République Algérienne Démocatique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene

Faculté d'Informatique

Filière : Informatique

Spécialité : Securite des Systemes Informatiques

Module : Systeme d'exploitations

Rapport de projet

Projet

Plateforme de Reconnaissance pour la Collecte d'Informations Systémiques (Système d'Exploitation, Batterie, Processeur, etc.)

Présenté par :

NADOUR	Amani-zohra	212133021152
RAHMOUNE	Fatima Zahra	212131052015
SADOK	Wissam	212131013550
BENHAGOUGA	Raouaa	212131078997
LAMDANI	Hiba	212131095325

Table des matières

Table des figures

Introduction

La phase de reconnaissance, première étape de la Cyber Kill Chain, est cruciale pour tout attaquant cherchant à cibler une organisation ou un système. Elle consiste à collecter un maximum d'informations sur la cible afin de mieux comprendre son environnement, ses vulnérabilités potentielles et ses mécanismes de défense. Cette phase inclut des techniques comme la cartographie réseau, l'identification des systèmes d'exploitation, la collecte de données publiques (OSINT), et même des interactions directes ou indirectes avec la cible. Ce projet vise à aborder la technique d'identification des ordinateurs créant et mettant en œuvre une application desktop de reconnaissance à double mode (local et à distance) afin de reproduire l'interrogation du système.

La première étape pour aborder un projet de cette nature est l'analyse. Il s'agit de définir clairement les objectifs, d'identifier les limitations éventuelles et de déterminer les technologies à utiliser pour garantir la bonne réalisation du projet. Une fois les objectifs définis ainsi que les technologies choisies, l'étape suivante consiste à conceptualiser le projet, puis à passer à sa réalisation concrète et à son développement. Enfin, la dernière étape consiste à identifier les imperfections et proposer des suggestions pour son développement futur. Ce rapport comprend une analyse pré-conceptuelle, un aperçu de l'architecture du système, ainsi que des recommandations pour de futures améliorations et optimisations. Il aborde également les défis rencontrés au cours du développement et les solutions mises en œuvre.

Ainsi, ce projet vise à mieux comprendre les systèmes informatiques ainsi que les mesures de sécurité nécessaires pour les protéger efficacement contre les menaces potentielles.

Analyse

La première phase de ce projet se concentre sur l'analyse et l'évaluation des différentes approches afin d'atteindre son objectif, qui se divise en deux tâches principales : 1) La reconnaissance et l'interrogation du système local, et 2) La reconnaissance et l'interrogation du système à distance. Ce processus consiste à recueillir des informations complètes, notamment :

— la marque et la version du système d'exploitation les spécifications du processeur (telles que la capacité)

— les spécifications du processeur (telles que la capacité)

— la taille totale de la mémoire

— les périphériques connectés

— le pourcentage actuel de la batterie

Dans ce chapitre, nous évaluerons les options disponibles pour chaque tâche, en tenant compte de leur pertinence, efficacité et performance. L'objectif est d'identifier l'approche la plus optimale tout en assurant leur alignement avec les objectifs du projet.

1.1 Reconnaissance Locale

En ce qui concerne la reconnaissance locale, le processus s'est avéré beaucoup plus simple que prévu. Tout ce qui est nécessaire, le système et récupère les informations à l'aide de différents modules.

Cependant, il existe quelques contraintes. Celles-ci concernent principalement les droits d'accès, car bien que la plupart des commandes fonctionnent parfaitement pour les utilisateurs non-administrateurs, d'autres, notamment celles liées aux périphériques, peuvent rencontrer des problèmes.

1.2 Reconnaissance à distance

Pour pouvoir extraire des données d'un autre hôte, il faut effectivement un mécanisme de communication entre les deux machines. Après des études et des tests sur différentes méthodes, chacune a prouvé qu'elle comporte ses propres avantages et inconvénients. Celles-ci sont présentées ci-dessous.

1.2.1 TCP/IP OS Discovery

TCP/IP OS discovery est une technique utilisée pour identifier le système d'exploitation (OS) exécuté sur un système distant en analysant sa réponse aux sondes réseau et aux modèles de trafic. Des outils comme Nmap et Wireshark sont fréquemment utilisés pour effectuer cette découverte.

Nmap (Network Mapper) est un outil open-source puissant, largement utilisé pour l'exploration de réseaux et l'audit de sécurité. Il permet de découvrir les hôtes actifs sur un réseau, d'identifier les ports ouverts et d'obtenir des informations détaillées sur les services et systèmes d'exploitation des machines distantes.

Toutefois, il est important de noter que **Nmap** se limite à des tâches telles que le *banner grabbing*, le scan des ports et la détection des systèmes d'exploitation. Il ne fournit pas d'accès direct aux ressources matérielles comme le CPU, la mémoire ou les périphériques des machines distantes.

Wireshark est un outil performant pour l'analyse du trafic réseau et l'observation des protocoles, offrant une vue précieuse sur le comportement d'un système distant. Cependant, ses capacités de reconnaissance des systèmes d'exploitation sont limitées par sa nature passive, dépendant du trafic réseau existant et du type d'OS. De plus, son incapacité à interagir activement avec les cibles et sa limite face aux communications chiffrées en font un outil insuffisant pour une découverte approfondie des systèmes.

Il existe d'autres méthodes de découverte du système d'exploitation via TCP/IP, cependant, toutes ont finalement abouti à la même limitation : des informations insuffisantes.

1.2.2 Connexion à distance via SSH

SSH (Secure Shell) permet de se connecter à distance à une machine de manière sécurisée, couramment utilisé sur Linux et macOS, mais nécessitant l'installation préalable d'OpenSSH ou PuTTY sur Windows. Ce dernier requiert souvent des privilèges administratifs pour installer et configurer le serveur SSH. Une fois la connexion établie, un mot de passe ou une clé privée est utilisé pour garantir la sécurité de la communication. SSH permet ainsi d'exécuter des commandes à distance et de récupérer des informations système.

Paramiko, une bibliothèque Python, facilite la gestion de connexions SSH, en permettant d'exécuter des commandes et de manipuler des fichiers à distance via ce protocole. Bien qu'elle simplifie l'utilisation de SSH dans des scripts Python, elle repose sur les mêmes principes que SSH pour l'établissement des connexions. Nous n'avons pas utilisé SSH dans le projet, car ces deux méthodes permettent de réaliser la reconnaissance uniquement avec l'accord explicite de l'utilisateur.

1.2.3 Connexion à distance via PowerShell

Pour des machines fonctionnant sous **Windows**, **PowerShell Remoting** PowerShell Remoting, basé sur WinRM, est une solution courante pour les connexions à distance

sur Windows et peut être automatisé avec Python. Il permet d'exécuter des commandes pour collecter des informations système comme le processeur, la mémoire ou les périphériques. Cependant, nous ne l'avons pas utilisé dans notre projet en raison de ses limitations, notamment ; l'activation manuelle de WinRM, la dépendance aux ports spécifiques (5985/5986), et surtout la nécessité d'une authentification valide (mot de passe ou autre).

1.2.4 SNMP

SNMP (Simple Network Management Protocol) permet la surveillance et l'interrogation à distance pour gérer et collecter des informations. La reconnaissance SNMP consiste à identifier les détails des périphériques connectés, comme les adresses IP et les versions des logiciels, en utilisant des requêtes spécifiques et des OIDs. L'interrogation SNMP, via des commandes telles que GET et GETNEXT, facilite la récupération des données ou la modification des paramètres des dispositifs.La reconnaissance et l'interrogation via SNMP dépendent de l'activation et de la configuration correcte du protocole sur la machine cible. Si le protocole SNMP n'est pas activé ou est configuré de manière restrictive, l'accès aux informations peut être limité ou impossible. Par conséquent, cette méthode s'avère peu intéressante.

1.2.5 Connexion client-serveur

Une autre approche a été considéré, qui est une application web. Cependant, dans une application web, il est impossible d'accéder aux informations système détaillées de l'utilisateur final (comme le CPU, la mémoire, etc.) en raison des restrictions de sécurité des navigateurs. Ces derniers fonctionnent dans un environnement sandboxé qui empêche tout accès direct aux ressources locales de la machine de l'utilisateur pour protéger sa confidentialité. Bien qu'une technologie côté serveur puisse collecter des informations sur la machine qui héberge le serveur c'est a dire localement, cela reste limité à ce dernier et n'est pas applicable à pour une reconnaisance a distance. Nous n'avons pas adopté cette méthode dans notre projet , car elle pourrait fonctionner localement mais ne permettrait pas d'accéder aux données d'un autre utilisateur dans une application web. Pour obtenir de telles informations, une application native installée localement avec les autorisations nécessaires serait indispensable

1.2.6 Connexion Bidirectionnelle par Socket

Les sockets sont des interfaces de communication permettant l'échange de données entre des programmes, principalement entre un client et un serveur, via un réseau. Cette communication peut se produire soit localement (sur une même machine), soit à distance, à travers un réseau.

Pour établir une connexion par socket, un serveur doit d'abord se mettre en écoute sur un port précis. Ensuite, un client peut se connecter à ce serveur en utilisant son adresse IP et le port spécifié. Une fois la connexion établie, le client peut envoyer des requêtes ou des données, auxquelles le serveur peut répondre.

Dans le cadre de notre projet, l'utilisateur jouerait le rôle du serveur, tandis que la cible agirait comme un client. Cette méthode présente deux inconvénients principaux :

— **L'adresse IP (et parfois le port) change avec le changement de réseau** : Lorsque la cible change de réseau (par exemple en passant d'un réseau Wi-Fi à un réseau mobile), son adresse IP peut changer. Cela complique la connexion entre le serveur (l'utilisateur) et le client, car l'utilisateur doit connaître l'adresse IP actuelle de la cible pour pouvoir se reconnecter. De plus, si un port spécifique est utilisé, celui-ci peut également changer, surtout dans les cas où le client utilise des configurations de type NAT ou un pare-feu.

— **La cible doit tout de même donner son autorisation, quel que soit qu'elle en soit consciente ou non, pour transférer les données vers l'utilisateur** : Même si la cible est consciente ou inconsciente du transfert de données, elle doit toujours donner son accord pour établir la connexion et permettre l'échange de données. Si cette autorisation est obtenue sans le consentement explicite de la cible, cela peut soulever des problèmes éthiques et juridiques.

1.3 conclusion

Dans le chapitre précédent, nous avons exploré et examinée diverses méthodes de reconnaissance locale et à distance, notamment la découverte du système via TCP/IP, SSH, PowerShell, et une solution client-serveur. Après analyse, nous avons choisi la connexion bidirectionnelle par socket pour sa flexibilité et sa communication en temps réel. Le deuxième chapitre détaillera la conception et les résultats de cette solution.

Conception et Développement

Les fonctionnalités seront mises en œuvre à l'aide d'une application bien conçue. Dans ce chapitre, nous aborderons la structure de cette application, les différentes étapes et cadres disponibles, ainsi que la justification des choix techniques effectués au cours de son développement.

2.4 Conception

Les mécanismes et les étapes de fonctionnement ont été choisis à partir des besoins et des technologies disponibles. notamment :

2.4.1 Scénario local

Le scénario local consiste simplement en une exécution interrogative du système qui récupère les informations et les présente à l'utilisateur. Il utilise différentes commandes et fonctions selon le système d'exploitation installé. Les informations acquises sont : le système d'exploitation et sa version, le hostname, la marque du CPU ainsi que sa capacité et le nombre de cœurs, la taille de la mémoire, les partitions du disque, les périphériques (imprimantes, écrans, claviers et souris), ainsi que le pourcentage de batterie. Les informations concernant les périphériques sont difficiles à récupérer, car si l'utilisateur n'est pas administrateur, le résultat peut varier en fonction de ses droits d'accès.

- Pour Windows, le script fonctionne pour les utilisateurs non administrateurs, mais les requêtes concernant les périphériques via WMI (Windows Management Instrumentation) pourraient poser problème en fonction de la configuration du système.
- Pour Linux, les informations de base sur les périphériques et le système sont accessibles, mais certaines commandes spécifiques pourraient nécessiter des privilèges supplémentaires.
- Pour macOS, le script fonctionnera généralement, mais certaines commandes pourraient nécessiter des permissions élevées pour accéder à des informations système détaillées.

En implémentant une gestion d'erreurs, le script pourra anticiper et gérer les limitations liées aux droits d'accès, garantissant ainsi une exécution fluide, même en cas de permissions insuffisantes ou de configurations système variées.

2.4.2 Scénario d'attaque

Le scénario d'attaque est un peu plus complexe, mais un mécanisme solide a été défini, et voici comment il se déroule :

1. L'utilisateur (l'interrogateur) génère un exécutable qui établit la connexion bidirectionnelle avec sa machine, c'est-à-dire qu'il contient l'adresse IP ainsi que le port qui sera en écoute.
2. L'utilisateur envoie ensuite l'exécutable à la cible et met sa machine en mode écoute.
3. Au moment où la cible, qu'elle le fasse consciemment ou inconsciemment, double-clique ou exécute l'exécutable, la connexion est établie et les informations sont envoyées.

Le principal problème rencontré dans cette partie réside dans le fait que l'adresse IP et le port utilisés changent. l'adresse ip en cas de changement de sous-réseau, et le port au cas ou un port est ouvert pour une autre fonction. De plus, la génération d'un exécutable à chaque fois est pratiquement impossible sans l'installation de Python sur l'ordinateur. Notre solution consiste à créer un exécutable (appelons-le *connect.exe*) qui prend l'adresse IP et le port comme arguments, puis établit la connexion. Cet exécutable est ensuite intégré à l'exécutable principal de l'application (voir la figure 2.1).

Concernant les ports, le processus de sélection automatique d'un port consiste à analyser la plage des ports dynamiques, qui s'étend généralement de 49152 à 65535 (selon les standards IANA). La procédure commence par vérifier chaque port de cette plage, afin de déterminer son statut. Le processus se poursuit jusqu'à ce qu'un port non utilisé (fermé) soit trouvé, puis sélectionné pour être utilisé. Cette méthode garantit que l'application ou le service acquiert dynamiquement un port libre, évitant ainsi les conflits avec les ports bien connus ou déjà occupés. Par contre, l'adresse ip est disponible via la fonction socket.gethostbyname(socket.hethostname()).

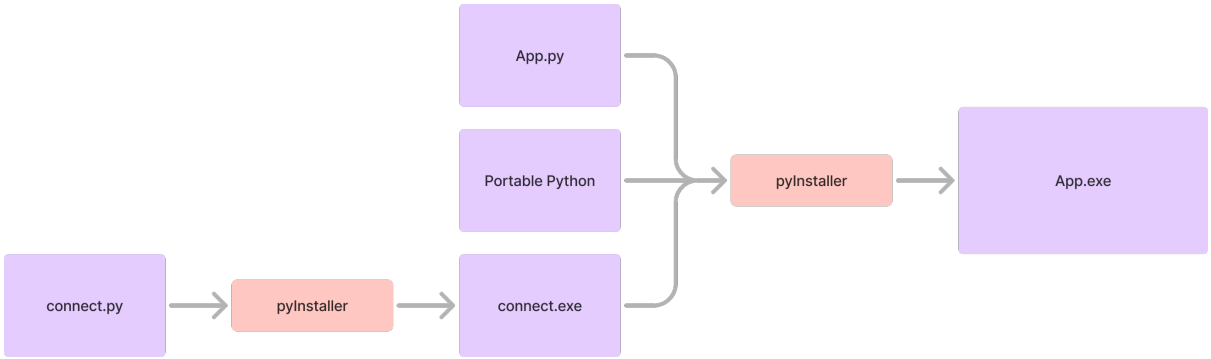

FIGURE 2.1 – Intégration des fichiers dans *App.exe*

Nous avons trouvé un convertisseur portable de batch en exécutable et l'avons déjà intégré dans l'application. Cependant, nous avons décidé de ne pas l'utiliser, car il générait un avertissement indiquant que *cible.exe* contenait un virus.

Ensuite, nous avons trouvé un interpréteur Python portable pour Windows [1], que nous avons également intégré à l'exécutable principal (voir la figure 2.1), et qui prend en charge des bibliothèques très limitées, dont *py2exe*. Bien que cela reste difficile, car de

nombreux modules cruciaux, comme *subprocess*, ne sont pas supportés, nous avons trouvé des alternatives à chacun d'eux (e.g. *os.system()* comme alternative à *subprocess.run()*).

De plus, py2exe ne regroupe pas de fichiers supplémentaires, ce qui pose un probleme, car nous ne voulions pas que la cible ait des dépendances avec différents fichiers, mais uniquement un fichier unique. Alors pour garantir l'unicité de l'exécutable, nous avons encodé *connect.exe* (voir la figure 2.2), qui sera ensuite décodé par l'exécutable de la cible et supprimé immédiatement après son exécution (voir la figure 2.3).

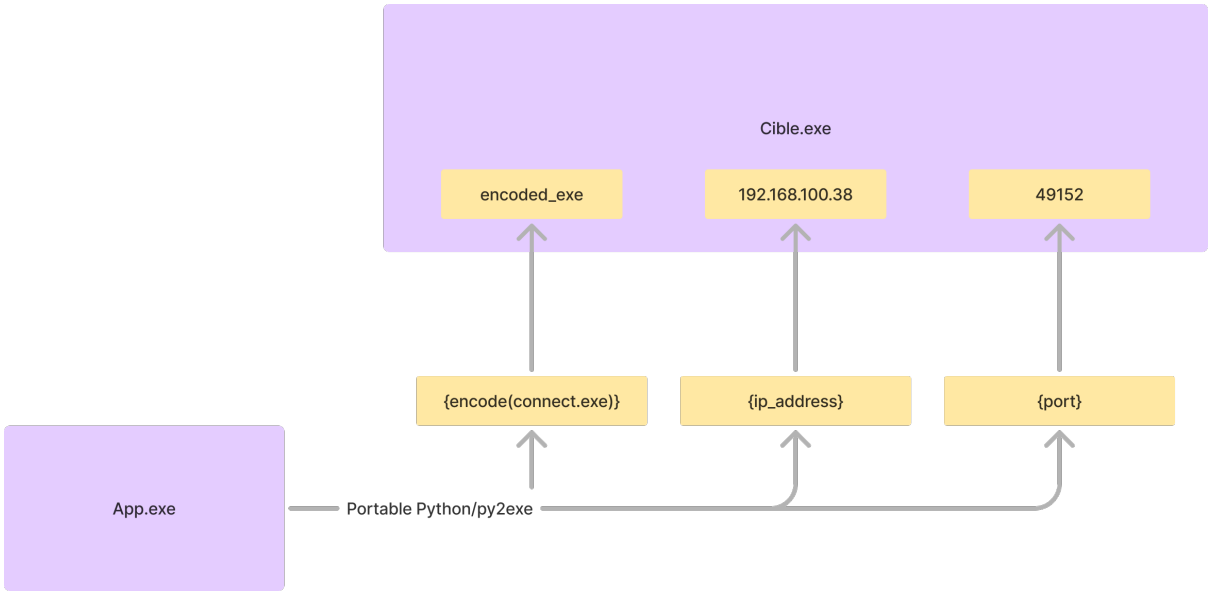

FIGURE 2.2 – Génération de *cible.exe*

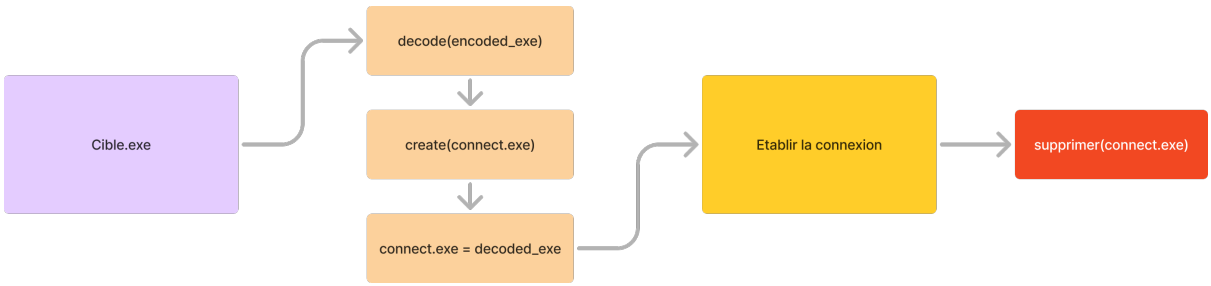

FIGURE 2.3 – Exécution de *cible.exe*

L'exécutable cible.exe se trouve dans un emplacement de destination spécifié par l'utilisateur ; Pendant sa génération, App.exe génère quelques fichiers (setup.py et cible.py) ainsi que des répertoires (build et dist) qui seront automatiquement supprimés.

L'interrogateur envoie ensuite l'exécutable à la cible. Il a la liberté de changer le nom de cible.exe et de jouer avec le concept de social engineering pour inciter ou appâter la cible à l'exécuter.

Généralement, les systèmes technologiques sont sécurisés, et les violations ou l'accès aux informations personnelles sont souvent liés à l'inconscience de la cible face aux protocoles de sécurité.

2.5 Développement

2.5.1 Choix de langage de programmation

Le langage de programmation choisi pour concrétiser cette conception est **Python**. Ce choix est justifié par plusieurs facteurs clés :

— **Facilité de développement** : Python est réputé pour sa syntaxe claire et concise, ce qui facilite l'écriture et la maintenance du code.

— **Richesse des bibliothèques** : Python dispose d'une large gamme de bibliothèques adaptées aux besoins variés.

— **Rapidité de prototypage** : Python permet de créer rapidement des prototypes et de tester des concepts, ce qui accélère le cycle de développement.

— **Programmation système** : Python offre des modules comme *os* et *subprocess*, qui permettent une interaction efficace avec le système, rendant possible l'exécution de tâches de bas niveau.

— **Communauté active** : Une communauté étendue et active permet d'accéder facilement à des ressources, des modules et du support, rendant le développement plus rapide et efficace.

De plus, Python offre de nombreux outils pour développer des interfaces graphiques (**Tkinter** dans ce cas) qui simplifient la création d'interfaces utilisateurs attrayantes et intuitives. Grâce à ces bibliothèques, l'intégration d'une interface graphique dans notre travail est non seulement facile mais aussi rapide, ce qui permet d'améliorer l'interactivité et l'expérience utilisateur.

2.5.2 Présentation de l'Application

Dans cette section, nous présenterons les différentes captures d'écran illustrant les étapes de la reconnaissance : 2.4, 2.5, 2.6, 2.7 et 2.8, qu'elle soit effectuée localement ou à distance.

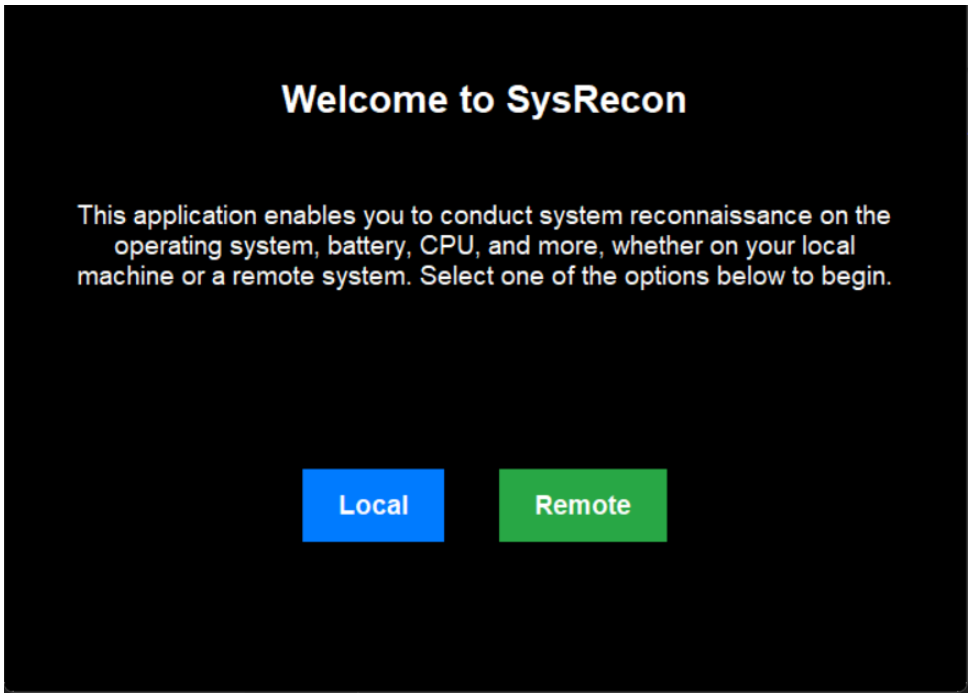

FIGURE 2.4 – Page d'accueil

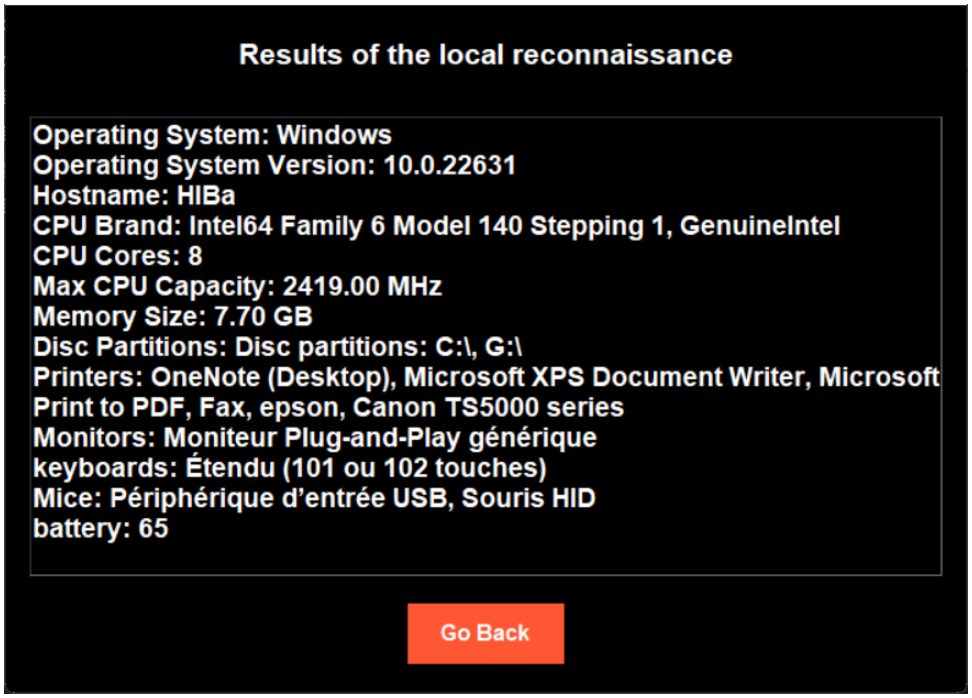

FIGURE 2.5 – Reconnaissance locale

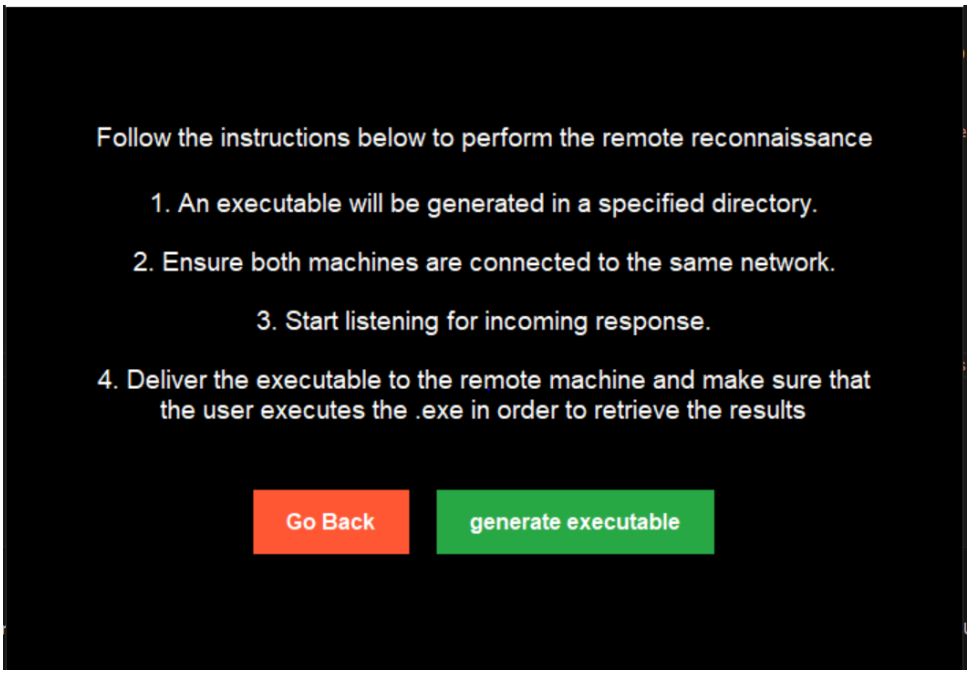

FIGURE 2.6 – Génération d'exécutable *cible.exe*

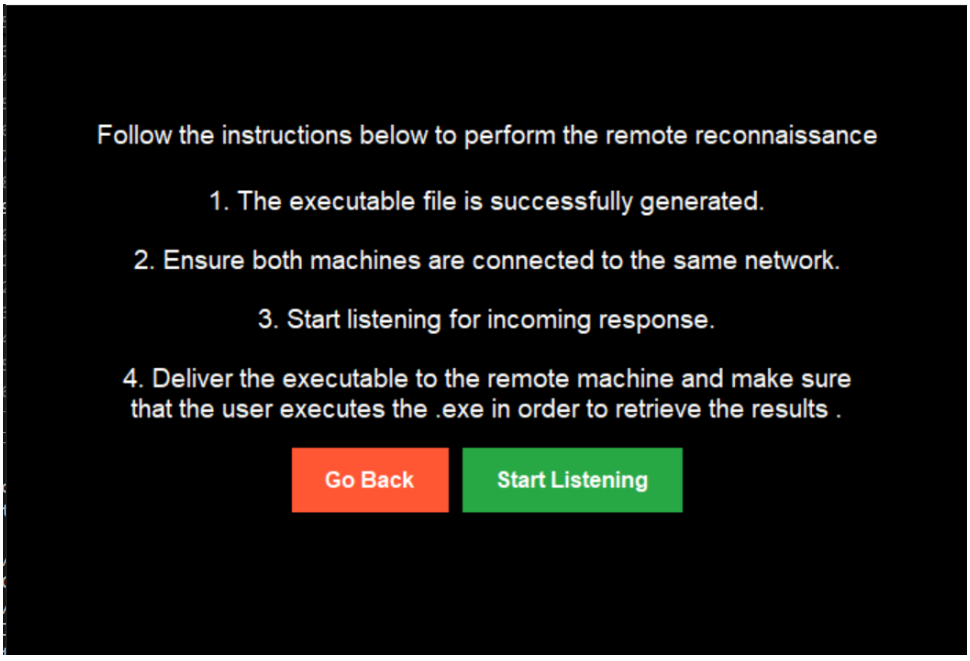

FIGURE 2.7 – Commencer l'ecoute

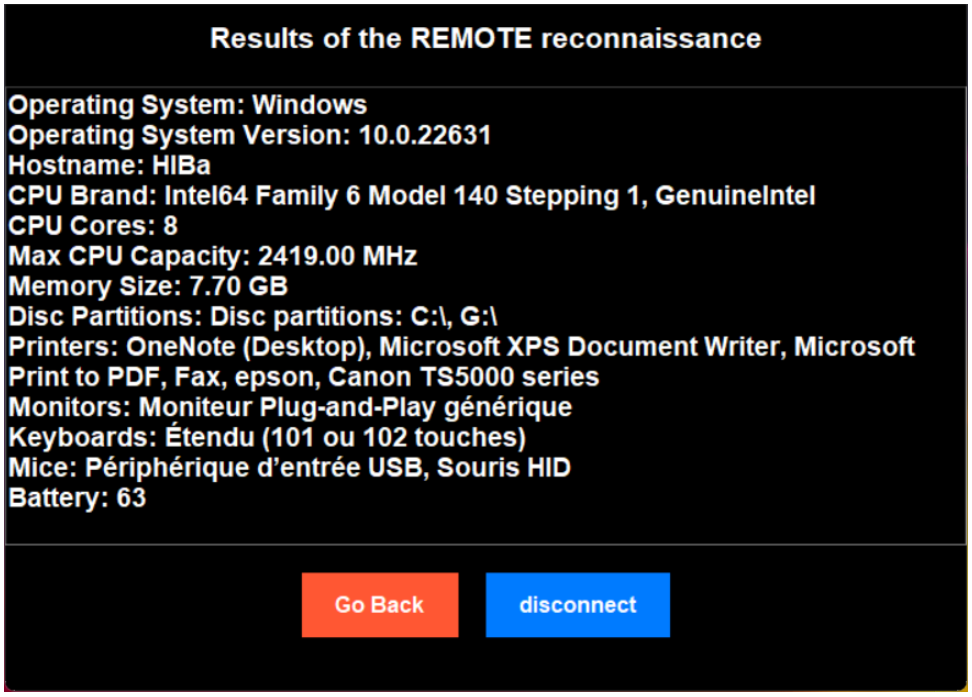

FIGURE 2.8 – Reconnaissance à distance

Améliorations futures

Dans le cadre de ce projet, plusieurs fonctionnalités ont été envisagées pour enrichir et optimiser l'application. Cependant, en raison de certaines limitations, notamment le temps imparti et les ressources disponibles, leur mise en œuvre n'a pas été possible dans la version actuelle, mais nous envisageons de les intégrer dans des versions futures, afin de rendre l'application plus performante.

Parmi ces améliorations potentielles, nous pouvons citer :

- **Reconnaissance simultanée de plusieurs machines à distance :** Actuellement, l'application permet d'interroger une seule machine à la fois. Une évolution majeure consisterait à intégrer la possibilité de reconnaître et d'interroger plusieurs machines simultanément. Cela permettrait une gestion plus efficace des systèmes dans des environnements multi-utilisateurs ou réseaux complexes.
- **Amélioration de l'interface utilisateur :** Bien que l'interface actuelle soit fonctionnelle, une refonte pour la rendre plus intuitive et interactive pourrait améliorer l'expérience utilisateur. Par exemple, la possibilité d'afficher les informations collectées sous forme de tableaux dynamiques ou de graphiques visuels pourrait offrir une meilleure lisibilité.
- **Rapports et sauvegarde des données :** Intégrer une fonctionnalité permettant de générer des rapports complets (PDF ou Excel) et de sauvegarder les données collectées ou de les exporter sous des formats universels faciliterait leur partage et leur réutilisation, notamment dans un contexte professionnel ou technique.
- **Sécurisation avancée des connexions :** Pour renforcer la confidentialité et protéger les données échangées entre les machines, l'intégration de protocoles de sécurisation comme SSL/TLS pourrait être envisagée. Cela garantirait que les communications soient cryptées et inviolables.
- **Support multiplateforme étendu :** Bien que l'application soit actuellement conçue pour fonctionner uniquement sur Windows, des adaptations ou améliorations pourraient être envisagées pour offrir une expérience plus homogène sur d'autres systèmes d'exploitation tels que Linux et macOS.

Ces pistes d'amélioration font partie d'une vision à long terme, visant à créer une application robuste et flexible.

Conclusion

En conclusion, ce projet a fait des progrès significatifs vers la réalisation des objectifs principaux définis au départ. Bien que tous les objectifs n'aient pas été entièrement atteints (la reconnaissance a distance n'est pas passive), le développement et la mise en œuvre de l'application, qui comprend des tâches de reconnaissance locale et de reconnaissance à distance, mettent en évidence la praticité et l'efficacité partielle de notre approche. L'application a été conçue pour tirer pleinement parti des mécanismes et solutions choisis, tout en offrant une interface intuitive et conviviale, permettant ainsi à l'utilisateur de naviguer facilement et d'interagir avec les différentes fonctionnalités sans difficulté.

Le parcours du projet a inclus des phases de recherche, de conception et de tests, chacune contribuant à la robustesse du produit final à de divers degrés. La tâche de reconnaissance locale a servi de base fondamentale, permettant de valider les fonctionnalités principales dans un environnement contrôlé. La réalisation de la reconnaissance à distance a permis de mieux comprendre les systèmes informatiques et les différentes manières dont ils peuvent se connecter entre eux, que ce soit de manière passive ou active.

Des défis tels que la génération d'exécutables à partir de l'application ont été relevés grâce à des choix et des solutions innovants, améliorant à la fois la compréhension technique et les compétences en résolution de problèmes. Les résultats d'apprentissage issus de ces expériences ont renforcé notre compréhension des techniques d'exploitation des systèmes, des mécanismes de sécurité et de leurs applications pratiques.

L'un des enseignements les plus marquants de ce projet est que la principale raison pour laquelle la sécurité d'un système informatique est compromise réside souvent dans l'erreur humaine. Cette conclusion souligne l'importance cruciale d'une éducation, d'une sensibilisation et d'une formation robustes des utilisateurs en tant que composantes essentielles de toute stratégie de cybersécurité. Bien que les mesures de protection techniques soient indispensables, le facteur humain reste le plus faible, ce qui souligne la nécessité d'une base d'utilisateurs bien informée et vigilante pour maintenir l'intégrité de la sécurité.

En résumé, ce projet a offert une occasion précieuse d'appliquer des connaissances théoriques dans un cadre pratique, renforçant nos compétences techniques et analytiques. L'application développée a permis de valider certaines approches tout en identifiant des axes d'amélioration pour de futurs travaux.

Bibliographie

[1] Portablepython. https://portablepython.com/, 2022.