



# PYTHON POUR LA CYBERSÉCURITÉ – FICHE DE RÉVISION

## 1. INTRODUCTION AU PENTEST ET À PYTHON

- **Pentest (test d'intrusion)** : simulation d'une attaque réelle pour tester la sécurité d'un SI.
  - Objectif : exploiter les vulnérabilités pour en démontrer l'impact.
  - Différence avec un **scan de vulnérabilité** : le pentest va **jusqu'à l'exploitation**.
- **Pourquoi Python ?**
  - Simple, lisible, très documenté.
  - Permet d'automatiser des tâches (scans, injections, phishing...).
  - Compatible avec de nombreuses bibliothèques : **scapy**, **nmap**, **mechanize**, **bs4**, etc.
  - Crée des attaques sans signature → difficile à détecter.

## 2. PHASE 1 – RECONNAISSANCE


### a. Réseau

- Détection d'IP/ports ouverts → **Scan de ports** TCP/UDP :
  - TCP SYN, NULL, FIN, XMAS scans.
  - UDP Scan + analyse ICMP.
- **Outils** :
  - **socket** (Python natif)
  - **nmap** (via **python-nmap**)
  - **scapy**, **dpkt** : construction/analyse de paquets
- Capture de **bannières** : identifier services exposés

### b. Web

- **Web Scraping** : extraction de données depuis HTML
  - Bibliothèques : **requests**, **BeautifulSoup**, **mechanize**
  - Techniques : user-agent rotation, proxy aléatoire, suppression de cookies
- Objectifs : collecter mails, noms, techno utilisées, failles JS visibles, fichiers exposés
- Phishing par email : collecte d'emails, hiérarchies, fournisseurs, etc.

### 3. PHASE 2 – ARMEMENT

- Préparation des outils d'attaque adaptés à la cible
- **Attaques par force brute / dictionnaire / rainbow tables**
  - Force brute : toutes les combinaisons possibles
  - Dictionnaire : mot de passe probable basé sur règles
  - Rainbow : hash → mot de passe via table précalculée
- **Accès à `/etc/passwd` et `/etc/shadow`** : extraction de comptes et hash
-  **Rançongiciel (ransomware)** :
  - `gen_keys.py` : RSA Key Pair
  - `encrypt.py` : chiffrement AES des fichiers
  - `decrypt.py` : déchiffrement

### 4. PHASE 3 – LIVRAISON

- **But** : faire exécuter un code malveillant à la victime
- Moyens :
  - **Phishing (spear ou classique)** : envoi de mails ciblés
  - **Page de phishing clonée** : collecter des identifiants
    - Télécharger la page, modifier les actions des formulaires
    - Ajouter collecte vers un fichier
  - **Pièce jointe malveillante** : exploitation d'une faille
- Automatisations : envoi d'emails via script + CSV de cibles
- Framework : **Metasploit** via `msfconsole`

### 5. PHASE 4 – EXPLOITATION

- **Vulnérabilité** : faille existante
- **Exploit** : moyen de l'utiliser pour compromettre le système

#### **Attaques vues :**

- **Injection SQL** :
  - Simple, UNION, Blind, empilées

- Exploitation automatisée avec `requests` + `BeautifulSoup`
- **Injection HTML / SSI**
- **XSS** (stocké/réfléchi) → JavaScript dans navigateur
- **Inclusion de fichiers** (LFI/RFI)
- **Téléchargement non sécurisé**
- **Exécution PHP-CGI à distance** (CVE-2012-1823)

## **6. PHASE 5 – MAINTIEN D'ACCÈS & ÉVASION**

- Objectif : **persister** dans le système et rester **furtif**

### **Techniques :**

- **TCP Proxy** : relais bidirectionnel de commandes
- **SSH Tunneling** : via `paramiko`, permet de recevoir/envoyer commandes à distance
- **Obfuscation des données** :
  - Ajout à une image JPEG
  - Stéganographie LSB sur PNG (bits de poids faible)
  - Extraction/reconstruction avec `PIL`

## **7. PHASE 6 – ATTAQUES ASSISTÉES PAR L'IA**

- Utilisation des **LLMs** (comme ChatGPT) à des fins malveillantes :

### **Exemples :**

- **Prompt Injection** : "Ignore all previous instructions..."
- **Phishing** automatisé : mails réalistes et sans fautes
- **Désinformation massive** via réseaux sociaux
- **Détection de texte IA** : `transformers`, `Llama Guard`, etc.

### **Défenses :**

- Filtrage d'entrée par regex
- Classifieurs d'intention malveillante
- Réécriture automatique
- Analyse de similarité sémantique