RAPPORT DE DÉVELOPPEMENT DE PROJET : HomeGuard

---------------------------------

-- SYNTHÈSE DU PROJET (EXECUTIVE SUMMARY) --

---------------------------------

Nom du Projet : HomeGuard

Mission : Offrir aux particuliers un outil simple et intuitif pour sécuriser leur réseau domestique en identifiant les appareils connectés (IoT) et en les alertant sur les vulnérabilités de sécurité connues.

Problème Résolu : La prolifération des objets connectés (caméras, assistants vocaux, ampoules, etc.) a créé une nouvelle surface d'attaque massive au sein des foyers. Les utilisateurs n'ont souvent aucune conscience des risques : mots de passe par défaut, logiciels non mis à jour, ports ouverts, etc. HomeGuard démocratise la cybersécurité domestique.

Proposition de Valeur : Un scan en un clic pour transformer le chaos des appareils connectés en un rapport de sécurité clair, compréhensible et actionnable.

---------------------------------

-- 1. ARCHITECTURE DE LA SOLUTION --

---------------------------------

Le projet est décomposé en 4 modules principaux qui communiquent entre eux.

1.1. Module de Découverte (Discovery):

- Rôle : Identifier tous les appareils connectés sur le même réseau local (Wi-Fi/Ethernet) que l'ordinateur qui lance le scan.

- Fonctionnement : Envoi de requêtes sur le réseau (ex: scan ARP, Ping scan) pour lister les adresses IP et les adresses MAC actives.

1.2. Module d'Identification (Fingerprinting):

- Rôle : Pour chaque appareil découvert, essayer de deviner de quoi il s'agit (marque, type d'appareil, etc.).

- Fonctionnement :

\* Analyse de l'adresse MAC : Les 3 premiers octets d'une adresse MAC identifient le fabricant (ex: "Apple", "Google", "TP-Link").

\* Scan de ports et de services : Scanne les ports ouverts sur chaque appareil pour identifier les services en cours d'exécution (ex: port 80 pour un serveur web).

\* Analyse des bannières : Récupère les "bannières" des services pour connaître la version du logiciel (ex: "Apache/2.4.29").

1.3. Module d'Analyse de Vulnérabilités (Vulnerability Analysis):

- Rôle : Cœur intelligent de l'application. Il croise les informations collectées avec des bases de données de menaces.

- Fonctionnement :

\* Recherche de CVE : Pour chaque service et version identifiés, il interroge une base de données de vulnérabilités publiques (CVE) pour voir s'il existe des failles connues.

\* Vérification des Mots de Passe Faibles : Il tente de se connecter aux services standards (SSH, FTP, web) avec une liste de mots de passe par défaut très courants ("admin/admin", "root/password", etc.).

\* Détection de Configuration à Risque : Il vérifie si des ports dangereux sont ouverts sur Internet sans raison.

1.4. Module de Rapport (Reporting & UI):

- Rôle : Présenter les résultats de manière claire et simple à l'utilisateur via une interface graphique (GUI).

- Fonctionnement : Affiche la liste des appareils, leur attribue un score de sécurité (ex: vert, orange, rouge) et donne des détails sur les failles ainsi que des conseils de remédiation simples.

---------------------------------

-- 2. CHOIX TECHNOLOGIQUES (STACK TECHNIQUE) --

---------------------------------

\* Langage de Programmation : Python 3.x

- Pourquoi ? Écosystème de bibliothèques réseau et sécurité immense, facile à apprendre, parfait pour le prototypage rapide.

\* Bibliothèques Python Essentielles :

- Scan Réseau et Identification : python-nmap

\* Pourquoi ? Wrapper simple pour Nmap, le standard de l'industrie. Fait 80% du travail d'identification. (Commande: pip install python-nmap)

- Recherche de Fabricant (MAC) : API de macvendors.com ou similaire via la bibliothèque `requests`.

- Recherche de Vulnérabilités (CVE) : API de cve-search.org (via `requests`) ou bibliothèque `cve-search` en local.

- Test de Mots de Passe : `paramiko` pour SSH, `ftplib` pour FTP.

\* Interface Utilisateur (GUI) - Options :

- Option 1 (Recommandée pour débuter) : Streamlit

\* Pourquoi ? Incroyablement simple pour créer une application web interactive à partir d'un simple script Python. (Commande: pip install streamlit)

- Option 2 (Classique Desktop) : PyQt6 ou Tkinter

\* Pourquoi ? Pour créer une application de bureau native. PyQt est plus moderne, Tkinter est plus simple et inclus de base.

- Option 3 (Web Avancée) : Flask ou Django

\* Pourquoi ? Pour une application web complète et personnalisable. Plus complexe.

---------------------------------

-- 3. PLAN DE DÉVELOPPEMENT DÉTAILLÉ (ROADMAP) --

---------------------------------

PHASE 1 : LE CŒUR FONCTIONNEL (Scripts en Ligne de Commande)

Objectif : Faire fonctionner toute la logique sans interface graphique.

1. Mise en Place de l'Environnement :

- Installer Python 3.

- Installer Nmap sur le système d'exploitation.

- Créer un environnement virtuel : `python -m venv venv`

- Activer l'environnement et installer les bibliothèques : `pip install python-nmap requests`

2. Étape 1 : Découverte et Identification

- Créer un script `scan.py`.

- Utiliser `python-nmap` pour scanner le réseau local (ex: `192.168.1.0/24`).

- Pour chaque hôte, afficher dans la console : IP, état, adresse MAC, nom du fabricant.

3. Étape 2 : Scan de Services et Versions

- Améliorer `scan.py` pour utiliser les arguments Nmap de détection de services et versions (`-sV`).

- Afficher la liste des ports ouverts et des services pour chaque hôte.

4. Étape 3 : Intégration de la Recherche de Vulnérabilités

- Créer une fonction qui prend un nom de service/version, interroge l'API de `cve-search.org` et retourne les CVE.

- Intégrer cette fonction dans la boucle principale du scan.

PHASE 2 : CRÉATION DE L'INTERFACE UTILISATEUR (avec Streamlit)

Objectif : Présenter les résultats de manière claire.

1. Création de l'App :

- Créer un fichier `app.py`.

- Importer streamlit (`import streamlit as st`) et ajouter un titre (`st.title(...)`).

2. Intégration du Scan :

- Ajouter un bouton `if st.button('Lancer le Scan'):`.

- Quand on clique, appeler les fonctions de la Phase 1.

- Afficher un message "Scan en cours...".

3. Affichage des Résultats :

- Utiliser les composants Streamlit pour un affichage clair.

- Pour chaque appareil : `st.expander(f"Appareil X (IP) - Score: Risqué")` pour créer une section cliquable.

- À l'intérieur, lister les détails : MAC, ports, et les vulnérabilités trouvées.

PHASE 3 : AMÉLIORATIONS ET FINITIONS

Objectif : Polir le projet pour le rendre impressionnant.

1. Ajouter des Conseils de Remédiation : Pour chaque faille, ajouter un texte simple comme "ACTION REQUISE : Mettez à jour ce logiciel".

2. Génération de Rapport PDF : Ajouter un bouton pour exporter les résultats en utilisant une bibliothèque comme `FPDF2`.

3. Packaging de l'Application : Utiliser `PyInstaller` ou `cx\_Freeze` pour créer un fichier exécutable (.exe, .app) qui ne nécessite pas d'installer Python.

---------------------------------

-- 4. PRÉPARATION POUR LA PRÉSENTATION AU JITECH --

---------------------------------

4.1. La Démonstration Live (Le Clou du Spectacle) :

- Préparation : Utiliser un point d'accès Wi-Fi personnel (téléphone) pour plus de fiabilité.

- La "victime" : Connecter au hotspot un vieil appareil connu pour être vulnérable (ex: Raspberry Pi avec mot de passe par défaut).

- Scénario de la démo :

1. Phrase choc d'introduction.

2. Lancer HomeGuard et son interface simple.

3. Cliquer sur "Lancer le Scan". Le public voit les appareils apparaître en temps réel.

4. Le scan se termine et l'appareil "victime" est marqué en ROUGE vif.

5. Cliquer dessus pour afficher le rapport clair : "ALERTE : Mot de passe par défaut détecté".

6. Conclure sur la simplicité et la puissance de l'outil.

4.2. Le Pitch (Discours de présentation) :

- Problème : Chiffres clés sur les hacks d'objets connectés.

- Solution : Présenter HomeGuard comme la solution simple et accessible.

- Démo : Exécuter le scénario ci-dessus.

- Impact : Expliquer comment le projet aide les gens à sécuriser leur vie numérique.

- Futur : Évoquer les améliorations possibles (version mobile, alertes automatiques).