_list.append([name, port])

    for name, port in service_list:
        service_insert = "INSERT INTO SERVICE (NAME) SELECT '%s' FROM DUAL
WHERE NOT EXISTS( \
                SELECT 1 FROM SERVICE WHERE NAME = '%s') LIMIT 1;"
        host_service_insert = "INSERT INTO HOST_SERVICE (HOST_id,
SERVICE_NAME, SERVICE_PORT) VALUES ('%s', '%s', '%s');"

        try:
            ...
        except:
            ...

    #4 Définit l'état du scan comme terminé
    host_update = "UPDATE HOST SET STATE = 3 WHERE HOST_id = '%s' AND
SCAN_id ='%s';"
    try:
        ...
    except:
        ...

    print("Scan finished for host :", self.ip)

    return service_list

```

cve_scan()

```
def cve_scan(self, service_list):

    #1 Définit le flag de contrôle
    scan_update = "UPDATE SCAN SET VULN_STATE = '%s' WHERE SCAN_id ='%s';"

    ...

    #2 Création d'une liste regroupant les services trouvés
    port_list = []
    for x, y in service_list:
        port_list.append(y)

    port_list = str(port_list).strip('[]')

    #3 Exécution du scan de vulnérabilité sur base de la liste des
    services
    try:
        name = "nmap"
        now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        command = ['nmap', self.ip, '-p', port_list, '--script',
        'vulners', '-sv']
        result = run(command, stdout=PIPE, stderr=PIPE,
universal_newlines=True)
        report = result.stdout

        tool_insert = "INSERT INTO TOOL (HOST_id, NAME, COMMAND, RESULT,
TOOL_START) VALUES ('%s', '%s', '%s', '%s', '%s');"

        try:
            #4 Encodage du résultat du scan afin d'éviter les caractères
            spéciaux et ajout à la BD
            b64_report = report.encode("utf-8")
            b64_report = base64.b64encode(b64_report)
            b64_report = b64_report.decode("utf-8")
            command = str(command).replace("'", "")
            cursor.execute(tool_insert % (self.host_id, name, command,
b64_report, now))
            db.commit()
        except:
            db.rollback()

    except:
        print("scan crash")
        report = ""

    return report
```

parse_cve()

```
def parse_cve(self, report):

    #1 Utilisation du module regex afin de parser les services et chacune
    des cve qui leurs sont associées
    pattern = re.compile(r'(\d*)(tcp|udp)\s+(open|closed|filtered)\s+
    (\w+)(.*?)')
    matches = pattern.finditer(report)

    for match in matches:
        ...

        pattern = re.compile(r'(\w*)/(tcp|udp)\s+(\w+)\s+(\w+-*\w+/\?\w+)\s+
        (.*)(\n|.*)+(CVE-\d{4}-\d{4,5})\s(\d{1,2}\.\d)')
        matches = pattern.finditer(report)

        for match in matches:
            ...

            pattern = re.compile(r'(CVE-\d{4}-\d{4,5})\s(\d{1,2}\.\d)')
            matches = pattern.finditer(match.group(0))

            for match in matches:
                ...
```

json_parser.py

```
import json
import mysql.connector
import glob

...

cursor = db.cursor()
list_full = []

#1 Utilisation du module glob afin de créer une liste contenant des noms
de fichier
file_list = glob.glob("./nvdcve*.json")
print(file_list)

#2 Chacun des fichiers est ouvert en écriture afin d'en extraire les CVE
for file in file_list:
    with open(file) as f:
        data = json.load(f)

        print("\n")

        for cve in data['CVE_Items']:
```

```

cve_ID = cve['cve']['CVE_data_meta']['ID']
cve_impact = cve['impact']

for x in cve['cve']['description']['description_data']:
    cve_Description = x['value']

try:
    cve_BaseScore = cve_impact['baseMetricV3']['cvssV3']
    ['baseScore']
    cve_complexity = cve_impact['baseMetricV3']['cvssV3']
    ['attackComplexity']
    cve_severity = cve_impact['baseMetricV3']['cvssV3']
    ['baseSeverity']
except:
    try:
        cve_BaseScore = cve_impact['baseMetricV2']['cvssV2']
        ['baseScore']
        cve_complexity = cve_impact['baseMetricV2']['cvssV2']
        ['accessComplexity']
        cve_severity = cve_impact['baseMetricV2']['severity']
    except:
        cve_BaseScore = ""
        cve_complexity = "UNKNOWN"
        cve_severity = "UNKNOWN"

cve_list = [cve_ID, cve_BaseScore, cve_complexity, cve_severity,
cve_Description]
list_full.append(cve_list)

#3 Ajoute les CVE extraites des fichiers vers la base de données
for cve in list_full:

    query = "INSERT INTO mitre (name, score, severity, complexity,
description) VALUES ('%s','%s', '%s', '%s', '%s');"
    try:
        cursor.execute(query % (cve[0],cve[1],cve[3],cve[2],cve[4]))
        db.commit()
    except:
        db.rollback()

db.close()

```

FEEDS - <https://nvd.nist.gov/vuln/data-feeds>

exploit.py

Structure

```
from exploit import enumeration, network, ftp, ssh, dns, http, smtp, rpc
from exploit import kerberos, ldap, smb

import nmap
import re
import sys
import mysql.connector
from subprocess import PIPE, run
from datetime import datetime
import base64

db = mysql.connector.connect(…)
cursor = db.cursor()

class Host:

    def __init__(self, host_id, scan_id, ip, mac, hostname, os, vendor, last_boot, state):…

    def get_service_info(self):…

    def run_exploit(self, service_list, subnet):…

def check_hosts(scan_id, start_state, process_state, error_state, subnet):…

def main(scan_id, subnet):

    check_hosts(scan_id, 3, 5, 6, subnet)
    db.close()

if __name__ == "__main__":
    main(scan_id)
```

check_hosts

```
def check_hosts(scan_id, start_state, process_state, error_state, subnet):

    #1 Démarrage de la phase d'exploit pour le scan
    scan_update = "UPDATE SCAN SET EXPLOIT_STATE = '%s' WHERE SCAN_id = '%s';"
    try:
        ...
    except:
        ...

    while True:

        #2 Sélectionne les informations du scan et de l'hôte à exploiter
        scan_select = "SELECT * FROM SCAN WHERE SCAN_id = '%s' LIMIT 1;"
        host_select = "SELECT * FROM HOST WHERE SCAN_id = '%s' AND STATE = '%s' LIMIT 1;"
        host_update = "UPDATE HOST SET STATE = '%s' WHERE HOST_id = '%s';"

        ...

    #3 Crée un objet sur base des informations de l'hôte et lance les
    #fonctions get_service_info() et run_exploit()
```

```

if host:

    ...

    host = Host(host[0], host[1], host[2], host[3], host[4],
host[5], host[6], host[7], host[8])
    service_list = host.get_service_info()
    host.run_exploit(service_list, subnet)

#4 Interrrompt l'exploitation en cas d'erreur
else:

    scan_update = "UPDATE SCAN SET EXPLOIT_STATE = '%s' WHERE
SCAN_id = '%s';"
    try:
        ...
    except:
        ...

    return False

```

get_service_info()

```

#1 Récupère la liste des services découverts sur l'hôte
def get_service_info(self):

    service_list = []
    service_select = "SELECT SERVICE_NAME, SERVICE_PORT FROM
HOST_SERVICE WHERE HOST_id = '%s';"

    try:
        cursor.execute(service_select % (self.host_id))
    except:
        print("Database Error On service Select")
        sys.exit(0)

    records = cursor.fetchall()

    for row in records:
        service_list.append([row[0], row[1]])

    return service_list

```

run_exploit()

```

def run_exploit(self, service_list, subnet):
    ...

```

```

#1 Exécution des scripts ne dépendant pas de service
result_list = network.main(subnet)
results.append(result_list)

result_list = enumeration.main(self.ip)
results.append(result_list)

#2 Exécution des scripts propre à chaque service
for service, port in service_list:

    ...

#3 Exécution des attaques par brute-force
if ftp_state == 1:
    result_list = ftp.main(self.ip, ftp_port)
    results.append(result_list)

if ssh_state == 1:
    result_list = ssh.main(self.ip, ssh_port)
    results.append(result_list)

#4 Parsing des résultats & push en base de données
try:
    for x in results:
        for name, command, report in x:
            ...
except:
    pass

#5 Parsing des utilisateurs & push en base de données
try:
    for x in users:
        for y in x:
            ...
except:
    pass

print("Exploitation finished for host :", self.ip)

```

http.py (Tous les scripts d'exploit ont la même structure)

```

from subprocess import PIPE, run

#2 Fonction d'exploitation
def http_buster(ip, port):

    name = "recursive-gobuster"
    command = ["/home/user/recursive-gobuster/recursive-gobuster.pyz", "-w",
               "/usr/share/seclists/Discovery/Web-Content/common.txt", ip]

```

```
#3 Exécution des tools
    result = run(command, stdout=PIPE, stderr=PIPE,
universal_newlines=True)
    report = result.stdout

    return name, command, report

def http_vuln(ip, port):

    name = "nikto"
    command = ["nikto", "-host", ip, "-port", port, "-Tuning", "x", "6",
"-ask", "no"]
    result = run(command, stdout=PIPE, stderr=PIPE,
universal_newlines=True)
    report = result.stdout

    return name, command, report

def main(ip, port):

    result_list = []

    #1 Exécution des fonctions d'exploitation
    name, command, report = http_buster(ip, port)
    result_list.append([name, command, report])

    name, command, report = http_vuln(ip, port)
    result_list.append([name, command, report])

    return result_list
```

10.2.3 Documentation du Dashboard

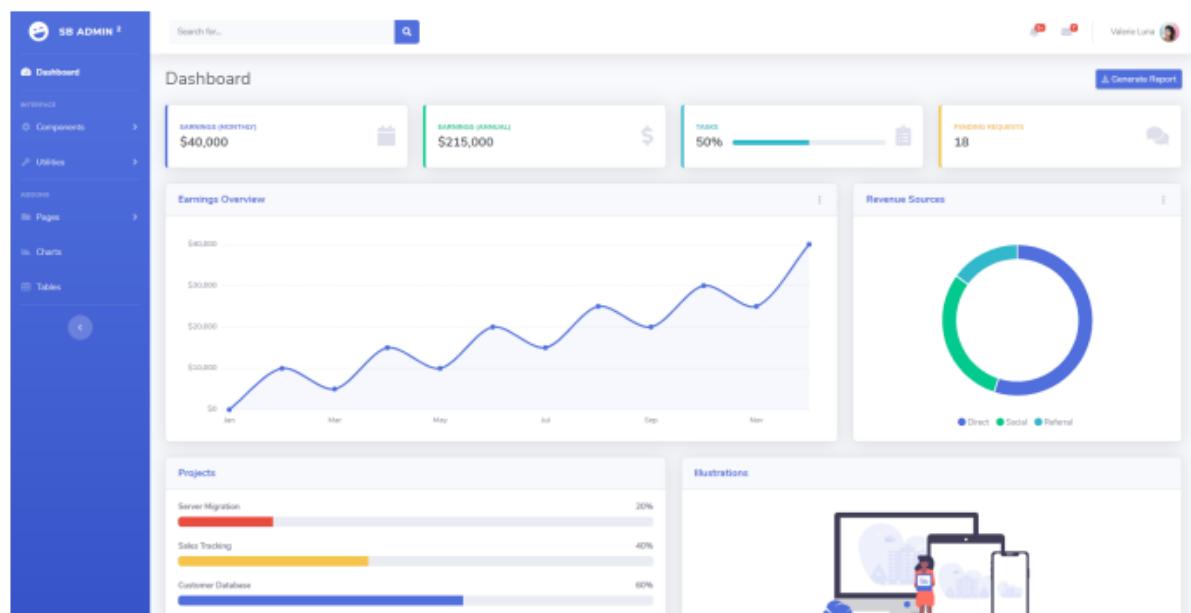
Configuration

Le dossier du dashboard est à placer dans le répertoire /var/www/html/ sur la RPI serveur ou sur un serveur externe.

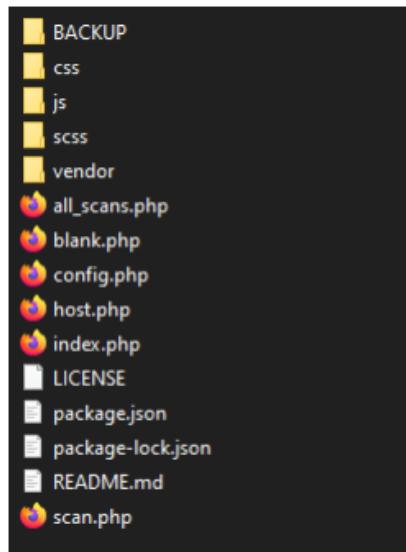
Le site fonctionne sur base de l'API bootstrap, de codes écrits en PHP ainsi que quelques scripts en javascript.

Template utilisé : SB Admin 2

Template - Website



Structure du répertoire



index.php

Présente les informations suivantes :

- nombre total de scans terminés
- nombre total de clients connectés
- les 5 derniers scans effectués
- les informations des clients actuellement connectés et actifs sur un scan
- une barre de progression qui s'affiche lorsqu'un scan est effectué

The screenshot shows the Argus application interface. At the top, there's a header with the Argus logo, a search bar, and navigation links for Home, All scans, and History. A "Generate Report" button is also present.

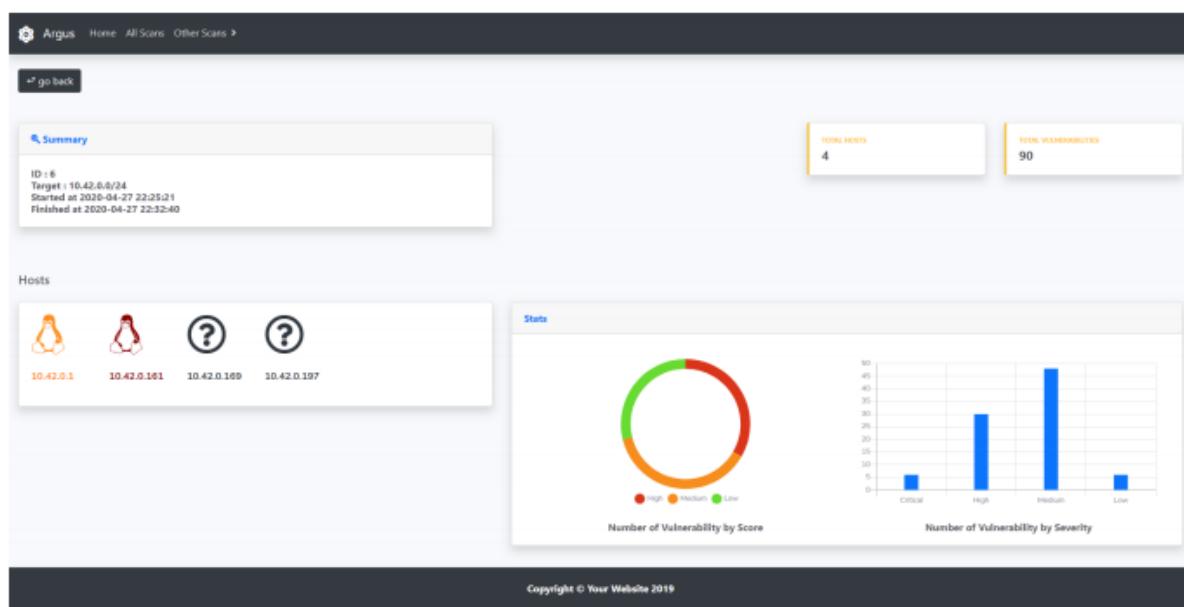
The main dashboard area has two primary sections:

- Latest scans:** This section lists four completed scans with their details:
 - Scan 22:** Target: 10.42.0.0/24, Started at 2020-05-13 22:45:12, Finished at 2020-05-13 22:45:12. A "View info" button is next to it.
 - Scan 21:** Target: 10.42.0.0/24, Started at 2020-05-13 15:15:42, Finished at 2020-05-13 15:16:18. A "View info" button is next to it.
 - Scan 20:** Target: 192.168.0.0/24, Started at 2020-05-11 10:18:27, Finished at 2020-05-13 15:06:58. A "View info" button is next to it.
 - Scan 19:** No details shown.
- Connected RPI:** This section shows a single connected device:
 - RPI 1:** MAC: dc:0f:02:d0:6a:9b, Hostname: kali, IP: 192.168.0.30. It also indicates "Currently working on scan: 22".

scan.php

Présente les informations suivantes :

- résumé des informations relatives au scan (cible et timestamp)
- les hôtes ayant été découverts sur le réseau ciblé (représentés par un logo du type d'équipement et par la couleur de la plus haute CVE découverte)
- statistiques (nombre total d'hôtes et de vulnérabilités)
- graphiques représentant les CVE découvertes par score et par sévérité



host.php

Présente les informations suivantes :

- résumé des informations relatives à l'hôte (IP, OS, Vendeur, dernier redémarrage)
- graphiques représentant les CVE découvertes par score et par sévérité ainsi que le total d'informations récoltées selon leurs types.
- tableau des services découverts ainsi que leur protocole, port et version
- tableau des cve découvertes triées par score
- liste des utilisateurs découverts sur l'hôte
- liste des outils utilisés et leur résultat

Argus Home All Scans Other Scans >

go back 11 - 10.42.0.1 12 - 10.42.0.161 13 - 10.42.0.169 14 - 10.42.0.197

Summary

ID : 12
SCAN ID : 6
IP : 10.42.0.161
OS : Linux 2.6.9 - 2.6.35
Vendor : Apple
Last Boot : Mon Apr 27 22:18:55 2020

Stats

Number of Vulnerability by Score

Score	Count
Critical	5
High	22
Medium	30
Low	2

Number of Vulnerability by Severity

Severity	Count
Service	25
User	28
File	10

Number of information gathered

CVE Detected

Click to Hide	score	name	severity	complexity	service	info
View	1.0	CVE-2010-0425	HIGH	LOW	http	modules/arch/wei/2/mod_expicic in mod_expicic in the Apache HTTP Server 2.0.57 through 2.0.63, 2.2.0 through 2.2.14, and 2.3.x before 2.3.7, when running on Windows, does not ensure proper validation of the host header before calling the user-defined function for an ISAPI dll module, which allows remote attackers to execute arbitrary code via unspecified vectors related to a crafted request, a reset packet, and...
View	1.0	CVE-2019-16211	CRITICAL	LOW	mysql	
View	1.0	CVE-2015-3166	CRITICAL	LOW	mysql	
View	1.0	CVE-2015-0244	CRITICAL	LOW	mysql	
View	1.0	CVE-2018-1312	CRITICAL	LOW	http	
View	1.0	CVE-2017-7679	CRITICAL	LOW	http	
View	1.0	CVE-2018-1115	CRITICAL	LOW	mysql	
View	1.0	CVE-2015-0243	HIGH	LOW	mysql	
View	3.5	CVE-2010-0733	LOW	MEDIUM	mysql	
View	2.6	CVE-2012-2467	LOW	HIGH	http	
View	2.6	CVE-2008-5161	LOW	HIGH	ssh	
View	2.6	CVE-2009-4022	LOW	HIGH	domain	
View	1.0	CVE-2011-4415	LOW	HIGH	http	

Services

port	name	version
21	ftp	vsftpd 2.3.4
22	ssh	OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23	telnet	Linux telnetd
25	smtp	Postfix smtpd
53	domain	ISC BIND 9.4.2
80	http	Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111	rpcbind	
139	netbios-ssn	
445	microsoft-ds	
512	exec	
513	login	
514	shell	

Tools

[enum](#)

[nbtscan](#)

```
CMD : [nbtscan, 10.42.0.0/24]
Started at 2020-04-27 22:32:06
Doing NBT name scan for addresses from 10.42.0.0/24
IP address NetBIOS Name Server User MAC address
10.42.0.197 STRRR acbc:32:88:42:3f
10.42.0.161 METASPLLOITABLE METASPLLOITABLE 00:00:00:00:00:00
```

[enum4linux](#)

[smtp-user-enum](#)

[dig](#)

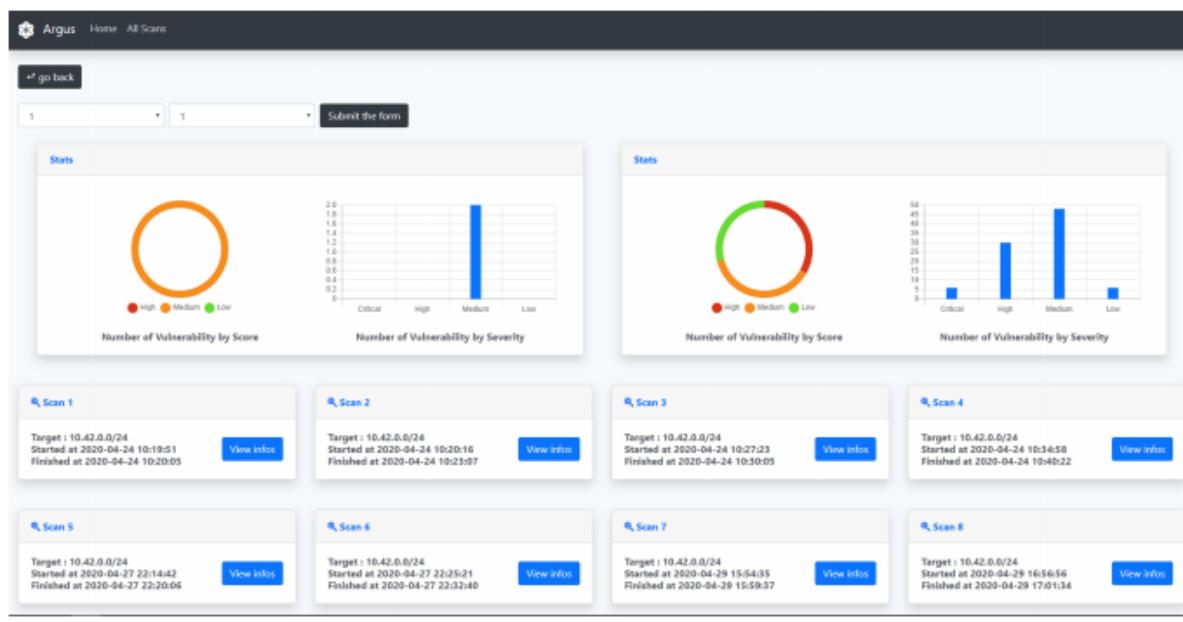
Users

- BACRUP
- MAIL
- NEWS
- POSTMASTER
- ROOT
- SYS
- Service
- USER
- User
- bin
- daemon
- ftp
- games
- lp
- mail
- man
- nobody
- postgres
- postmaster
- root
- root
- root@localhost
- service
- sync
- sys
- user
- ucp

all_scans.php

Présente les informations suivantes :

- liste complète des scans effectués et liés à ce serveur
- deux cadres permettant la comparaison entre deux scans sélectionnés



config.php

Permet de définir les informations de connexion à la base de données

```
<?php

define('DB_SERVER', 'localhost');
define('DB_USERNAME', 'bob');
define('DB_PASSWORD', 'bob');
define('DB_DATABASE', 'STAGE');
$db = mysqli_connect(DB_SERVER,DB_USERNAME,DB_PASSWORD,DB_DATABASE);

?>
```