Fiche Technique – Projet 7
Analyse forensique post-phishing

1. Objectif

Développer un **outil automatisé en Python** capable d'**analyser un email malveillant simulé** (généré localement) pour **extraire les Indicateurs de Compromission (IOCs)** critiques tels que :

- **URLs de phishing** (liens vers faux login),
- **Adresses IP sources** (présentes dans les headers),
- **Pièces jointes malveillantes** (fichiers `.exe`, `.zip`, macros Office),
- **Métadonnées** (si fichiers joints sont fournis).

Le résultat final est un **rapport technique structuré** listant les IOCs détectés avec **recommandations de mitigation**.

Pourquoi ce projet ? L'ingénierie sociale (phishing) est la **première cause d'intrusion** en entreprise (Verizon DBIR 2024). Ce projet reproduit le travail d'un **analyste SOC** confronté à un email rapporté par un employé, et permet de comprendre **comment décortiquer un email malveillant sans l'ouvrir**.

2. Topologie réseau exigée

- **VM locale** : Kali Linux ou Parrot OS
- **Aucune connexion sortante** pendant la démo finale
- **Email source** : fichier texte brut (`.eml` ou `.txt`) généré par le groupe
- **Pièces jointes** : fichiers locaux (`.pdf`, `.docx`, `.jpg`) générés par le groupe

3. Étapes détaillées (guide d'exécution autonome)

Étape 1 : Génération de l'email malveillant simulé

Le groupe crée un fichier `phishing_email.eml` contenant :

- **Headers complets** :

```
Received: from [192.168.1.100] (unknown [10.0.2.15])
     by mail.example.com with SMTP
From: "Microsoft Security" <security@m1crosoft-support.com>
To: victim@example.com
Subject: Urgent: Suspicious Login Detected
```

- **Corps HTML** avec :
  - Lien de phishing : `<a href="http://10.0.2.15/fake-login">Verify Account</a>`
  - Pièce jointe simulée : `<img src="cid:invoice.pdf">`

— **Pièce jointe** : créer un fichier `invoice.pdf` (vide ou généré localement).

Étape 2 : Développement du script `email_forensics.py`
Le script doit :

— **Parser l'email** avec la librairie `email` :

```
import email
with open("phishing_email.eml", "r") as f:
    msg = email.message_from_file(f)
```

— **Extraire les headers critiques** :
  — `Received` → liste des IPs sources,
  — `From` → adresse expéditeur (vérifier domaine suspect : `m1crosoft` au lieu de `microsoft`).

— **Analyser le corps HTML** :
  — Regex pour URLs : `r'<a+href=["]́(.*?)["]́'`,
  — Détection de mots-clés : `"urgent"`, `"verify"`, `"suspicious"`.

— **Extraire les pièces jointes** :
  — Si `msg.is_multipart()`, itérer sur les parties,
  — Sauvegarder les pièces jointes dans un dossier `attachments/`.

— **Scanner les métadonnées** (si pièce jointe est un `.jpg`/`.pdf`) :

```
import subprocess
subprocess.run(["exiftool", "-j", "attachments/invoice.pdf"])
```

Étape 3 : Génération du rapport technique
Le rapport PDF (5–8 pages) doit inclure :

— **Résumé exécutif** (1 paragraphe)
— **Méthodologie** : topologie, outils, étapes
— **Résultats** : tableau des IOCs détectés :

Type	Valeur	Risque
IP source	`10.0.2.15`	Moyen
URL phishing	`http://10.0.2.15/fake-login`	Critique
Expéditeur	`security@m1crosoft-support.com`	Élevé (typosquatting)

— **Recommandations** :
  — Former les employés à vérifier les domaines,
  — Bloquer les IPs sources en pare-feu,
  — Analyser les pièces jointes en sandbox.

— **Section éthique** :
« L'email a été **généré localement par le groupe**. Aucun email réel n'a été analysé. Conforme à la Loi 19-05. »

Étape 4 : Démo vidéo (3 min)
Montre :

- Le fichier `phishing_email.eml` (sans l'ouvrir),
- L'exécution du script `email_forensics.py`,
- Le rapport PDF généré,
- Une explication orale des **2 IOCs les plus critiques**.

4. Outils / Bibliothèques autorisés

- **Parsing email** : `email` (librairie standard Python)
- **Regex** : `re`
- **Métadonnées** : `exiftool` (appel CLI via `subprocess`)
- **Reporting** : `reportlab`
- **Génération email** : Éditeur de texte (pas d'outils externes)
- **Interdits** : Shodan, Metasploit, outils d'envoi d'email réel (`smtp`)

5. Dataset initial

- **Aucun dataset fourni** — les étudiants génèrent eux-mêmes :
  - `phishing_email.eml` (fichier texte),
  - `invoice.pdf` (fichier PDF vide ou généré localement).

6. Livrables attendus

- **Code source** :
  - `email_forensics.py` (exécutable en ligne de commande),
  - `phishing_email.eml` (fichier source d'analyse),
  - `requirements.txt` (dépendances : `reportlab`).
- **Rapport technique** : PDF anonymisé (5–8 pages)
- **Vidéo démo** : MP4 ou WebM (3 min, nommée `G7_EmailForensics_demo.mp4`)

7. Consignes éthiques impératives

> « L'email **ne doit pas être envoyé à des tiers**. Il doit être **généré localement et conservé dans le lab du groupe**. Le rapport **ne doit pas divulguer de données réelles**. **Toute violation = note 0.** »

8. FAQ interne (à ne pas poser)

- **Q** : Puis-je analyser un email reçu dans ma boîte Gmail ?
  R : **Non.** Seulement des emails **générés localement par le groupe**.

— **Q** : Faut-il utiliser `exiftool` obligatoirement ?
R : **Oui**, pour les pièces jointes. Si pas de pièce jointe, cette partie est ignorée.
— **Q** : Puis-je utiliser `BeautifulSoup` pour parser le HTML ?
R : **Oui**, mais les regex suffisent (plus léger).