RUNBOOK — MSPR_scan / `scanner.py`

Sommaire

1) Résumé rapide

- Emplacement : `MSPR_scan/scripts/` (ou `MSPR_scan/Script/`)
- Script principal : `scripts/scanner.py`
- Sorties : `scripts/reports/*.json` (ne pas committer les rapports)

2) Prérequis

- **Python 3.8+**
- (Optionnel) **Nmap** binaire + `python-nmap` pour détection services/OS.
- Accès GitHub + droits pour créer/merger PR (si vous utilisez l'UI).
- Recommandé : `jq` pour lire JSON en local.

3) Installation (rapide)

Windows (PowerShell)

```
 # aller dans le repo (adapter le chemin)
Set-Location "C:\chemin\vers\MSPR_1\MSPR_scan"

# créer + activer venv
python -m venv .venv
.\.venv\Scripts\Activate.ps1
# si bloqué :
# Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass

# pip
python -m pip install --upgrade pip
pip install -r scripts/requirements.txt   # si présent

# créer dossier rapports
New-Item -ItemType Directory -Force scripts\reports
```

Linux / macOS

```
# aller dans le repo
cd /chemin/vers/MSPR_1/MSPR_scan

# venv + activation
python3 -m venv .venv
source .venv/bin/activate

# pip
pip install --upgrade pip
pip install -r scripts/requirements.txt   # si présent

# dossier rapports
mkdir -p scripts/reports
```

---

4) Installer Nmap (optionnel, recommandé)

Debian / Ubuntu

```
sudo apt update
sudo apt install -y nmap
pip install python-nmap
```

macOS (Homebrew)

```
brew install nmap
pip install python-nmap
```

Windows

- Télécharger/installer Nmap depuis https://nmap.org/download.html
- Ensuite : `pip install python-nmap` dans le venv.

> Le script fonctionne **sans** nmap : fallback TCP multithread. `--use-nmap` active nmap si disponible.

---

5) Quickstart — test de fonctionnement

Activez le venv, puis lancez un scan simple :

```
# venv activé
python scripts/scanner.py --hosts 127.0.0.1
```

Vérifier qu'un fichier `scripts/reports/report_*.json` a été généré.

---

6) Commandes d'utilisation — exemples

Afficher l'aide :

```
python scripts/scanner.py --help
```

Scan simple (fallback TCP) :

```
python scripts/scanner.py --hosts 127.0.0.1
```

Scanner plusieurs hôtes :

```
python scripts/scanner.py --hosts 10.0.0.5 10.0.0.6 192.168.1.10
```

Scanner depuis un inventaire JSON :

```
python scripts/scanner.py --inventory scripts/inventory.json
```

Scanner un CIDR (avec nmap si dispo) :

```
python scripts/scanner.py --cidr 192.168.1.0/24 --use-nmap
```

Options avancées (exemple) :

```
python scripts/scanner.py --hosts 10.0.0.5 \
  --ports 22 80 443 \
  --timeout 1.5 \
  --workers 50 \
  --wan-probe 1.1.1.1 \
  --report-name myreport
```

---

7) Options utiles (CLI)

- `--hosts` (nargs="+") : liste d'IP/hostnames (mutuellement exclusif avec `--inventory` et `--cidr`)
- `--inventory` : chemin vers JSON (liste ou `{"hosts":[...]}`)
- `--cidr` : plage CIDR (ex. `192.168.1.0/24`)
- `--use-nmap` : activer `python-nmap` si disponible
- `--ports` : liste de ports pour fallback (ex. `--ports 22 80 443`)
- `--timeout` : timeout socket (secondes)
- `--workers` : nombre de threads pour fallback TCP
- `--wan-probe` : IP pour mesurer latence WAN (ex. `8.8.8.8`)
- `--report-name` : nom du fichier report (sans extension)

---

8) Vérifier les rapports (validation)

Lister :

```
ls scripts/reports
```

Inspecter avec `jq` :

```
jq '.' scripts/reports/report_*.json | less
```

Champs essentiels à vérifier :

- `meta.generated_at`
- `meta.source`
- `meta.tool_version` (si vous ajoutez)
- `results` (liste d'hôtes avec ports et états)

Sans `jq` :

```
grep -n '"generated_at"\|"source"\|"results"' scripts/reports/*.json || true
```

---

9) Versioning minimal (procédure simple)

1. Ajouter numéro de version dans `scripts/scanner.py` (en haut) :

```
__version__ = "0.1.0"
```

2. Inclure `tool_version` dans `meta` du rapport (si possible).
3. Workflow (UI) simple :
   - Créer branche via UI (`Create branch`) ou local : `git checkout -b feat/xxx`
   - Modifier -> Commit -> Push -> Ouvrir PR -> Merge.
4. Après merge, créer release/tag via GitHub UI (ex. `v0.1.0`) si nécessaire.

10) Automatisation simple

Cron (Linux) — exécution quotidienne 02:00

```
0 2 * * * cd /chemin/MSPR_1/MSPR_scan && /chemin/.venv/bin/python scripts/scanner.py --inventory scripts/inventory.json >> /var/log/
```

systemd (one-shot)

Fichier `/etc/systemd/system/mspr-scan.service` :

```
[Unit]
Description=MSPR scanner

[Service]
Type=oneshot
WorkingDirectory=/chemin/MSPR_1/MSPR_scan
ExecStart=/chemin/MSPR_1/MSPR_scan/.venv/bin/python scripts/scanner.py --inventory scripts/inventory.json
```

Puis :

```
sudo systemctl daemon-reload
sudo systemctl start mspr-scan.service
```

---

11) Upload post-run (optionnel)

Si vous devez envoyer le JSON à un endpoint (Nester) :

curl (bash)

```
curl -X POST -H "Authorization: Bearer $NESTER_TOKEN" -F "file=@scripts/reports/report_NAME.json" "$NESTER_URL"
```

Python (requests)

```
import os, requests
NESTER_URL = os.getenv("NESTER_URL")
NESTER_TOKEN = os.getenv("NESTER_TOKEN")
with open("scripts/reports/report_NAME.json","rb") as f:
    headers = {"Authorization": f"Bearer %s" % NESTER_TOKEN}
    r = requests.post(NESTER_URL, files={"file": f}, headers=headers, timeout=30)
    print(r.status_code, r.text)
```

> **Important** : ne **jamais** stocker de token dans le repo. Utiliser variables d'environnement / secret manager.

---

12) `.gitignore` recommandé (ajouter à la racine)

```
# environnement
.venv/
.vscode/

# rapports générés
scripts/reports/

# backups
*.bak
*.prepatch

# OS
.DS_Store
Thumbs.db
```

---

13) Nettoyage des backups committés

Si des `*.bak` ou `*.prepatch` ont déjà été committés :

- Via UI GitHub : supprimer fichier -> commit -> PR -> merge
- Ou local :

```
git rm --cached scripts/*.bak
git rm --cached scripts/*.prepatch
git commit -m "chore: remove backup files"
git push
```

Ajouter `*.bak` et `*.prepatch` au `.gitignore`.

---

14) Dépannage rapide

- **Pas de rapport** : vérifier `scripts/reports/` et droits d'écriture.
- **venv activation bloquée (PowerShell)** : `Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass`
- **nmap absent** : installer binaire + `pip install python-nmap` — fallback TCP toujours possible.
- **avg_ping_ms() retourne None** : parsing dépend de la locale ; utiliser `--wan-probe`.
- **Fichiers .bak gênants** : retirer et `.gitignore`.

---

15) Checklist avant PR / livraison

- ☐ `scanner.py` testé localement (au moins 1 run)
- ☐ Rapport JSON valide présent dans `scripts/reports/` (pour test)
- ☐ `meta.tool_version` présent (si vous suivez le versioning)
- ☐ `.gitignore` mis en place et mergé
- ☐ Backups (`*.bak` , `*.prepatch`) retirés du repo
- ☐ `RUNBOOK.md` (ou `README`) ajouté & mergé
- ☐ (Optionnel) Release/tag `v0.x.x` créé

---

16) Notes sécurité & bonnes pratiques

- Scanner **uniquement** des cibles autorisées.
- Ne pas committer de secrets / tokens.
- Utiliser PRs + reviews pour changements critiques.
- Variables d'environnement pour configurations sensibles.
