LABORATORY - OpenVAS

OpenVAS : L'Oeil de Sauron sur le Réseau

OpenVAS (Open Vulnerability Assessment System), aujourd'hui cœur de la **Greenbone Community Edition**, est un scanner de vulnérabilités open-source.

Contrairement à un outil de détection comme **Nmap** (qui se contente de frapper aux portes pour voir qui répond), OpenVAS est un auditeur impitoyable. Il ne se demande pas seulement "Le port 445 est-il ouvert ?", mais il tente d'interagir avec le service pour déterminer :

- Sa version exacte (Fingerprinting).
- S'il est mal configuré (Ex: mot de passe par défaut).
- S'il contient des failles connues (CVE - Common Vulnerabilities and Exposures).

> **Analogie :** Si Nmap est un vigile qui vérifie si les fenêtres sont fermées, OpenVAS est un cambrioleur qui teste la solidité de la vitre avec un pied-de-biche (virtuel).

Rôle dans la Cyberdéfense

Dans une infrastructure, OpenVAS occupe le rôle de **"Vulnerability Management"**. Son but est de réduire la surface d'attaque avant qu'un attaquant ne l'exploite.

Il automatise trois phases critiques :

- **Reconnaissance Active :** Cartographie des IP et des ports ouverts.
- **Énumération :** Identification des OS et des services (bannières).
- **Analyse de Vulnérabilité :** Corrélation entre les services détectés et une base de données de +170 000 signatures (NVT).

Architecture Technique (Micro-services)

L'implémentation choisie sur **Debian 13 - Trixie** repose sur une architecture conteneurisée (Docker). Cette approche permet de segmenter les fonctions critiques :

Composant	Rôle Technique
gvmd (Manager)	Orchestrateur central. Il gère la base de données, les utilisateurs et pilote les scans.
ospd-openvas (Scanner)	Moteur d'exécution. Il forge les paquets TCP/IP et exécute les scripts NVT contre les cibles.
gsa (Interface)	*Greenbone Security Assistant*. Interface Web (HTTPS) pour l'administration et le reporting.
pg-gvm (PostgreSQL)	Stockage persistant des configurations et résultats.
redis (Cache)	Base de données en mémoire pour les données volatiles durant le scan.

PROCÉDURE D'INSTALLATION & DÉPLOIEMENT

Installation du Moteur de Conteneurisation

Avant de déployer la solution, on doit préparer le socle Docker avec les dépôts officiels pour garantir la stabilité des versions.

Installation des dépendances

```
thomas@legion:~$ sudo apt update && sudo apt install ca-certificates curl gnupg
```

Ajout de la clé de sécurité GPG Docker

```
thomas@legion:~$ sudo install -m 0755 -d /etc/apt/keyrings
```

```
thomas@legion:~$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
```

```
thomas@legion:~$ sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

Configuration du dépôt (Repository)

```
thomas@legion:~$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/debian bookworm stable" | sudo tee /etc/apt/sources.list.d/docker.list >
/dev/null
```

Installation des binaires Docker

```
thomas@legion:~$ sudo apt update && sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
Hit:1 http://deb.debian.org/debian trixie InRelease
Hit:2 http://security.debian.org/debian-security trixie-security InRelease
Hit:3 http://deb.debian.org/debian trixie-updates InRelease
Get:4 https://download.docker.com/linux/debian bookworm InRelease [46.6 kB]
Get:5 https://repo.steampowered.com/steam stable InRelease [3,622 B]
Get:6 https://download.docker.com/linux/debian bookworm/stable amd64 Packages [61.3 kB]
Get:7 https://apt.syncthing.net syncthing InRelease [24.2 kB]
Fetched 136 kB in 0s (422 kB/s)
15 packages can be upgraded. Run 'apt list --upgradable' to see them.
Installing:
  containerd.io  docker-buildx-plugin  docker-ce  docker-ce-cli  docker-compose-plugin

Installing dependencies:
  docker-ce-rootless-extras  pigz

Suggested packages:
  cgroupfs-mount  | cgroup-lite  docker-model-plugin

Summary:
  Upgrading: 0, Installing: 7, Removing: 0, Not Upgrading: 15
  Download size: 96.6 MB
  Space needed: 391 MB / 877 GB available

Continue? [Y/n]
```

```
thomas@legion:~$ docker -v
Docker version 29.2.1, build a5c7197
```

Déploiement de la Stack Greenbone

Conformément à la documentation officielle Greenbone (*Greenbone Community Containers 22.4*), on procède à la création manuelle du manifeste de déploiement. Cette méthode offre un contrôle total sur les services déclarés.

Préparation de l'environnement de travail

On crée un répertoire dédié pour isoler les fichiers de configuration

```
thomas@legion:~$ sudo mkdir -p /opt/docker/openvas
thomas@legion:~$ cd /opt/docker/openvas/
```

Création du fichier de composition

```
thomas@legion:/opt/docker/openvas$ sudo nano compose.yml
```

```
name: greenbone-community-edition

services:
  vulnerability-tests:
    image: registry.community.greenbone.net/community/vulnerability-tests
    environment:
      FEED_RELEASE: "24.10"
      KEEP_ALIVE: 1
    volumes:
      - vt_data_vol:/mnt

  notus-data:
    image: registry.community.greenbone.net/community/notus-data
    environment:
      KEEP_ALIVE: 1
    volumes:
      - notus_data_vol:/mnt

  scap-data:
    image: registry.community.greenbone.net/community/scap-data
    environment:
      KEEP_ALIVE: 1
    volumes:
      - scap_data_vol:/mnt

  cert-bund-data:
    image: registry.community.greenbone.net/community/cert-bund-data
    environment:
      KEEP_ALIVE: 1
    volumes:
      - cert_data_vol:/mnt

  dfn-cert-data:
    image: registry.community.greenbone.net/community/dfn-cert-data
    environment:
      KEEP_ALIVE: 1
    volumes:
      - cert_data_vol:/mnt
    depends_on:
      cert-bund-data:
        condition: service_healthy

  data-objects:
    image: registry.community.greenbone.net/community/data-objects
    environment:
      FEED_RELEASE: "24.10"
      KEEP_ALIVE: 1
    volumes:
      - data_objects_vol:/mnt

  report-formats:
    image: registry.community.greenbone.net/community/report-formats
```

```
    environment:
      FEED_RELEASE: "24.10"
      KEEP_ALIVE: 1
    volumes:
      - data_objects_vol:/mnt
    depends_on:
      data-objects:
        condition: service_healthy

  gpg-data:
    image: registry.community.greenbone.net/community/gpg-data
    volumes:
      - gpg_data_vol:/mnt

  redis-server:
    image: registry.community.greenbone.net/community/redis-server
    restart: on-failure
    volumes:
      - redis_socket_vol:/run/redis/

  pg-gvm:
    image: registry.community.greenbone.net/community/pg-gvm:stable
    restart: on-failure:10
    volumes:
      - psql_data_vol:/var/lib/postgresql
      - psql_socket_vol:/var/run/postgresql
    depends_on:
      pg-gvm-migrator:
        condition: service_completed_successfully

  pg-gvm-migrator:
    image: registry.community.greenbone.net/community/pg-gvm-migrator:stable
    restart: no
    volumes:
      - psql_data_vol:/var/lib/postgresql
      - psql_socket_vol:/var/run/postgresql

  gvmd:
    image: registry.community.greenbone.net/community/gvmd:stable
    restart: on-failure
    volumes:
      - gvmd_data_vol:/var/lib/gvm
      - scap_data_vol:/var/lib/gvm/scap-data/
      - cert_data_vol:/var/lib/gvm/cert-data
      - data_objects_vol:/var/lib/gvm/data-objects/gvmd
      - vt_data_vol:/var/lib/openvas/plugins
      - psql_data_vol:/var/lib/postgresql
      - gvmd_socket_vol:/run/gvmd
      - ospd_openvas_socket_vol:/run/ospd
      - psql_socket_vol:/var/run/postgresql
    depends_on:
      pg-gvm:
        condition: service_started
      scap-data:
        condition: service_healthy
      cert-bund-data:
        condition: service_healthy
      dfn-cert-data:
        condition: service_healthy
      data-objects:
        condition: service_healthy
      report-formats:
        condition: service_healthy

  gsa:
    image: registry.community.greenbone.net/community/gsa:stable
    restart: on-failure
    ports:
      - 127.0.0.1:9392:80
    volumes:
      - gvmd_socket_vol:/run/gvmd
    depends_on:
      gvmd:
        condition: service_started
  # Sets log level of openvas to the set LOG_LEVEL within the env
  # and changes log output to /var/log/openvas instead /var/log/gvm
  # to reduce likelyhood of unwanted log interferences
  configure-openvas:
    image: registry.community.greenbone.net/community/openvas-scanner:stable
    volumes:
      - openvas_data_vol:/mnt
```

```
      - openvas_log_data_vol:/var/log/openvas
    command:
      - /bin/sh
      - -c
      - |
        printf "table_driven_lsc = yes\nopenvasd_server = http://openvasd:80\n" > /mnt/openvas.conf
        sed "s/127/128/" /etc/openvas/openvas_log.conf | sed 's/gvm/openvas/' > /mnt/openvas_log.conf
        chmod 644 /mnt/openvas.conf
        chmod 644 /mnt/openvas_log.conf
        touch /var/log/openvas/openvas.log
        chmod 666 /var/log/openvas/openvas.log

  # shows logs of openvas
  openvas:
    image: registry.community.greenbone.net/community/openvas-scanner:stable
    restart: on-failure
    volumes:
      - openvas_data_vol:/etc/openvas
      - openvas_log_data_vol:/var/log/openvas
    command:
      - /bin/sh
      - -c
      - |
        cat /etc/openvas/openvas.conf
        tail -f /var/log/openvas/openvas.log
    depends_on:
      configure-openvas:
        condition: service_completed_successfully

  openvasd:
    image: registry.community.greenbone.net/community/openvas-scanner:stable
    restart: on-failure
    environment:
      # `service_notus` is set to disable everything but notus,
      # if you want to utilize openvasd directly, remove `OPENVASD_MODE`
      OPENVASD_MODE: service_notus
      GNUPGHOME: /etc/openvas/gnupg
      LISTENING: 0.0.0.0:80
    volumes:
      - openvas_data_vol:/etc/openvas
      - openvas_log_data_vol:/var/log/openvas
      - gpg_data_vol:/etc/openvas/gnupg
      - notus_data_vol:/var/lib/notus
    # enable port forwarding when you want to use the http api from your host machine
    # ports:
    #   - 127.0.0.1:3000:80
    depends_on:
      vulnerability-tests:
        condition: service_healthy
      notus-data:
        condition: service_healthy
      configure-openvas:
        condition: service_completed_successfully
      gpg-data:
        condition: service_completed_successfully
    networks:
      default:
        aliases:
          - openvasd

  ospd-openvas:
    image: registry.community.greenbone.net/community/ospd-openvas:stable
    restart: on-failure
    hostname: ospd-openvas.local
    cap_add:
      - NET_ADMIN # for capturing packages in promiscuous mode
      - NET_RAW # for raw sockets e.g. used for the boreas alive detection
    security_opt:
      - seccomp=unconfined
      - apparmor=unconfined
    command:
      [
        "ospd-openvas",
        "-f",
        "--config",
        "/etc/gvm/ospd-openvas.conf",
        "--notus-feed-dir",
        "/var/lib/notus/advisories",
        "-m",
        "666",
      ]
```

```
    volumes:
      - gpg_data_vol:/etc/openvas/gnupg
      - vt_data_vol:/var/lib/openvas/plugins
      - notus_data_vol:/var/lib/notus
      - ospd_openvas_socket_vol:/run/ospd
      - redis_socket_vol:/run/redis/
      - openvas_data_vol:/etc/openvas/
      - openvas_log_data_vol:/var/log/openvas
    depends_on:
      redis-server:
        condition: service_started
      gpg-data:
        condition: service_completed_successfully
      configure-openvas:
        condition: service_completed_successfully
      vulnerability-tests:
        condition: service_healthy
      notus-data:
        condition: service_healthy

  gvm-tools:
    image: registry.community.greenbone.net/community/gvm-tools
    volumes:
      - gvmd_socket_vol:/run/gvmd
      - ospd_openvas_socket_vol:/run/ospd
    depends_on:
      - gvmd
      - ospd-openvas

volumes:
  gpg_data_vol:
  scap_data_vol:
  cert_data_vol:
  data_objects_vol:
  gvmd_data_vol:
  psql_data_vol:
  vt_data_vol:
  notus_data_vol:
  psql_socket_vol:
  gvmd_socket_vol:
  ospd_openvas_socket_vol:
  redis_socket_vol:
  openvas_data_vol:
  openvas_log_data_vol:
```

Le contenu exact du fichier YAML (Services vulnerability-tests, gvmd, ospd-openvas, pg-gvm, etc.) correspond à la configuration standard documentée par Greenbone et consultable ici (https://greenbone.github.io/docs/latest/22.4/container/index.html)

Lancement de l'infrastructure

On instancie les conteneurs en spécifiant le fichier et le nom du projet pour un suivi clair

```
thomas@legion:/opt/docker/openvas$ sudo docker compose -f compose.yml -p greenbone-community up -d
[+] up 167/168
 ✔ Image registry.community.greenbone.net/community/report-formats         Pulled
1.5ss
 ✔ Image registry.community.greenbone.net/community/data-objects           Pulled
1.3ss
 ✔ Image registry.community.greenbone.net/community/cert-bund-data         Pulled
7.9ss
 ✔ Image registry.community.greenbone.net/community/gsa:stable             Pulled
8.3ss
 ✔ Image registry.community.greenbone.net/community/openvas-scanner:stable Pulled
14.3s
 ✔ Image registry.community.greenbone.net/community/gvmd:stable            Pulled
29.9s
 ✔ Image registry.community.greenbone.net/community/scap-data              Pulled
51.6s
 ✔ Image registry.community.greenbone.net/community/gvm-tools              Pulled
13.5s
 ✔ Image registry.community.greenbone.net/community/pg-gvm:stable          Pulled
20.2s
 ✔ Image registry.community.greenbone.net/community/dfn-cert-data          Pulled
3.9ss
 ✔ Image registry.community.greenbone.net/community/ospd-openvas:stable    Pulled
25.1s
 ✔ Image registry.community.greenbone.net/community/gpg-data               Pulled
4.4ss
 ✔ Image registry.community.greenbone.net/community/pg-gvm-migrator:stable Pulled
27.0s
 ✔ Image registry.community.greenbone.net/community/notus-data             Pulled
13.0s
 ✔ Image registry.community.greenbone.net/community/vulnerability-tests    Pulled
75.6s
 ✔ Image registry.community.greenbone.net/community/redis-server           Pulled
7.8ss
 ✔ Network greenbone-community_default                                     Created
0.0s
 ✔ Volume greenbone-community_redis_socket_vol                             Created
0.0s
 ✔ Volume greenbone-community_cert_data_vol                                Created
0.0s
 ✔ Volume greenbone-community_scap_data_vol                                Created
0.0s
 ✔ Volume greenbone-community_psql_data_vol                                Created
0.0s
 ✔ Volume greenbone-community_psql_socket_vol                              Created
0.0s
 ✔ Volume greenbone-community_data_objects_vol                             Created
0.0s
 ✔ Volume greenbone-community_gvmd_data_vol                                Created
0.0s
 ✔ Volume greenbone-community_ospd_openvas_socket_vol                      Created
0.0s
 ✔ Volume greenbone-community_openvas_data_vol                             Created
0.0s
 ✔ Volume greenbone-community_notus_data_vol                               Created
0.0s
 ✔ Volume greenbone-community_openvas_log_data_vol                         Created
0.0s
 ✔ Volume greenbone-community_gpg_data_vol                                 Created
0.0s
 ✔ Volume greenbone-community_vt_data_vol                                  Created
0.0s
 ✔ Volume greenbone-community_gvmd_socket_vol                              Created
0.0s
 ✔ Container greenbone-community-notus-data-1                              Healthy
7.0s
  ... 17 more
```

Le système procède alors au téléchargement ("Pull") de toutes les images nécessaires, incluant redis, mqtt, et les différents volumes de données (NVT, SCAP, CERT) ...

Post-Installation & Configuration Initiale

Une fois les conteneurs démarrés, des actions manuelles sont requises pour rendre le service opérationnel.

Vérification de l'état des services

On s'assure que tous les conteneurs sont en statut Up ou Up (healthy)

```
thomas@legion:/opt/docker/openvas$ sudo docker compose -p greenbone-community ps
NAME                                    IMAGE
COMMAND                  SERVICE             CREATED       STATUS                 PORTS
greenbone-community-cert-bund-data-1    registry.community.greenbone.net/community/cert-bund-data
"/bin/init.sh"           cert-bund-data        2 hours ago   Up 2 hours (healthy)
greenbone-community-data-objects-1      registry.community.greenbone.net/community/data-objects
"/bin/init.sh"           data-objects          2 hours ago   Up 2 hours (healthy)
greenbone-community-dfn-cert-data-1     registry.community.greenbone.net/community/dfn-cert-data
"/bin/init.sh"           dfn-cert-data         2 hours ago   Up 2 hours (healthy)
greenbone-community-gsa-1               registry.community.greenbone.net/community/gsa:stable
"/usr/local/bin/entr…"   gsa                   2 hours ago   Up 2 hours             127.0.0.1:9392->80/tcp
greenbone-community-gvmd-1              registry.community.greenbone.net/community/gvmd:stable
"/usr/local/bin/entr…"   gvmd                  2 hours ago   Up 2 hours
greenbone-community-notus-data-1        registry.community.greenbone.net/community/notus-data
"/bin/init.sh"           notus-data            2 hours ago   Up 2 hours (healthy)
greenbone-community-openvas-1           registry.community.greenbone.net/community/openvas-scanner:stable
"/bin/sh -c 'cat /et…"   openvas               2 hours ago   Up 2 hours
greenbone-community-openvasd-1          registry.community.greenbone.net/community/openvas-scanner:stable
"/bin/sh -c /usr/loc…"   openvasd              2 hours ago   Up 2 hours
greenbone-community-ospd-openvas-1      registry.community.greenbone.net/community/ospd-openvas:stable
"/usr/bin/tini -- /u…"   ospd-openvas          2 hours ago   Up 2 hours
greenbone-community-pg-gvm-1            registry.community.greenbone.net/community/pg-gvm:stable
"/usr/local/bin/entr…"   pg-gvm               2 hours ago   Up 2 hours
greenbone-community-redis-server-1      registry.community.greenbone.net/community/redis-server
"docker-entrypoint.s…"   redis-server          2 hours ago   Up 2 hours             6379/tcp
greenbone-community-report-formats-1    registry.community.greenbone.net/community/report-formats
"/bin/init.sh"           report-formats        2 hours ago   Up 2 hours (healthy)
greenbone-community-scap-data-1         registry.community.greenbone.net/community/scap-data
"/bin/init.sh"           scap-data             2 hours ago   Up 2 hours (healthy)
greenbone-community-vulnerability-tests-1   registry.community.greenbone.net/community/vulnerability-tests
"/bin/init.sh"           vulnerability-tests   2 hours ago   Up 2 hours (healthy)
```

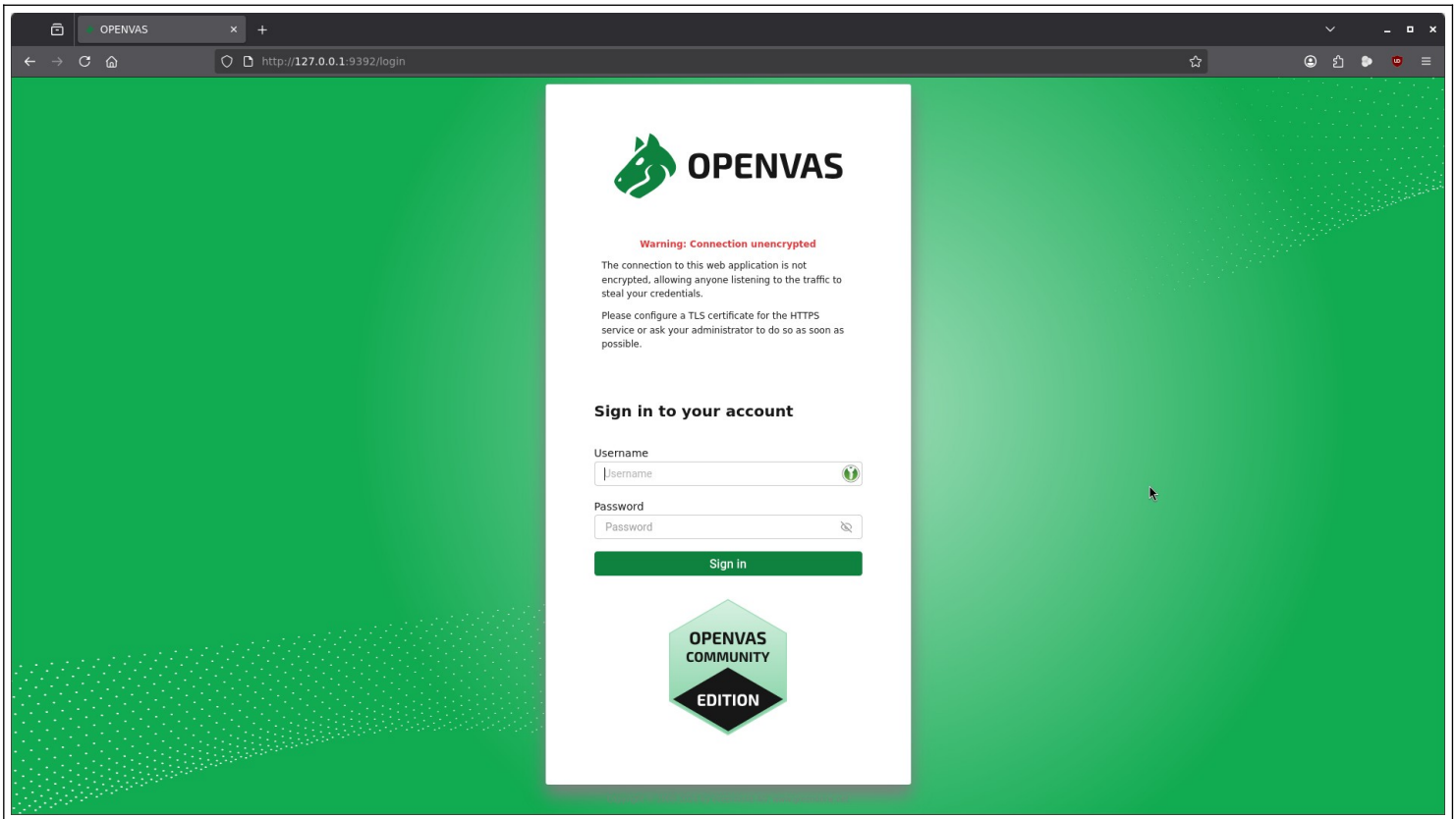
Définition des accès Administrateur

Le mot de passe par défaut n'étant pas toujours défini ou sécurisé, on force la modification du mot de passe pour l'utilisateur admin directement via le conteneur Manager (gvmd)

```
thomas@legion:/opt/docker/openvas$ sudo docker exec -u gvm greenbone-community-gvmd-1 gvmd --user=admin --new-password='p@ssw0rd'
```

Accès à l'interface

Le service est désormais accessible via l'interface web sécurisée



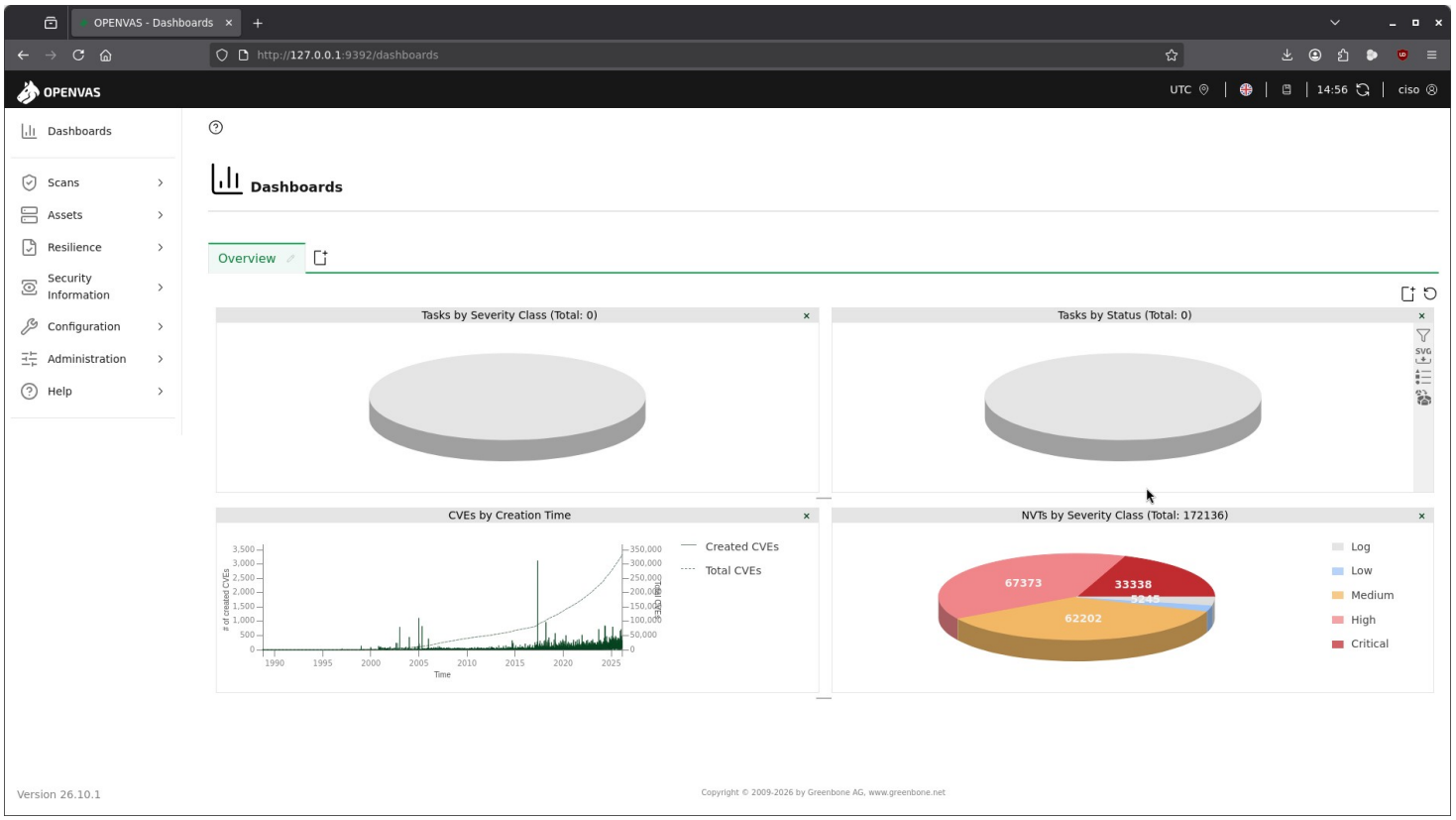
EXÉCUTION D'UN AUDIT DE VULNÉRABILITÉ

L'audit de sécurité dans OpenVAS ne se lance pas au hasard. Il suit une logique séquentielle précise :

- **Vérification > Inventaire > Ciblage > Exécution**.

Validation de la Base de Connaissances

Avant toute action offensive, l'auditeur doit s'assurer que le scanner dispose des signatures d'attaque les plus récentes.

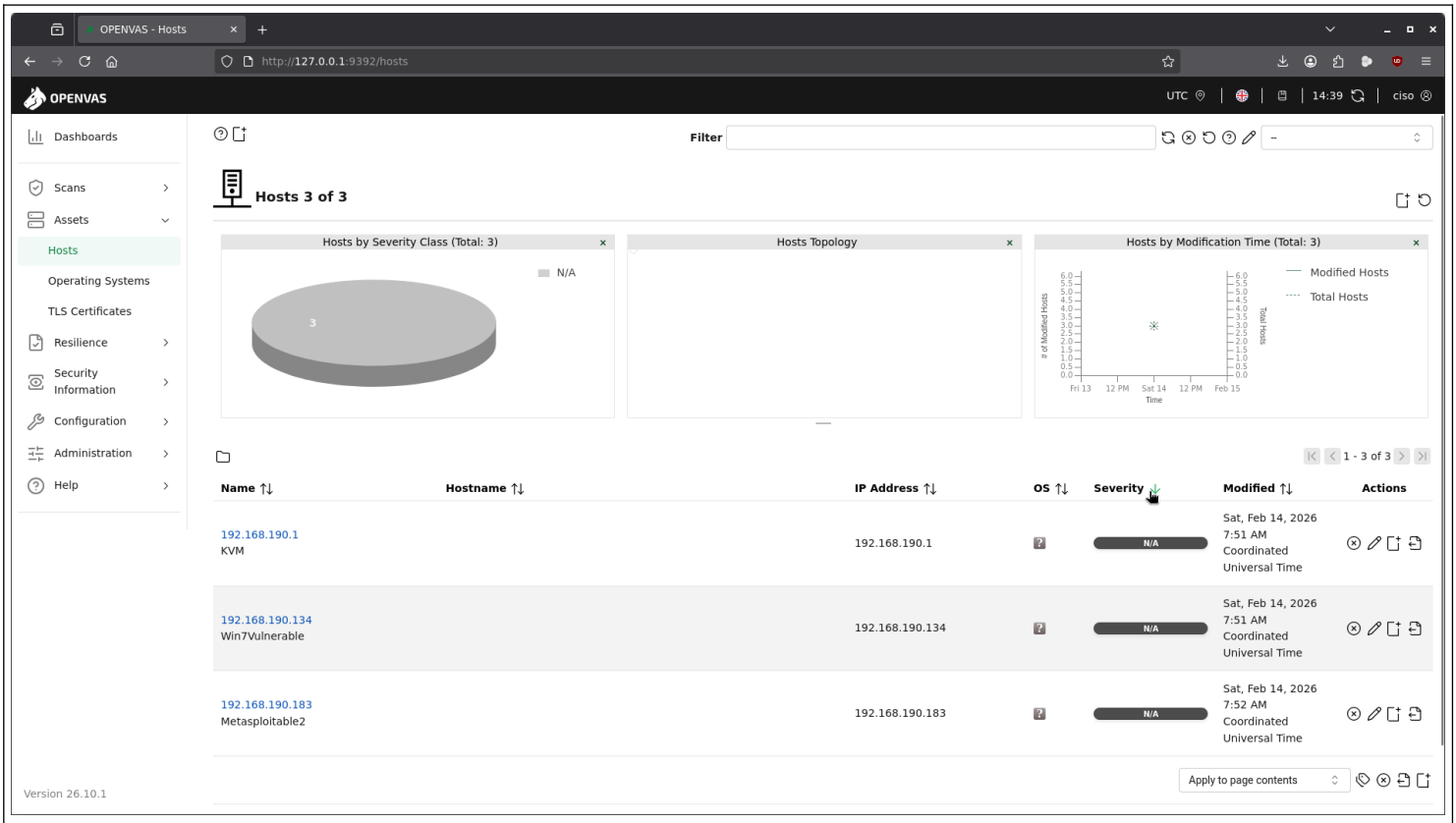


Analyse du Tableau de Bord : Sur la vue globale (*Dashboards*), l'indicateur critique se situe dans le quadrant inférieur droit : **"NVTs by Severity Class"**.

- **Observation :** Le compteur affiche un total de **172 136** NVT (*Network Vulnerability Tests*).
- **Interprétation Technique :** Ce chiffre confirme que la synchronisation des flux (*Feed Sync*) est terminée. La base de données contient plus de 172 000 scripts de détection, couvrant les niveaux de sévérité Log, Low, Medium, High et Critical.
- **Action requise :** Si ce compteur est à zéro ou faible (< 50 000), toute tentative de scan est inutile (faux négatifs garantis). Ici, le système est **opérationnel**.

Inventaire des Actifs (Reconnaissance)

Cette vue recense les machines détectées ou déclarées manuellement dans le périmètre du laboratoire.



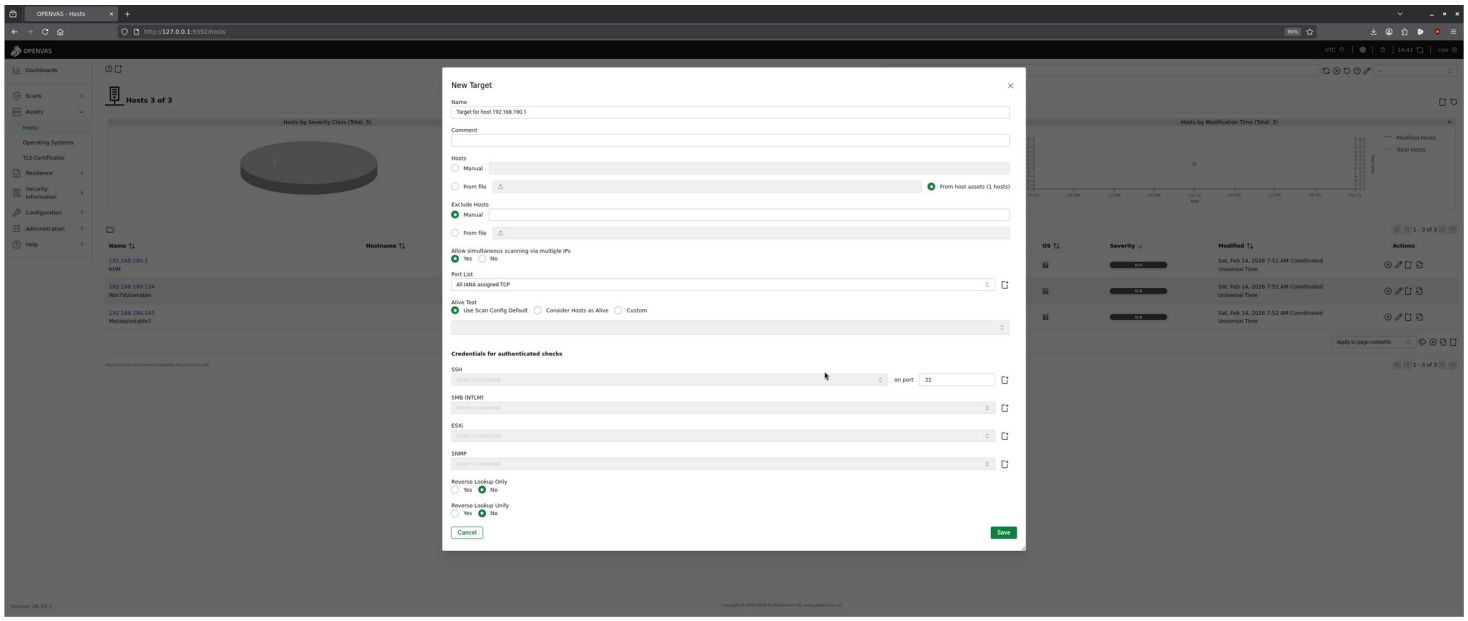
Analyse de l'Inventaire : On identifie trois entrées distinctes :

1. **localhost (192.168.190.1) :** L'hyperviseur lui-même (dans notre cas de lab virtualisé).
2. **metasploitable2 (192.168.190.183) :** Cible Linux volontairement vulnérable.
3. **Win7Vulnerability (192.168.190.134) :** Cible Windows Legacy.

Note : Dans OpenVAS, un "Asset" (Actif) est une simple adresse IP ou un nom d'hôte. Ce n'est pas encore une cible de scan. Pour scanner ces machines, il faut les transformer en objets "Target".

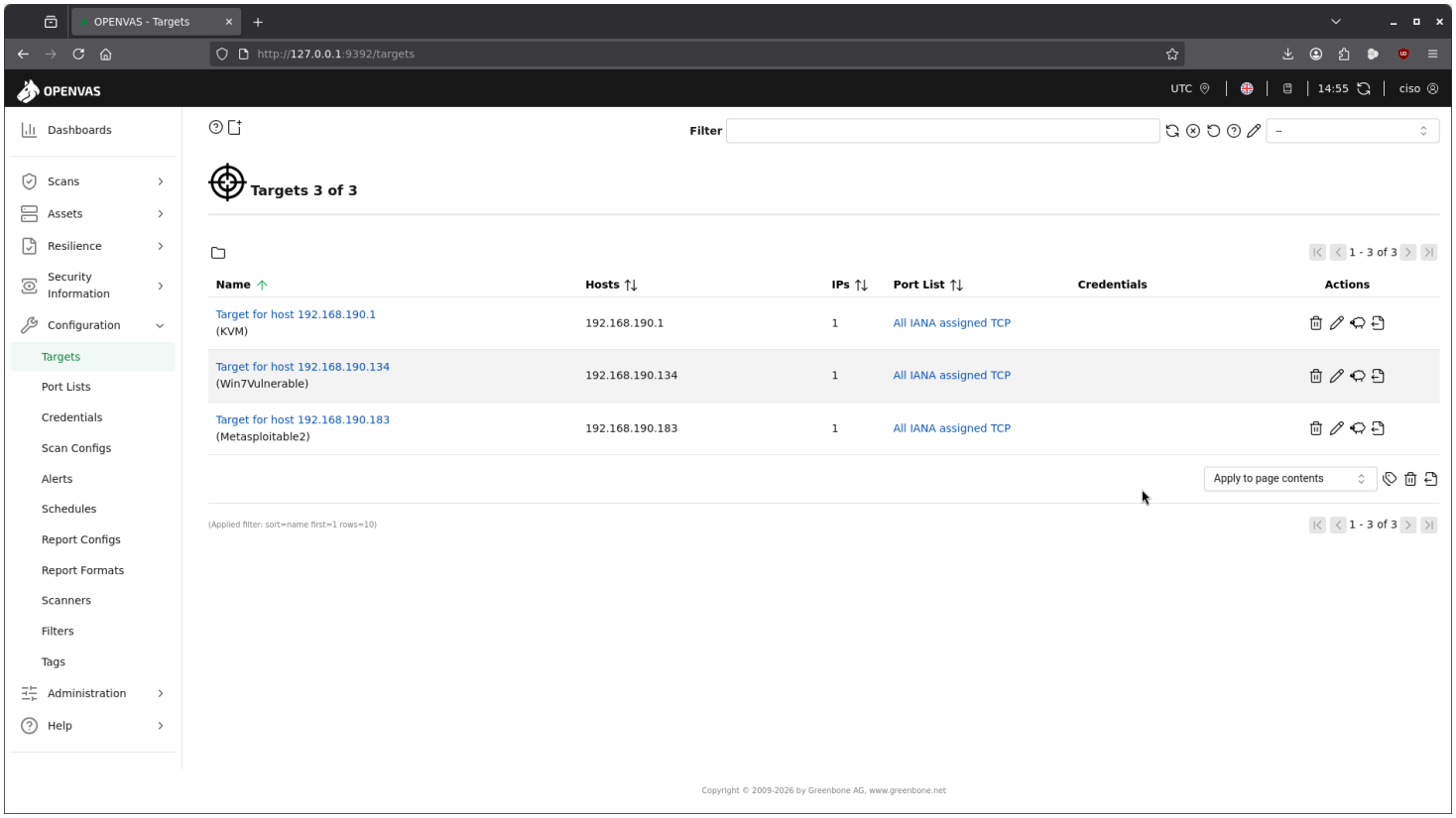
Définition de la Cible (Targeting)

Cette étape consiste à transformer les adresses IP brutes (Hosts) en un objet "Cible" scannable par le moteur OpenVAS.



Accès et Création :

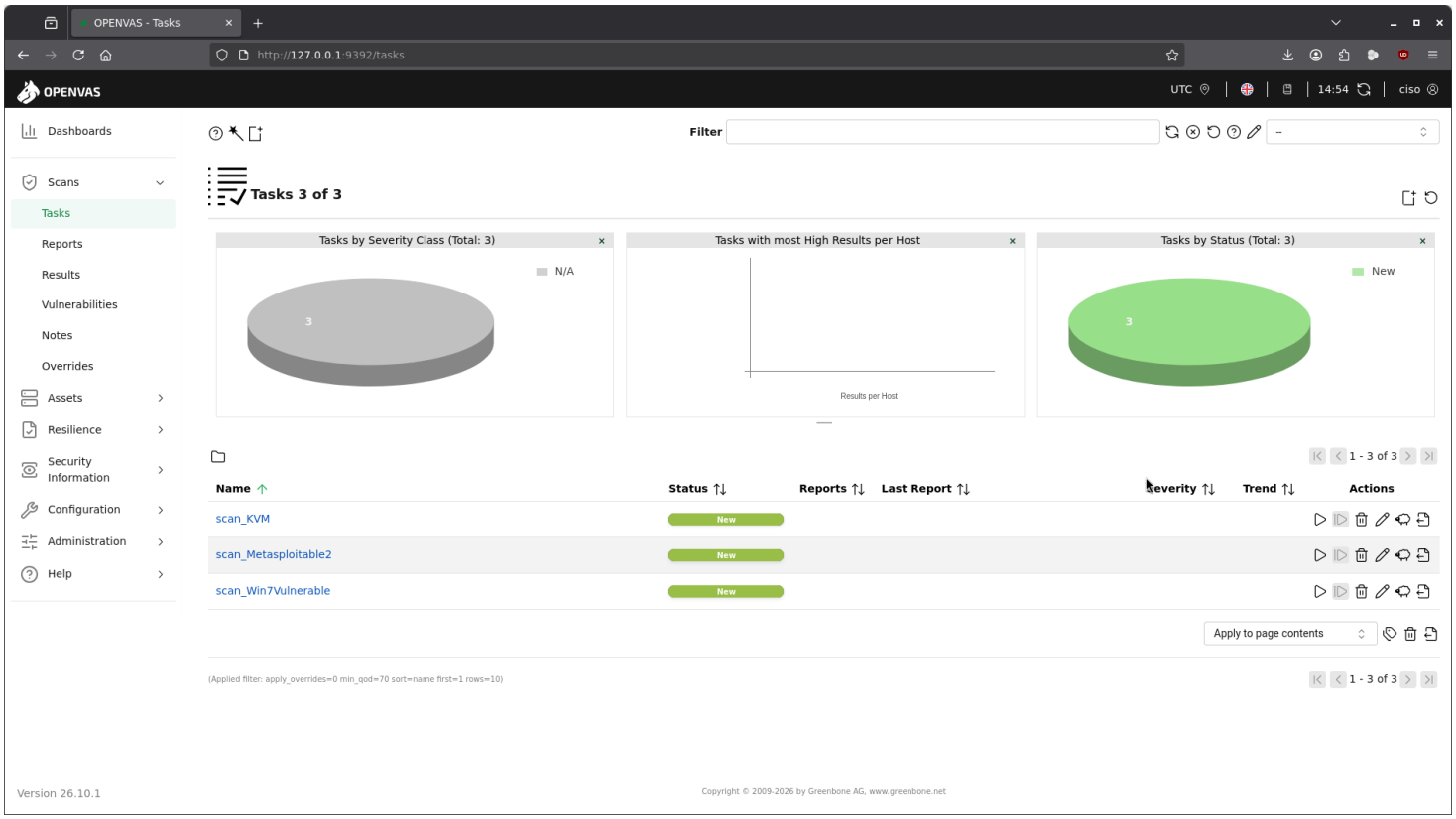
- On navigue dans le menu principal via **Configuration > Targets,** ou directement via **Assets > Hosts.**
- Pour déclarer une nouvelle cible, on clique sur l'icône **"New Target"** située dans la barre d'outils en haut à gauche de la liste (l'icône représente une **feuille blanche avec une petite étoile/plus**).



- **Name :** On donne un nom explicite pour distinguer ce scan des autres futurs audits. Éviter les noms génériques comme "Target 1".
- **Hosts :** On saisit manuellement les IP des machines victimes. *(Note : On sépare les IP par une virgule, sans espace)*. On ne sélectionne pas "localhost" qui est le scanner lui-même.
- **Port (ListAll IANA assigned TCP)** : Par défaut, OpenVAS ne scanne qu'une petite liste. On choisit cette liste (environ 5000 ports) pour couvrir tous les services standards (Web, Mail, SSH, SMB, RDP) sans perdre de temps à scanner les 65 535 ports, ce qui prendrait des heures.
- **Alive Test** : Par défaut, OpenVAS "ping" la machine. Si le Pare-feu Windows bloque le ping (ICMP), OpenVAS croit que la machine est éteinte et ne scanne rien (0 résultat).
    - En mettant "Consider Alive", on force OpenVAS à scanner les ports même si le ping échoue.

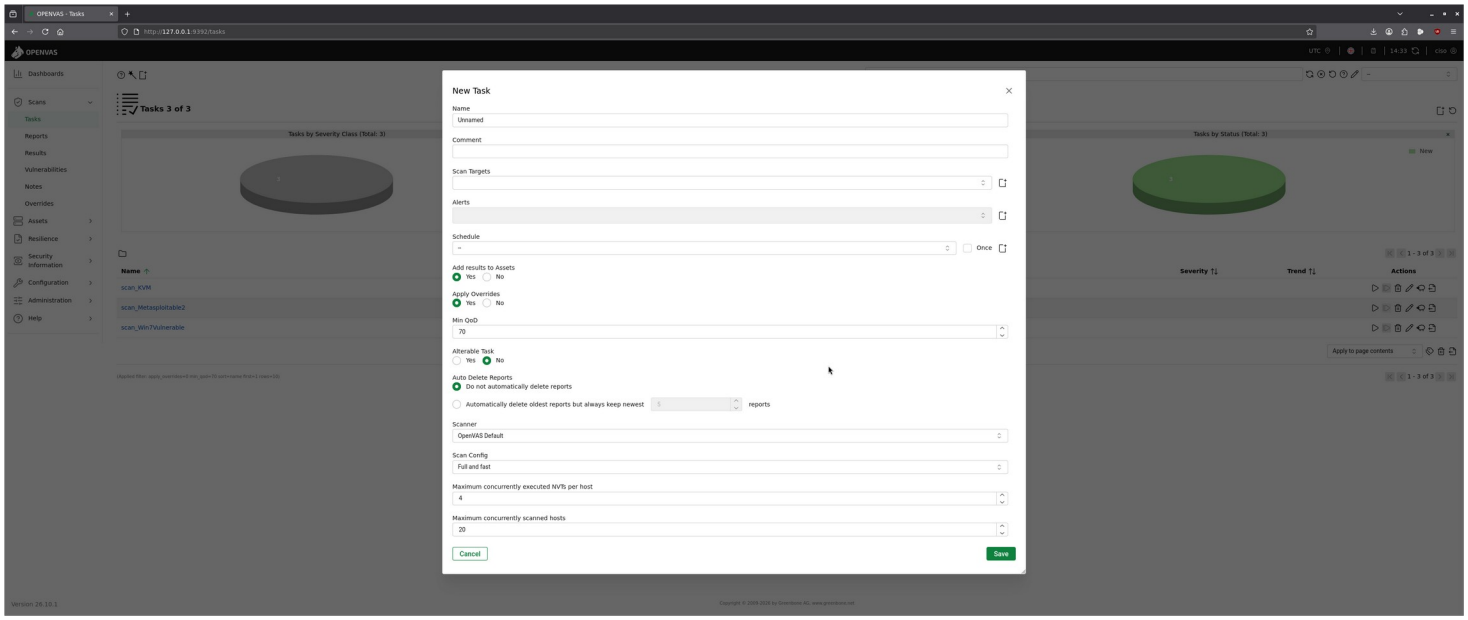
Création et Lancement de la Tâche d'Audit (Tasking)

Une fois la cible définie, elle reste inactive tant qu'elle n'est pas associée à une tâche d'exécution. La **Tâche (Task)** est l'objet qui orchestre l'audit : elle combine la cible, le moteur de scan et la configuration de recherche.



Accès et Initialisation :

- On se rend dans le menu **Scans > Tasks**.
- **Action :** On clique sur l'icône **"New Task"** (située dans la barre d'outils en haut à gauche, représentée par une page avec une étoile/plus).



- **Name** : On utilise une nomenclature standardisée (pas comme moi ;) : [Type d'Audit] - [Cible]. Cela facilite le tri historique dans les rapports futurs.
- **Scan Targets** : On sélectionne dans le menu déroulant la cible créée à l'étape précédente. *Ne pas laisser la valeur par défaut*. C'est ici que le lien se fait.
- **Scan Config (Full and Fast)** : Ce profil est le meilleur compromis :
    - 1. Il détecte les ports ouverts.
    - 2. Il identifie les services (Banner Grabbing).
    - 3. Il ne lance que les scripts (NVT) pertinents pour les services trouvés (ex: il ne testera pas des failles Apache si le port 80 est fermé), optimisant ainsi la vitesse et la charge réseau.
- **Scanner** : On s'assure d'utiliser le moteur ospd-openvas interne (ou OpenVAS Default Scanner selon la version). C'est le moteur qui possède les droits NET_RAW pour forger les paquets.
- **Order for target hosts :** Laisse l'option par défaut. Scanner les hôtes un par un évite de surcharger la table d'état du pare-feu (State Table) ou du routeur intermédiaire.

EXÉCUTION DE L'AUDIT & SURVEILLANCE

> **Note :** C'est le moment où, en appuyant sur 'Play', si un IDS est actif sur le réseau, il va s'allumer comme un sapin de Noël !

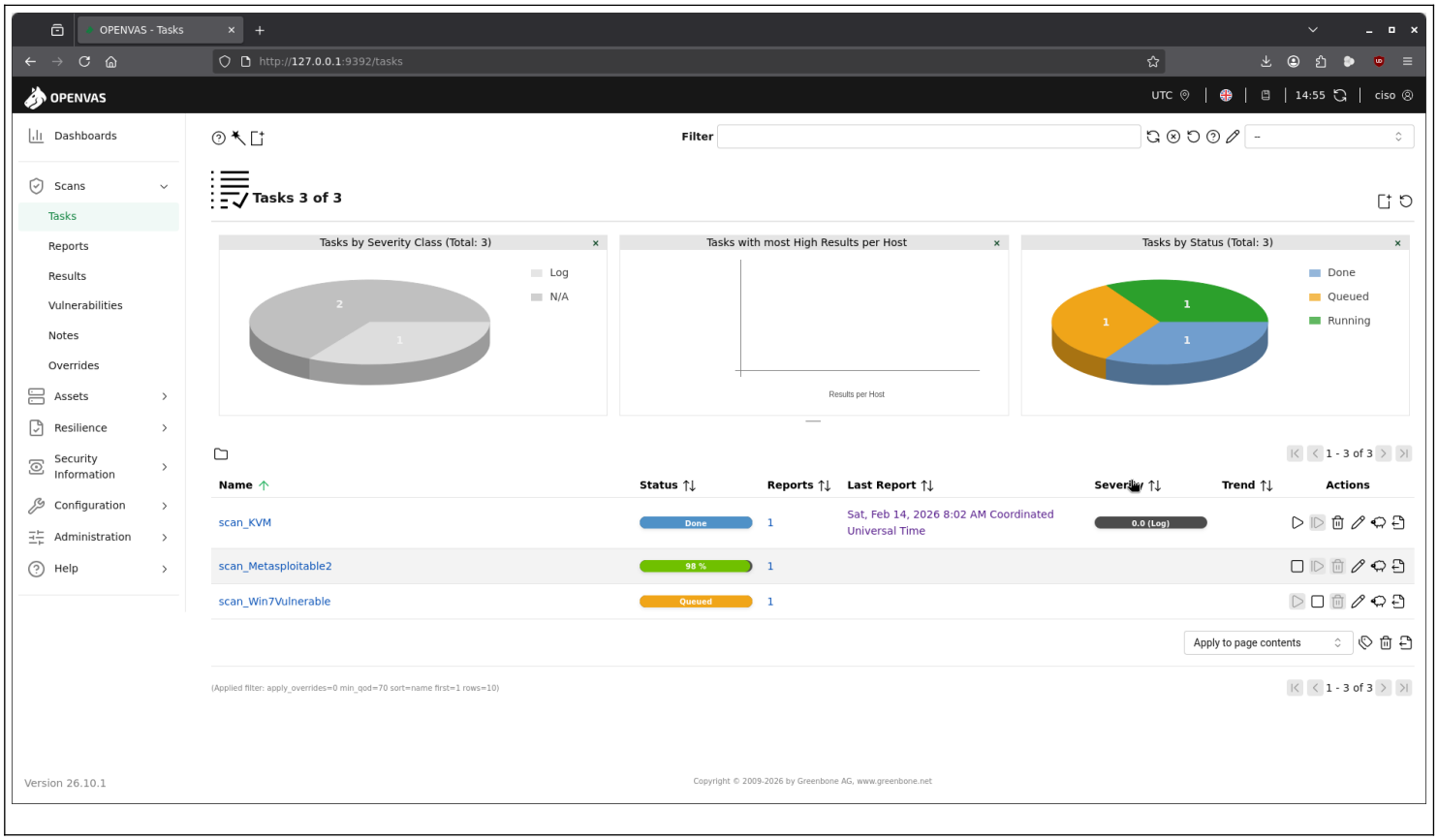
Jusqu'à présent, OpenVAS était un outil passif. En lançant la tâche, il devient un acteur bruyant et agressif sur le réseau.

L'exécution d'un audit en mode **"Full and Fast"** génère un trafic réseau conséquent et atypique. Le scanner ne se contente pas d'établir des connexions légitimes ; il envoie des paquets malformés, teste des dépassements de tampon (buffer overflows) et tente des authentifications par force brute.

Dès la validation de l'ordre de mission, la chaîne de commandement s'active :

1. Le **Manager (gvmd)** verrouille la cible et transmet les instructions au moteur de scan.
2. Le **Scanner (ospd-openvas)** charge en mémoire les milliers de scripts NVT (Network Vulnerability Tests) nécessaires.
3. Les séquences d'attaque sont lancées séquentiellement contre les hôtes définis (192.168.190.183 et 192.168.190.134).

> **Avertissement de Conformité :** Il est impératif de rappeler que ces actions, bien que simulées dans un cadre de laboratoire, peuvent déclencher des alertes de sécurité (SOC/SIEM) ou saturer des équipements réseaux fragiles (imprimantes, vieux routeurs IoT). On opère ici en environnement contrôlé ("White Box").



ANALYSE DES RÉSULTATS & REPORTING

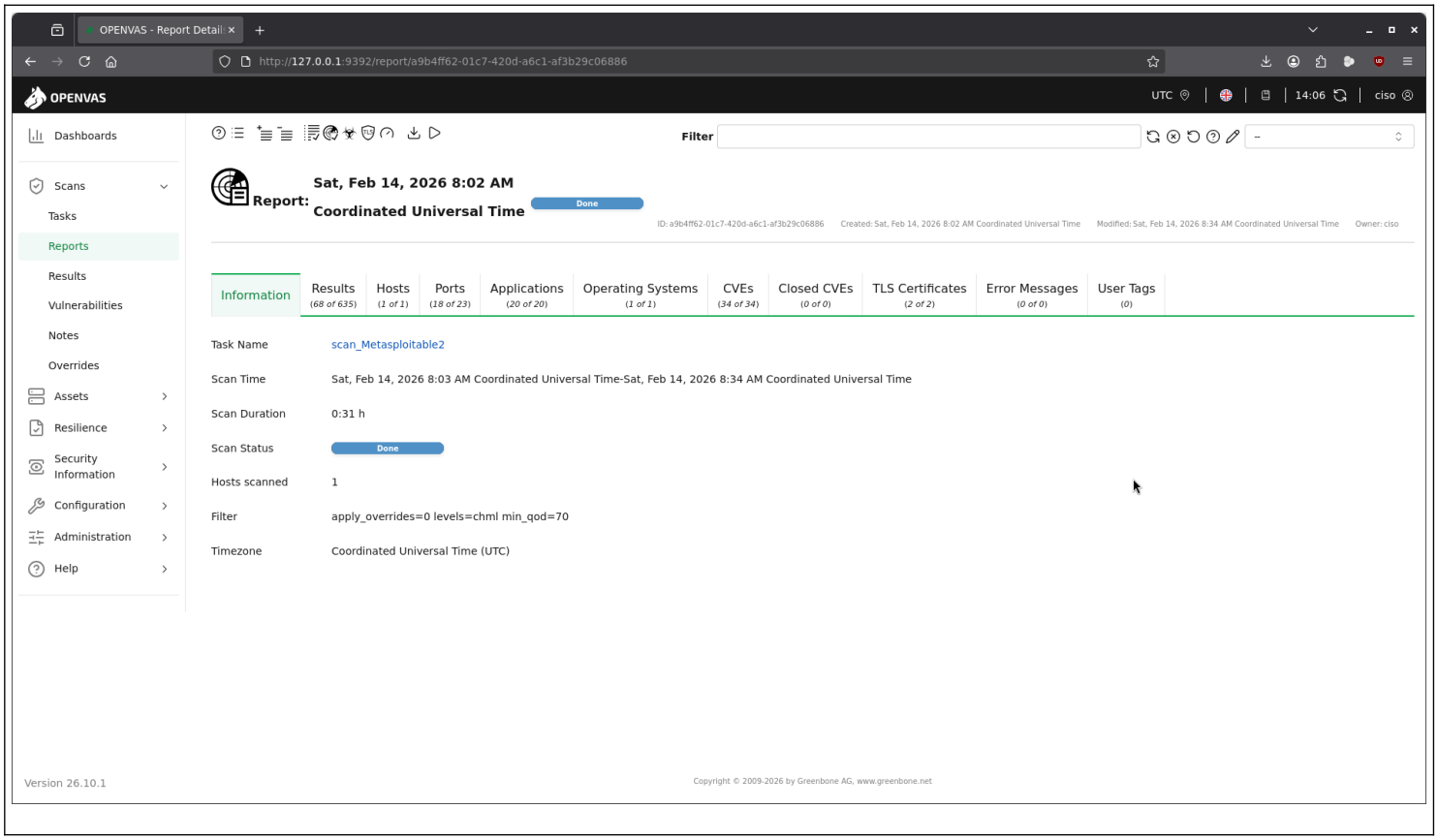
Une fois la barre de progression arrivée à 100% et le statut passé à **"Done"**, l'outil bascule de son rôle offensif (Scanner) à son rôle analytique (Reporting). C'est ici que la donnée brute se transforme en intelligence exploitable.

Accès au Rapport d'Audit

L'accès aux détails techniques ne se fait pas directement depuis la liste des tâches, mais via l'objet "Rapport" généré par celle-ci.

Procédure :

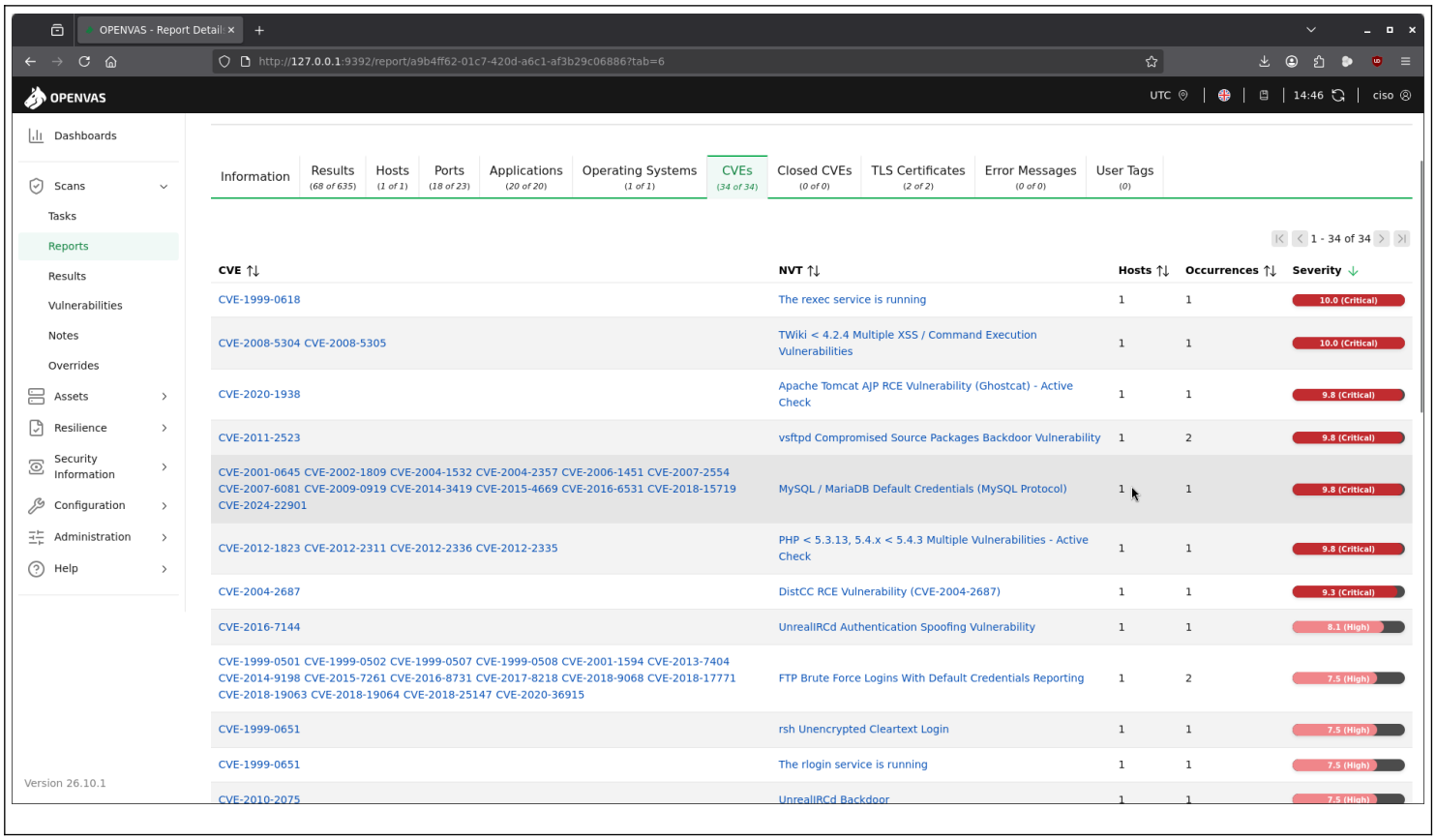
1. Depuis le menu **Scans > Tasks**, on identifie la tâche terminée.
2. Dans la colonne **"Last Report"**, on clique directement sur la date/heure (lien cliquable).
3. On est redirigé vers la vue **"Results"** du rapport spécifique.



Note : Il est possible d'avoir plusieurs rapports pour une même tâche (historique des scans). Toujours sélectionner le plus récent pour l'analyse courante.

Interprétation des Métriques (Le Tri)

Le tableau de bord des résultats présente une liste de vulnérabilités triées par sévérité. La compréhension des scores est impérative pour prioriser les actions correctives.



La Matrice de Gravité (CVSS)

OpenVAS utilise le standard **CVSS (Common Vulnerability Scoring System)** pour noter chaque faille de 0.0 à 10.0.

Niveau	Score CVSS	Code Couleur	Action Requise
Critical	9.0 - 10.0	Rouge Foncé	**Immédiate.** Risque de compromission totale (RCE, Root).
High	7.0 - 8.9	Rouge	**Urgent.** Exploitation possible, impact fort.
Medium	4.0 - 6.9	Orange	**Planifiée.** Nécessite souvent des conditions spécifiques.
Low	0.1 - 3.9	Bleu	**Veille.** Fuite d'information mineure.
Log	0.0	Gris	**Info.** Inventaire purement informatif (OS détecté, Ports ouverts).

Le Facteur de Confiance (QoD - Quality of Detection)

Chaque résultat possède un score **QoD** (barre colorée dans la liste).

- **70% - 100% :** La faille est confirmée par une réponse technique précise (Version exacte, retour de script).
- **< 70% :** La faille est "suspectée" (basée uniquement sur une bannière ou une version probable). Risque de **Faux Positif**.

Anatomie d'une Vulnérabilité

Pour comprendre comment corriger une faille, on clique sur son nom (ex: *Microsoft Windows SMB Server Multiple Vulnerabilities*). La fiche détaillée fournit trois informations critiques :

1. **Summary (Résumé) :** Explique la nature de la faille technique.
2. **Detection Result (Preuve) :** Affiche la réponse brute du serveur qui a déclenché l'alerte (ex: "Received banner: Samba 3.0.20"). C'est la preuve irréfutable pour l'administrateur système.
3. **Solution (Remédiation) :**
    - **Workaround :** Contournement temporaire (ex: Désactiver SMBv1).
    - **Vendor Fix :** Lien vers le patch officiel (KB Microsoft, mise à jour apt/yum).

Export & Livrable (Le PDF)

Pour transmettre les résultats à la hiérarchie ou aux équipes d'exploitation, le rapport doit être exporté hors de l'interface Web.

Procédure d'Export :

1. Dans la vue du rapport, on clique sur l'icône de **Téléchargement** (flèche vers le bas) située en haut à gauche de la barre d'outils.
2. Une fenêtre modale demande le format. On sélectionne :
    - **PDF :** Pour un rapport managérial ou d'archivage (lisible par un humain).
    - **CSV / XML :** Pour un traitement automatisé ou une intégration dans un Excel de suivi.
3. On valide par **OK**. Le fichier est généré et téléchargé localement.