Capev2

Objectif :

Mise en place d’une sandbox pour l’analyse de malwares.

Vocabulaire :

Sandbox : Mécanisme de sécurité qui exécute un programme dans un environnement isolé du système hôte afin de contenir et d’observer son comportement sans risque.

Hôte : machine physique principale sur laquelle tourne l’hyperviseur. Elle fournit les ressources matérielles.

Guest : machine virtuelle créée et exécutée sur l’hôte. Elle possède son propre système d’exploitation, totalement isolé, mais dépendant des ressources de l’hôte.

C’est quoi Cape ?

CAPE Sandbox est une solution open-source d’analyse automatique de malwares, dérivée de Cuckoo Sandbox.

Elle permet d’exécuter des fichiers suspects dans un environnement virtuel isolé pour instrumenter leur comportement dynamique (création de processus, modifications systèmes, trafic réseau, dumps mémoire, captures d’écran, etc.).

Parmi ses ajouts par rapport à Cuckoo : dépackage automatique, extraction de configurations, classification via signatures YARA, prise en charge des contre-mesures anti-sandbox.
CAPE est la version modernisée et enrichie de Cuckoo, conçue pour répondre aux besoins actuels des équipes SOC et des analystes en cybersécurité.

Architecture :

CAPE repose sur une architecture centralisée et modulaire, composée de deux éléments principaux :

- L'hôte :

  C'est la machine principale (souvent sous Linux) qui gère l'ensemble du processus d'analyse.

  Elle pilote les machines virtuelles, lance les analyses, capture le trafic réseau et génère les rapports finaux.

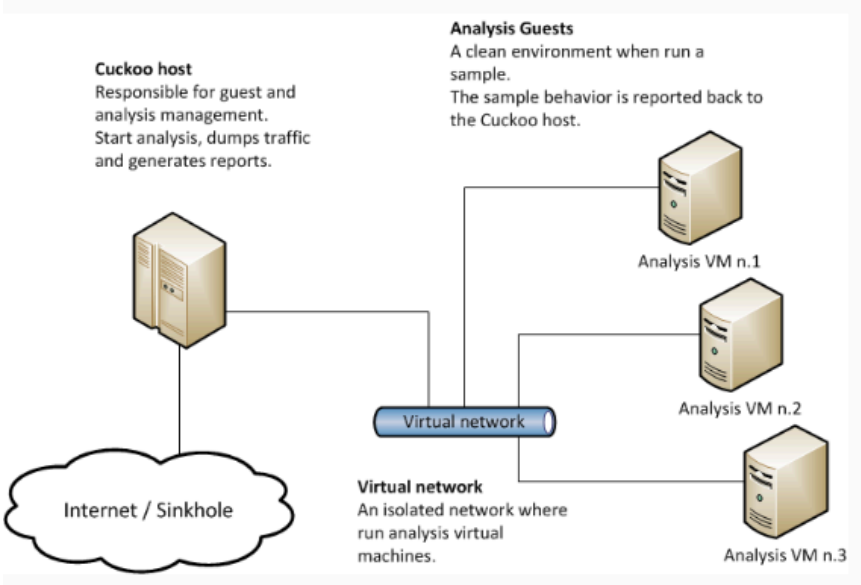
- Le/les Guest :

  Ce sont les machines virtuelles isolées dans lesquelles les fichiers suspects sont exécutés.

  Chaque analyse se fait dans un environnement propre, réinitialisé à chaque fois, garantissant qu'aucune trace d'un échantillon précédent ne perturbe la suivante.

Les machines invitées communiquent avec l'hôte via un réseau virtuel isolé.

Ce réseau peut être relié à Internet ou à un sinkhole (faux Internet) pour observer le comportement réseau sans compromettre la sécurité du système.



Configuration :

L’installation :

Pour fonctionner Cape a besoin :

- 1 hyperviseur (la doc conseil KVM)
- Virtual Manager (permet de créer les Guest (VM))
- Cape (le programme principale)
- Poetry (gestionnaire d’environnement Python)

KVM (Kernel-based Virtual Machine) a été choisi comme hyperviseur car il est nativement intégré à Linux, performant, et open source.

Il permet de créer et gérer des machines virtuelles directement via le noyau, ce qui garantit une meilleure stabilité et une intégration fluide avec CAPE.

Poetry sert à installer CAPE proprement et à gérer ses dépendances Python sans conflit.

Il crée un environnement isolé automatiquement et garde la même configuration sur toutes les machines.

Une fois cape installé tout se passe dans le repertoire `/opt/CAPEv2`.

La 1ere chose a faire est de modifier les fichier de configuration présent dans `/opt/CAPEv2/conf`

Les plus important sont :

- **cuckoo.conf** → paramètres généraux de CAPE (catégories d’analyse, suppression des fichiers, etc.)
- **routing.conf** → gestion du réseau et du proxy utilisé pendant les analyses
- **kvm.conf** → définition des machines virtuelles utilisées pour l’analyse : nom, IP, snapshot, système d’exploitation

Toutes les configuration sont documenté, ainsi on sait a quoi sert chaque paramètre.

cuckoo.conf :

Voici les parametre a changé en priorité :

- `machinery` → mettre son hyperviseur, ici kvm `machinery = kvm`
- `ip` → renseigner l'adresse passerelle de la machine guest (l'IP par laquelle CAPE communique avec la VM d'analyse). ex : `ip = 192.168.55.1`
- `port` → port d'écoute du serveur CAPE (par défaut **2042**). ex : `port = 2042`

routing.conf :

Les seuls valeurs a changé ici sont :

- `internet` → définir l'interface réseaux par lequel la machine guest va communiquer, il y a 2 valeur possible :
  - `none` : empêche toute sortie Internet (analyse totalement isolée).
  - `direct` : autorise la VM à accéder directement à Internet.(à éviter sans réseau contrôlé)

Il est possible aussi de rediriger le trafic via Tor, un VPN, ou **INetSim** pour simuler un faux Internet.

- `route` → soit `none` (machine totalement isolé d'internet) ou `internet` (redirige sur l'interface choisi)

kvm.conf :

Ce fichier permet à CAPE de **définir et gérer les machines virtuelles** utilisées pour les analyses.

Chaque section correspond à un guest et contient ses paramètres réseau, système et techniques.

Voici les paramètres a changé :

- `machines` → nom(s) des VM utilisées pour l'analyse.
- `interface` → interface réseau virtuelle utilisée

configuration d'une machine :

- `label` → nom de la VM dans virt-manager
- `platform` → système d'exploitation du guest
- `ip` → adresse IP de la machine guest
- `arch` → architecture du système (ex : `x64`).
- `snapshot` → nom du snapshot propre créé dans virt-manager
- `uuid` → identifiant unique de la VM (visible via `virsh list --all`)
- `resultserver_ip` → IP de la machine hôte
- `resultserver_port` → port du serveur CAPE

Exemple de configuration :

```
[kvm]
# Specify a comma-separated list of available machines to be used. For each
# specified ID you have to define a dedicated section containing the details
# on the respective machine. (E.g. cuckoo1,cuckoo2,cuckoo3)
machines = w10cape

interface = virbr1
# To connect to local or remote host
dsn = qemu:///system

# To allow copy & paste. For details see example below
[w10cape]
label = w10cape
platform = windows
ip = 192.168.55.36
arch = x64
tags = win10
snapshot = snapshot1
uuid = 2d7ce095-cf7b-41e1-8c78-52e003166a4a
resultserver_ip = 192.168.55.1
interface = virbr1
resultserver_port = 2042
```

Une fois la configuration de CAPE terminée, il faut préparer la machine virtuelle (guest) qui servira d'environnement d'analyse.

Préparation de la VM Guest :

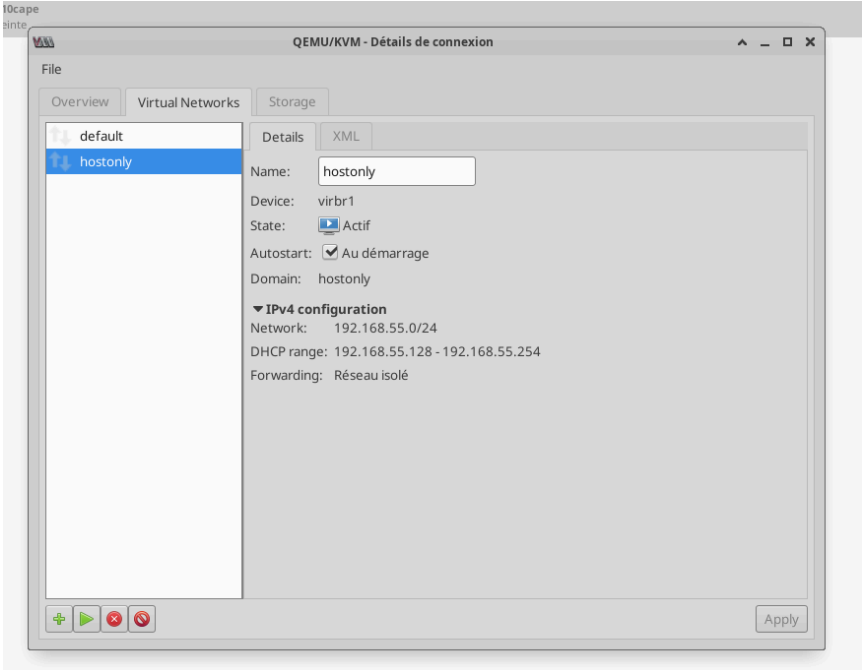
Après avoir créé la VM dans **virt-manager**, installez les composants suivants dans la VM :

- **Agent CAPE** (doit tourner en permanence pour permettre les analyses).
- **Python 3.10.6 (32-bits)** nécessaire pour l'agent
- Pour une vm complètement vulnérable on peux enlevé le pare feu et l'antivirus

Optionnel (pour plus de réalisme)

- Logiciels usuels selon le profil (navigateur, MS Office, runtimes Visual C++).
- Quelques fichiers utilisateurs (documents, images) et historique navigateur pour rendre l'environnement réaliste.

Pour éviter tout risque, il est essentiel de créer un réseau virtuel isolé dans virt-manager (cocher Réseau isolé).
Il est conseillé d'éviter le sous-réseau `192.168.122.0/24`, souvent détecté par les malwares.



Une fois cela fais on relance la VM et l'on fait les actions suivantes :

- configurer une **IP fixe** sur le réseau virtuel,
- vérifier que l'host et le Guest se ping
- vérifier que l'agent tourne bien (Host : `nc -zv 192.168.55.1 2042`)
- désactiver les mises à jour automatiques,
- créer un **snapshot propre** (`snapshot_clean`) une fois l'agent en état *ready*.

Une fois le snapshot propre créé, la VM est prête à être utilisée par CAPE pour exécuter des échantillons en toute sécurité.

Chaque analyse restaurera automatiquement ce snapshot afin de garantir un environnement vierge à chaque test.

A partir d'ici on passera en utilisateur cape (créer automatiquement après l'installation de cape.sh)

Creer un mot de passe pour l'user cape :

```
sudo passwd cape
```

Une fois cela fait on passe en user cape :

```
su - cape
```

Lancement du serveur CAPE (interface web Django)

Une fois la configuration terminée et la VM prête, on peut lancer le serveur CAPE.

Ce serveur utilise **Django**, un Framework web Python intégré à CAPE, pour afficher l'interface d'administration et d'analyse.

On se place dans le bon répertoire : `/opt/CAPEv2/web`

Ensuite, on lance le serveur avec **Poetry** :

```
poetry run python3 manage.py runserver_plus --traceback --keep-meta-shutdown
```

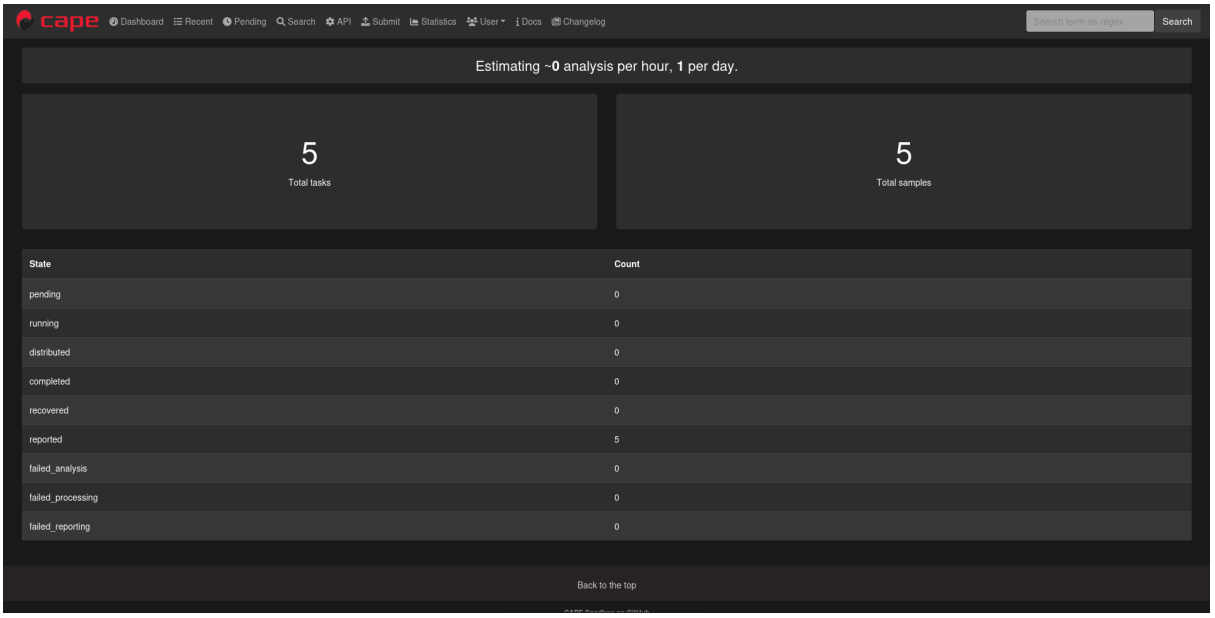
> Par défaut, le serveur démarre sur localhost:8000.
>
> Pour le rendre accessible depuis n'importe quelle machine du réseau, on utilise :
>
> ```
> poetry run python3 manage.py runserver_plus 0.0.0.0:8000 --traceback --keep-meta-shutdown
> ```

Le serveur web CAPE permet d'accéder à l'interface d'administration depuis un navigateur, ex :

http://192.168.203.60:8000

Présentation de l’interface Web :

Une fois connecté on arrive sur la page d’accueil de l’interface web de Cape et on y voit un Dashboard :



On y retrouve :

- le nombre total de tâches et d’échantillons,
- l’état des analyses (en attente, en cours, terminées, échouées),
- des statistiques sur le nombre d’analyses par jour et par heure.

....

Dans les différent menu présent en haut de la page les plus important sont :

- **Recent** → Permet de consulter les dernières analyses effectuées. Chaque ligne correspond à une tâche avec son état
- **Submit** → Sert à soumettre un nouveau fichier ou une URL pour analyse. On peut choisir la machine, la durée d’exécution, et le mode réseau. C’est le

point de départ de toutes les analyses manuelles.

Les autres sections moins importante servent à :

- **Pending** →Montre les tâches en attente d'exécution (analyses programmées mais non encore lancées).
- **Search** →Permet de rechercher un échantillon à partir de son nom, de son hash (MD5, SHA1, SHA256) ou d'un identifiant d'analyse.
- **API** → Donne accès à l'API REST de CAPE, utilisée pour automatiser les soumissions ou récupérer les rapports via des scripts externes.
- **Statistics** → Regroupe des données globales sur les analyses réalisées : volume, durée moyenne, plateformes utilisées, etc.
- **User** → Menu de gestion de compte : informations utilisateur, déconnexion et options d'administration.
- **Docs** → Lien vers la documentation officielle de CAPEv2 (ReadTheDocs).
- **Changelog** → Historique des mises à jour et évolutions de la plateforme.

Première Analyse :

Une fois la 1ere analyse faites on a accès a un analyse complète de toutes les actions que le processus a fait, on l'a retrouve dans la section "Recent", a l'intérieur on retrouve énormément d'information.

Quick Overview :

Dans cet onglet on retrouve 8 sections :

- **Analysis** → informations générales de l'analyse : la catégorie du fichier (ici un exécutable Windows), la date et l'heure de début et de fin, ainsi que la durée totale d'exécution.
- **Machine** → informations sur la machine virtuelle utilisée pour l'analyse. Elle indique le nom de la VM , son type d'hyperviseur (ici KVM), ainsi que les horodatages de démarrage et d'arrêt.

- **File Details** → **détails du fichier analysé** : son nom, son type, ainsi que sa taille. CAPE calcule automatiquement plusieurs **empreintes cryptographiques** (MD5, SHA1, SHA256, etc.) permettant d'identifier l'échantillon sur des bases de données publiques comme VirusTotal ou MalwareBazaar.
- **Signatures** → Dans cette section, on retrouve les signatures comportementales détectées durant l'exécution du fichier. Les signatures en bleu sont principalement informatives et décrivent des actions communes à de nombreux programmes. Certaines signatures peuvent aussi apparaître en rouge, et celles-ci sont les plus critiques. Elles signalent généralement un comportement clairement malveillant ou une technique d'attaque confirmée
- **Screenshots →** captures d'écran prises automatiquement pendant l'exécution du fichier dans la machine virtuelle. Elles permettent d'observer visuellement le comportement du programme : ouverture de fenêtres, messages d'erreur, installations, ou exécution de commandes. Ces captures sont particulièrement utiles pour confirmer une action suspecte détectée dans les logs, ou pour visualiser un comportement qui ne génère pas de trace réseau ou système claire.
- Host et DNS → liste les adresses IP ou noms d'hôtes contactés et montre les résolutions de noms effectuées.
- Summary → **résume les principales interactions système** effectuées par le programme durant son exécution
    - **Accessed Files** → fichiers consultés ou ouverts.
    - **Modified Files** → fichiers modifiés.
    - **Deleted Files** → fichiers supprimés.
    - **Registry Keys / Read Registry Keys** → clés du registre Windows créées, modifiées ou simplement lues.
    - **Mutexes** → objets de synchronisation utilisés pour empêcher plusieurs instances du programme de s'exécuter simultanément (souvent utilisés par les malwares pour s'assurer qu'ils ne se lancent qu'une fois).

Behavioral Analysis :

- Process Tree : CAPE affiche l'arborescence des processus créés ou manipulés pendant l'exécution du fichier.

Pour chaque nœud du *process tree*, CAPE propose une vue détaillée (chemin complet, ligne de commande, PID et PID parent) suivie d'un journal chronologique des appels système (NtOpenFile, RegSetValue, CreateProcess, socket, etc.) avec leurs arguments et codes de retour. Des filtres permettent d'isoler rapidement les catégories pertinentes registry, filesystem, network, process, crypto ce qui facilite la recherche d'actions suspectes.

Chaque filtre a un intérêt spécifique pour l'analyste :

- **registry** : permet d'identifier des tentatives de persistance (ajout de clés Run, modification du registre système, stockage de configurations malveillantes).
- **filesystem** : révèle les créations, lectures et suppressions de fichiers — souvent utilisées pour déposer un dropper, un script ou effacer des traces.
- **network** : met en évidence les connexions sortantes (C2, exfiltration de données, téléchargement d'un payload secondaire).
- **process** : utile pour repérer les créations de processus suspects ou les injections dans d'autres applications légitimes.
- **threading / synchronization** : indique un comportement avancé de gestion de threads, typique de malwares multi-étapes ou de techniques d'injection.
- **crypto** : permet de voir l'utilisation d'API cryptographiques (souvent associée à du chiffrement de données, ransomware, ou obfuscation).

Network Analysis :

Cette section sert à analyser **tout le trafic réseau généré pendant l'exécution du fichier** dans la sandbox.

Elle permet de voir :

- **Hosts / DNS** → les domaines ou IP contactés ;
- **TCP / UDP / HTTP / SMTP / IRC / ICMP** → les protocoles utilisés (exfiltration, C2, communication interne, ping test, etc.) ;

- et de **télécharger le fichier PCAP** pour l'ouvrir dans *Wireshark* et examiner les paquets en détail.

Dropped Files :

Liste tous les fichiers créés ou extraits par l'échantillon pendant son exécution.

Quand le malware écrit un nouveau fichier sur le disque (ex. un script `.ps1`, un exécutable, un `.dll`, un `.tmp`...), CAPE l'enregistre ici.

On y vois :

- le nom du fichier,
- son type (texte, PE, data, etc.),
- sa taille,
- ses empreintes hash (MD5, SHA1, SHA256...),
- et même des liens directs VirusTotal / MalwareBazaar s'ils existent.

Payloads :

Rassemble ce que CAPE a **extrait ou observé en mémoire** pendant l'exécution — c'est souvent le cœur malveillant qui n'apparaît pas dans les fichiers écrits sur disque.

Qu'est-ce qu'on y trouve (résumé) :

- **Type** (ex : Unpacked Shellcode, Injected DLL, Reflective DLL, Script décompressé).
- **Process** et **PID** qui ont chargé/exécuté la charge (important pour corrélation avec le process tree).
- **Adresse virtuelle** où le code a été injecté ou mappé.
- **Taille + hashes (MD5/SHA1/SHA256/TLSH/Ssdeep)** pour identifier et rechercher le payload.
- **Liens VirusTotal / MWDB / Bazaar** si le payload est déjà référencé.

- Boutons pour télécharger le payload, voir son binaire (BinGraph), ou l’analyser plus loin.

Cette section est très importante, quand un binaire est s’exécute dans la vm et si il déchiffre du code il va donc alloué de la mémoire et écrire dedans, cape surveille de très près cette activité et tente d’extraire le contenue de la zone mémoire exécutée.