Rapport de Projet

Développement d'une Application de Sécurisation et Détection de Phishing des Emails via IMAP

Réalisé par :

- **Nadir Meroua**
- **Bouchra Zaidi**
- **Mosaoui Imene**
- **Malek**

Module : Advanced programing

2026 / 2027

1. Introduction

Les attaques par email représentent aujourd'hui l'un des vecteurs principaux de cyberattaque. Les techniques de phishing permettent aux attaquants de tromper les utilisateurs afin de voler leurs identifiants, installer des malwares ou les rediriger vers des sites frauduleux. Les utilisateurs utilisent souvent des services de messagerie comme Gmail sans outils d'analyse approfondie du contenu des messages reçus.

Dans ce projet, nous avons développé une application de sécurité qui permet de récupérer les emails d'un compte Gmail via IMAP, d'analyser leur contenu, de détecter les tentatives de phishing, d'analyser les pièces jointes et de vérifier les liens avec le service VirusTotal. L'application fournit également un système de quarantaine pour isoler les emails suspects.

L'application est conçue sous forme d'une API backend avec FastAPI et une interface frontend en HTML/CSS/JavaScript.

2. Objectifs du projet

Les objectifs principaux du projet sont :

1. Permettre la création de comptes utilisateurs sécurisés.
2. Connecter un compte Gmail via IMAP.
3. Récupérer automatiquement les emails.
4. Extraire le contenu texte des messages.
5. Détecter les liens suspects dans les emails.
6. Vérifier les liens avec VirusTotal.
7. Analyser les pièces jointes.
8. Calculer un score de risque.
9. Mettre en quarantaine les emails dangereux.
10. Afficher les résultats dans une interface web claire.

3. Problème technique principal rencontré

Le principal problème rencontré concerne l'accès IMAP à Gmail.

Google ne permet plus l'accès IMAP avec le mot de passe normal du compte pour des raisons de sécurité. Si on essaye de se connecter avec le mot de passe Gmail classique, la connexion est bloquée.

Cela empêche une application externe de lire les emails via IMAP.

4. Solution adoptée — Google App Password

La solution consiste à utiliser le système Google App Password.

Processus suivi :

1. L'utilisateur active IMAP dans les paramètres Gmail.
2. L'utilisateur active la vérification en deux étapes.
3. L'utilisateur génère un mot de passe d'application Google.
4. Google fournit un mot de passe spécial à 16 caractères.
5. Ce mot de passe est utilisé par notre application pour IMAP.

Avantages :

- Le vrai mot de passe Gmail n'est jamais utilisé.
- Le mot de passe application peut être révoqué.
- Accès limité au service IMAP.
- Conforme aux règles de sécurité Google.

5. Architecture générale du système

Le système est composé de deux parties principales.

5.1 Backend

Développé avec FastAPI. Il gère :

1. API REST
2. Base de données
3. Authentification
4. Connexion IMAP
5. Analyse phishing
6. Analyse pièces jointes

7. VirusTotal
8. Quarantaine

5.2 Frontend

Développé avec :

- HTML
- CSS
- JavaScript

Il fournit :

1. Page inscription
2. Page connexion
3. Tableau de bord
4. Inbox
5. Quarantine
6. Page détails email

6. Base de données

Le projet utilise SQL Server avec plusieurs tables.

6.1 Table users

Contient :

1. id
2. username
3. email
4. password_hash
5. created_at

Le mot de passe est stocké sous forme hashée.

6.2 Table imap_accounts

Contient :

1. id
2. user_id
3. account_email
4. app_password_encrypted
5. last_checked

Permet de lier un utilisateur à son compte Gmail IMAP.

6.3 Table emails_text

Contient :

1. id
2. imap_account_id
3. subject
4. from_address
5. to_addresses
6. text_body
7. processed
8. is_quarantined
9. date_received

6.4 Table attachments

Contient :

1. id
2. email_id
3. filename
4. content_type
5. size_bytes
6. sha256
7. verdict

8. severity
9. score

6.5 Table quarantine

Contient :

1. email_id
2. reason
3. quarantined_at
4. released

7. Fonctionnement IMAP dans le projet

La fonction fetch_gmail_imap permet :

1. Connexion au serveur imap.gmail.com
2. Authentification avec email + app password
3. Sélection du dossier INBOX
4. Lecture des messages
5. Téléchargement du contenu
6. Extraction du sujet, expéditeur, destinataire
7. Extraction du corps texte
8. Stockage en base de données

8. Processus détaillé de détection de phishing

La détection de phishing est réalisée par la fonction principale de classification d'email.

8.1 Étape 1 — Nettoyage du contenu

Le message brut est converti en texte lisible :

1. Décodage bytes vers string
2. Séparation headers et body
3. Suppression balises HTML

4. Extraction texte visible

8.2 Étape 2 — Extraction des URLs

Une série d'expressions régulières est utilisée pour détecter :

1. Liens http et https
2. Liens commençant par www
3. Domaines sans protocole
4. Liens contenant une adresse IP
5. Liens avec format trompeur (user@domain)

Les liens sont ensuite nettoyés et dédupliqués.

8.3 Étape 3 — Détection de mots suspects

Une liste de mots suspects est comparée avec le texte :

Exemples :

- urgent
- verify
- password
- account
- click here
- reset
- confirm

Si plusieurs mots suspects sont trouvés, le score de suspicion augmente.

8.4 Étape 4 — Analyse du nombre de liens

Si le nombre de liens dépasse un seuil (ex : plus de 3), l'email est marqué suspect car les emails phishing contiennent souvent plusieurs liens.

8.5 Étape 5 — Vérification VirusTotal des URLs

Chaque URL est envoyée à l'API VirusTotal.

Processus :

1. Soumission de l'URL
2. VirusTotal lance l'analyse
3. Le système attend le résultat
4. Lecture du nombre de moteurs qui détectent une menace

Si malicious_count > 0 → URL dangereuse.

8.6 Étape 6 — Analyse des pièces jointes

Chaque pièce jointe est analysée.

Vérifications :

1. Calcul du hash SHA256
2. Détection type fichier
3. Analyse du contenu ZIP
4. Recherche exécutables cachés
5. Détection macros Office
6. Détection JavaScript PDF
7. Recherche commandes suspectes texte
8. Vérification VirusTotal hash

8.7 Étape 7 — Calcul du score de risque

Un score est calculé selon :

1. Présence exécutables
2. Macros
3. Scripts
4. Résultat VirusTotal
5. Taille fichier
6. Vulnérabilités détectées

Score → verdict :

- safe
- suspicious
- malicious

8.8 Étape 8 — Mise en quarantaine

Si email est suspect :

1. is_quarantined = 1
2. insertion table quarantine
3. raison enregistrée
4. date enregistrée

9. Fonctionnalités complètes du projet

9.1 Gestion utilisateurs

1. Création compte
2. Hash mot de passe
3. Vérification mot de passe fort
4. Login sécurisé

9.2 Gestion IMAP

1. Ajout compte IMAP
2. Liaison user / Gmail
3. Fetch emails
4. Mise à jour last_checked

9.3 Gestion emails

1. Liste emails utilisateur
2. Détail email

3. Statut quarantaine
4. Marquage processed

9.4 Analyse phishing

1. Extraction texte
2. Extraction URLs
3. Détection mots suspects
4. Analyse VirusTotal
5. Score risque

9.5 Analyse pièces jointes

1. Extraction fichiers
2. Sauvegarde disque
3. Scan automatique
4. Score sécurité
5. Verdict

9.6 Quarantaine

1. Liste emails suspects
2. Raisons affichées
3. Interface dédiée
4. Isolation

9.7 Frontend

1. Signup
2. Login
3. Dashboard
4. Inbox

5. Fetch Gmail bouton
6. Details email
7. Rapport phishing affiché
8. Page quarantaine

10. Routes API principales

10.1 Utilisateurs

POST /users/create
POST /users/login

10.2 IMAP

POST /imap/add
POST /imap/fetch/{id}
GET /imap/accounts/{user_id}

10.3 Emails

GET /emails/user/{user_id}
GET /emails/detail/{id}
GET /emails/quarantine/{user_id}

10.4 Analyse

POST /analyze_email

11. Conclusion

Ce projet fournit un système complet de protection email basé sur :

1. Connexion IMAP sécurisée
2. Analyse contenu
3. Détection phishing
4. Analyse pièces jointes
5. VirusTotal

6. Scoring
7. Quarantaine
8. Interface web
9. API modulaire

Le système peut être amélioré par :

1. Machine learning
2. Sandbox fichiers
3. Alertes temps réel
4. Tableau de bord statistiques