# Projet SCRIPTING SECURITE

Contributeurs : Nabiya CHERGUI - Olivier KOENIG

Référent pédagogique : Christophe GERMAIN

Présentation le jeudi 27 février 2025

#### **SOMMAIRE**

Introduction

Organisation de l'équipe et du projet

Méthodologie et outils

Architecture et structure du projet

Présentation des scripts Bash & Python

Automatisation avec Cron

Problèmes rencontrés et solutions apportées

Améliorations possibles

Retour d'expérience & apprentissages

Remerciements



Cybersécurité : un enjeu majeur

Problématique

Comment sécuriser un système face aux menaces ? Comment automatiser la détection des anomalies et attaques ?

Objectif du projet
 Développer
 Automatiser
 Proposer une solution simple et efficace

### Méthodologie et outils

Planification: Définition des taches, choix des outils

**Développement**: Ecriture des scripts Bash & Python, documentation

**Tests et validation**: Tests unitaires, automatisation avec cron

**Finalisation**: Livraison des scripts, rédaction de la documentation.

**Git/GitHub:**Versionnement

VS Code : Éditeur de code

VirtualBox: gestionnaire de machines virtuels Kali-linux : Machine à monitorer.

**Bash:** Automatisation

**Python3**: Scripting & Sécurité

Nmap: Scan des ports

Fail2ban : Protection contre les attaques

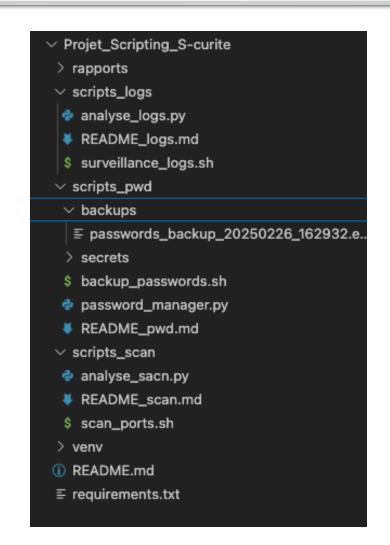
Cryptography: Chiffrementsécurisé

**Crontab :**Plannification des tâches

## Architecture et structure du projet

#### Notre objectif:

- Faciliter la navigation et l'évolution du projet
- Séparer chaque fonctionnalité
- Suivre les bonne pratiques en cyber et scripting



### Présentation des scripts -Gestion sécurisée des mots de passe

Stockage et gestion sécurisée via chiffrement

Politique de robustesse des mots de passe garantie

Automatisation de la sauvegarde des mots de passe

#### Scripts impliqués :

✓ scripts\_pwd
> backups
> secrets
\$ backup\_passwords.sh
password\_manager.py
▼ README\_pwd.md

Script Python (password\_manager.py) : Gère les mots de passe en assurant leur chiffrement, leur stockage sécurisé, et leur récupération contrôlée.

**Script Bash (backup\_passwords.sh) :** Automatise la **sauvegarde** des mots de passe chiffrés pour éviter toute **perte de données**.

 $\blacksquare$  **README\_pwd.md**  $\rightarrow$  Documentation associée.

## Fonctionnement du script Python en détail



#### Script Python avec **password\_manager.py** :

- Utilisation de la bibliothèque cryptography pour le chiffrement
- Génération de clé dans secrets/key.key
- Stockage dans secrets/passwords.enc
- Robustesse vérifiée
- Récupération sécurisée

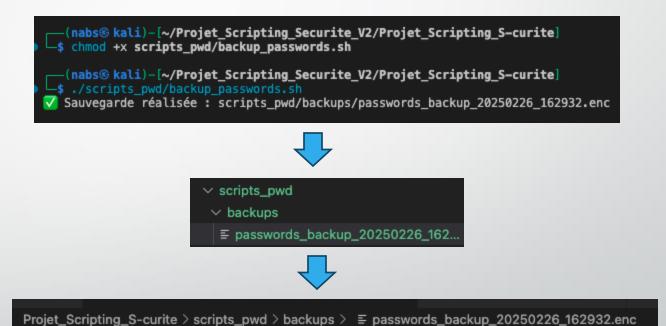




## Fonctionnement du script Bash en détail

#### Script Bash avec backup\_passwords.sh

- Vérification avec passwords.enc
- Copie et stockage dans backups/
- Récupération possible



gAAAAABnvypi2xi7XfEXm4NgMPeN8m04cSqNLfC2Rmzqj5GiNWgae4CJ8CghmloyBxrWoHq3VkOtgQSsl

### Présentation des scripts -Scan des ports et services ouverts

- Pourquoi utiliser Bash et Python ensemble ?
- Bash: Scanner les ports avec Nmap
- Le script Bash exécute nmap pour identifier les ports ouverts et les services actifs.
- Il **enregistre les résultats** dans un fichier pour une analyse ultérieure.
- Cela permet de **détecter les points d'entrée potentiels** sur un réseau.
- Python : Analyser les résultats et détecter les failles
- Python lit le rapport Nmap et extrait les services identifiés.
- Il compare les versions des services avec une base de vulnérabilités connues.
- Si un service vulnérable est détecté, il **génère une alerte** pour prévenir les risques.
- Objectif: Automatiser la détection des failles avant qu'un pirate ne les exploite!

#### ./scripts\_scan/scan\_ports.sh 192.168.1.5

- Bash: Scanner les ports avec Nmap
- Le script Bash exécute nmap pour identifier les ports ouverts et les services actifs.
- Il enregistre les résultats dans un fichier pour une analyse ultérieure.
- Cela permet de détecter les points d'entrée potentiels sur un réseau.
- générer un fichier : rapports/Bash\_scan\_result\_192\_168\_1\_5\_20250226\_202926.txt contenant les résultats du scan.

```
(olivier⊕lan-olivier)-[~/Projet_Scripting_S-curite]

$ ./scripts_scan/scan_ports.sh 192.168.1.5

Scan des ports ouverts sur 192.168.1.5...

Résultats enregistrés dans rapports/Bash_scan_result_192_168_1_5_20250226_202926.txt

Starting Nmap 7.95 ( https://nmap.org ) at 2025-02-26 20:29 CET

Nmap scan report for 192.168.1.5

Host is up (0.00080s latency).

All 1000 scanned ports on 192.168.1.5 are in ignored states.

Not shown: 1000 closed tcp ports (reset)

MAC Address: 08:00:27:B7:1D:4B (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/.

Nmap done: 1 IP address (1 host up) scanned in 0.62 seconds
```

#### ./scripts\_scan/analyse\_scan.py 192.168.1.5

- Python : Analyser les résultats et détecter les failles
- Python lit le rapport Nmap et extrait les services identifiés.
- Il compare les versions des services avec une base de vulnérabilités connues.
- Si un service vulnérable est détecté, il génère une alerte pour prévenir les risques.
- Il créera un fichier rapports/Python\_scan\_result\_192\_168\_1\_5\_20250227\_095135.txt contenant les résultats du scan.

```
-(olivier⊕lan-olivier)-[~/Projet Scripting S-curite]
└S ./scripts_scan/analyse_scan.py 192.168.1.5
🔍 Scan des ports ouverts sur : 192.168.1.5...
🔀 Patientez, le scan peut prendre quelques instants...
☑ Scan terminé ! Résultats enregistrés dans : rapports/Python_scan_result_192_168_1_5_20250227_095135.txt
Starting Nmap 7.95 ( https://nmap.org ) at 2025-02-27 09:51 CET
Nmap scan report for 192.168.1.5
Host is up (0.0019s latency).
All 1000 scanned ports on 192.168.1.5 are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: 08:00:27:B7:1D:4B (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.63 seconds
Aucun service vulnérable détecté.
```

### Présentation des scripts

Surveillance et analyse des logs de sécurité

Détection des tentatives

Identification des attaques

#### Scripts impliqués :

- ∨ scripts\_logs
- analyse\_logs.py
- README\_logs.md
- \$ surveillance\_logs.sh
- surveillance\_logs.sh (Bash) → Surveillance en direct des tentatives échouées.
- **iii** analyse\_logs.py (Python) → Analyse détaillée et rapport des IP suspectes.
- $\blacksquare$  **README\_logs.md**  $\rightarrow$  Documentation associée.

## Fonctionnement du script Bash (surveillance\_logs.sh)

#### Depuis un autre terminal :

```
(nabs® kali)-[~/Projet_Scripting_Securite_V2]

$ ssh faux_utilisateur@localhost
faux_utilisateur@localhost's password:
Permission denied, please try again.
faux_utilisateur@localhost's password:
Permission denied, please try again.
faux_utilisateur@localhost's password:
faux_utilisateur@localhost's password:
faux_utilisateur@localhost: Permission denied (publickey,password).
```

```
(nabs⊛ kali)-[~/Projet_Scripting_Securite_V2/Projet_Scripting_S-curite]
$ ./scripts_logs/surveillance_logs.sh

Surveillance en temps réel des échecs de connexion dans /var/log/auth.log

Tentative de connexion échouée : 2025-02-26T17:01:41.137770+01:00 kali sshd-session[927107]: Failed password for invalid user faux_utilisateur from ::1 port 58612 ssh2

Tentative de connexion échouée : 2025-02-26T17:01:45.693950+01:00 kali sshd-session[927107]: Failed password for invalid user faux_utilisateur from ::1 port 58612 ssh2

Tentative de connexion échouée : 2025-02-26T17:01:50.348235+01:00 kali sshd-session[927107]: Failed password for invalid user faux_utilisateur from ::1 port 58612 ssh2
```

#### Utilisation de tail -f /var/log/auth.log pour suivre les nouvelles entrées en direct.

```
(nabs@ kali) - [~/Projet_Scripting_Securite_V2/Projet_Scripting_S-curite]

$ tail -f /var/log/auth.log

2025-02-26T17:01:45.693950+01:00 kali sshd-session[927107]: Failed password for invalid user faux_utilisateur from ::1 port 58612 ssh2

2025-02-26T17:01:48.847850+01:00 kali sshd-session[927107]: pam_unix(sshd:auth): check pass; user unknown

2025-02-26T17:01:48.847995+01:00 kali sshd-session[927107]: pam_winbind(sshd:auth): getting password (0x00000388)

2025-02-26T17:01:48.848023+01:00 kali sshd-session[927107]: pam_winbind(sshd:auth): pam_get_item returned a password

2025-02-26T17:01:50.348235+01:00 kali sshd-session[927107]: Failed password for invalid user faux_utilisateur from ::1 port 58612 ssh2

2025-02-26T17:01:51.827654+01:00 kali sshd-session[927107]: Connection closed by invalid user faux_utilisateur ::1 port 58612 [preauth]

2025-02-26T17:01:51.827654+01:00 kali sshd-session[927107]: PAM 2 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=::1

2025-02-26T17:01:51.830234+01:00 kali sshd[718]: srclimit_penalise: ipv6: new ::1/128 deferred penalty of 5 seconds for penalty: failed authentication penalty: failed authentication
```

## Fonctionnement du script Python (analyse\_logs.py)

Rendre éxecutable le script si python3 non installé

```
(nabs@ kali)-[~/Projet_Scripting_Securite_V2/Projet_Scripting_S-curite]
$ chmod +x scripts_logs/analyse_logs.py
```



Lancer la lecture et l'analyse des logs pour générer le rapport

```
_____(nabs⊗ kali)-[~/Projet_Scripting_Securite_V2/Projet_Scripting_S-curite]
_$ ./scripts_logs/analyse_logs.py
✓ Analyse terminée. Rapport généré : rapports/analyse_logs_report.txt
```



Génération du rapport dans le dossier rapports/

```
    ✓ Projet_Scripting_S-curite
    ✓ rapports
    ≡ analyse_logs_report.txt
```

Grace à Fail<sub>2</sub>Ban, adresses IP bannies au bout de 3 tentatives échouées pendant 10 minutes :

```
(nabs⊛ kali) - [~/Projet_Scripting_Securite_V2/Projet_Scripting_S-curite]
$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
| |- Currently failed: 0
| |- Total failed: 0
| `- Journal matches: _SYSTEMD_UNIT=ssh.service + _COMM=sshd
`- Actions
|- Currently banned: 1
|- Total banned: 1
|- Total banned: 1
|- Banned IP list: 192.168.1.140
```

Rapport généré avec les IP suspectes et leur nombre de tentatives

#### Automatisation avec Cron

Bien que par défaut, il soit souvent installé, voici les commandes pour l'installer sur les différentes distributions : Installation Debian et Ubuntu :

> sudo apt install cron

Pour afficher le contenu du fichier crontab :

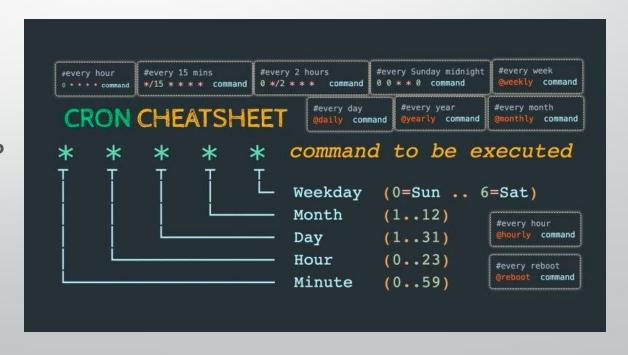
> crontab -1

Pour supprimer toutes les actions du fichier **crontab** 

> crontab -r

Pour éditer les actions du fichier crontab

crontab -e



#### Analyse du fichier crontab

#### BASH - Tâches exécutées à 02h00 du matin :

#### Scan des ports

0 2 \* \* \* bash ~/Projet\_Scripting\_Securite/scripts\_scan/scan\_ports.sh 192.168.1.1 Analyse les ports de l'adresse IP 192.168.1.1.

#### Surveillance des logs

0 2 \* \* \* bash ~/Projet\_Scripting\_Securite/scripts\_logs/surveillance\_logs.sh Exécute un script qui surveille les logs (journal des événements).

#### Sauvegarde des mots de passe

0 2 \* \* \* bash ~/Projet\_Scripting\_Securite/scripts\_pwd/backup-passwords.sh Effectue une sauvegarde des mots de passe.

#### PYTHON - Tâches exécutées à 08h30 du matin :

#### ✓ Analyse des scans

30 8 \* \* \* python3 ~/Projet\_Scripting\_Securite/scripts/analyse\_scan.py
Exécute un script Python qui analyse un scan (probablement généré par le scan des ports à 02h00).

#### Analyse des logs

30 8 \* \* \* python3 ~/Projet\_Scripting\_Securite/scripts\_logs/analyse\_logs.py Analyse les logs du système.

#### Gestionnaire de mots de passe

30 8 \* \* \* python3 ~/Projet\_Scripting\_Securite/scripts\_pwd/password\_manager.py Lance un gestionnaire de mots de passe.



```
GNU nano 8.3 /tmp/crontab.ds
0 2 * * * bash ~/Projet_Scripting_Securite/scripts_scan/scan_ports.sh 192.168.1.1
2 * * * bash ~/Projet_Scripting_Securite/scripts_logs/surveillance_logs.sh
0 2 * * * bash ~/Projet_Scripting_Securite/scripts_pwd/backup-passwords.sh
30 8 * * * python3 ~/Projet_Scripting_Securite/scripts/analyse_scan.py
30 8 * * * python3 ~/Projet_Scripting_Securite/scripts_logs/analyse_logs.py
30 8 * * * python3 ~/Projet_Scripting_Securite/scripts_pwd/password_manager.py
```

## Problèmes rencontrés et solutions apportées

- Fichier auth.log introuvable
- Pas de logs SSH disponibles
- ✓ Activation de la journalisation
- Droits administrateur insuffisants
- **Commandes bloquées sans** sudo
- √ Ajout utilisateur au groupe sudo
- Problème de push GitHub
- Conflit empêchant le git push
- ✓ git pull --rebase pour résoudre





- Renforcer la sécurité
- Intégration d'un pare-feu (UFW, iptables)
- Amélioration de la **politique de mots de passe**



- Automatisation avancée
- Ajout de notifications (email, Slack) pour alertes
- Génération de rapports détaillés en PDF



- Tests & évolutions
- Simulation d'attaques avec Hydra & Nikto
- Compatibilité avec d'autres systèmes Linux

## Améliorations possibles

Retour d'expérience & apprentissages

Approfondissement des outils de cybersécurité Montée en compétences sur l'automatisation

 GitHub & travail en équipe  Résolution de problèmes & autonomie

### Remerciements 🙏



#### Nous tenons à remercier :



- Notre formateur Christophe pour son accompagnement et ses précieux conseils.
- Nos camarades de classe pour leurs échanges et leur entraide.
- Notre binôme pour la collaboration efficace et la répartition des tâches.
- Les ressources en ligne et la documentation officielle qui nous ont aidés à surmonter les obstacles.
- Merci pour votre attention! Nous sommes disponibles pour vos questions