

! Warning

It is important that you know what you're doing, this guide is here to help you but it may contain some error. We are not responsible for any damage caused by following this tutorial.

This tutorial is the result of the program Project for Industry & Innovation for master II student by [ESILV](#) and in collaboration with [Risk&Co](#). It was written by Elliot AILLERIE, Alexis BOURDIN, Thomas NGO and Lucas WITVOET. The project team was under the supervision of Juliette DESORMONTS, Cybersecurity consultant at Risk&co and Walter PERETTI, Head of the IT, IOT & Security Department at ESILV.

For this article, we'll use [Ubuntu 20.04](#) and [KVM](#) to virtualize our Windows 7 machine (analysis machine).

Table of Contents

1. [CAPE V2 Installation](#)
 1. [Requirements](#)
 2. [KVM-QEMU](#)
 3. [CAPEv2](#)
 4. [Installation of the guest machine](#)
2. [OpenCTI Installation](#)
3. [MISP Installation](#)
 1. [Docker installation](#)
 2. [Manual Deployment](#)
4. [Connect Instances](#)
 1. [CAPE ↔ OpenCTI](#)
 2. [CAPE ↔ MISP](#)
 3. [MISP ↔ OpenCTI](#)
 4. [OpenCTI ↔ OpenCTI](#)
5. [Sources](#)

CAPE v2

Here is the link to the [Capev2 Github](#) and the [official documentation](#).

Requirements

- First start by updating your repository.

```
$ sudo apt update
```

- Then install python3 and pip for python3 if it's not already done.

```
$ sudo apt install python3  
$ sudo apt install python3-pip
```

- Install Pillow for python with the latest, this version can change . Take a look at [Pillow changelog's](#) to see what's the latest version.

```
$ python3 -m pip install Pillow==9.0.0
```

KVM-QEMU

Once CAPEv2 is installed, we need virtual machine to run the different malware. This machine is also called the Guest Machine. You can also use [Virtualbox](#) but it's recommended to use [KVM](#) as it's less unlikely to be detected as a VM by the malware. Quoting CAPE lead developer :

We strongly NOT recommend to use VirtualBox due to be super easy to detect by malware, use KVM as suggested in readme for amazing performance and anti-*

Like the cape2.sh we'll use doomedraven script and add the permission before executing it.

```
$ wget https://raw.githubusercontent.com/doomedraven/Tools/master/Virtualization/kvm-qemu.sh
```

When executing the scrip DO NOT FORGET to REPLACE USERNAME by your own USERNAME.

```
$ sudo ./kvm-qemu.sh all <username> | tee kvm-qemu.log
```

This installation will take some time, when finished reboot your computer.

```
$ sudo shutdown -r now
```

CAPEv2

We will use the script cape.sh to install it with all the optimization.

```
$ wget https://raw.githubusercontent.com/doomedraven/Tools/master/Sandbox/cape2.sh
```

We have to change the permission in order for the script to be executable :

```
$ sudo chmod a+x cape2.sh
```

We can then execute the script and add tee command to get a log file of the installation. This installation will take a certain time.

```
$ sudo ./cape2.sh base cape | tee cape2-installation.log
```

Then we modify the to allow [MITRE ATT&CK](#) technics and strategy to be prompt.

```
$ sudo chown -R cape:cape /opt/CAPEv2/data/
```

To finalise the installation of volatility, you will have to check if the windows symbols folder is present in volatility and if so, download it.

```
$ ls cd /usr/local/lib/python3.8/dist-packages/volatility3/symbols
```

Otherwise, if it's not present :

```
$ sudo wget https://downloads.volatilityfoundation.org/volatility3/symbols/windows.zip
-O /usr/local/lib/python3.8/dist-packages/volatility3/symbols/windows.zip
$ cd /usr/local/lib/python3.8/dist-packages/volatility3/symbols/
$ sudo unzip windows.zip
$ sudo rm windows.zip
```

Then we modify the rights on the downloaded files :

```
$ sudo chmod 777 windows/*
```

Once the installation finished reboot your machine :

```
$ sudo shutdown -r now
```

Installation of the guest machine

As mentioned in the official documentation : For analysis purposes you are recommended Windows 7 with User Access Control disable even if CAPE supports Windows 10 too.

Windows 7 VM installation

First run virt-manager and create a new virtual machine choosing your Windows 7 iso file. You can follow [this tutorial](#) to install a VM using KVM.

```
$ sudo virt-manager
```

Python3 and Pillow installation

- Install [Python 3.6](#) in the Windows 7 VM. Don't forget to include Python in PATH. Don't install a newer version of Python.
- Install [Pillow latest version](#), the same as on your host computer with the command :

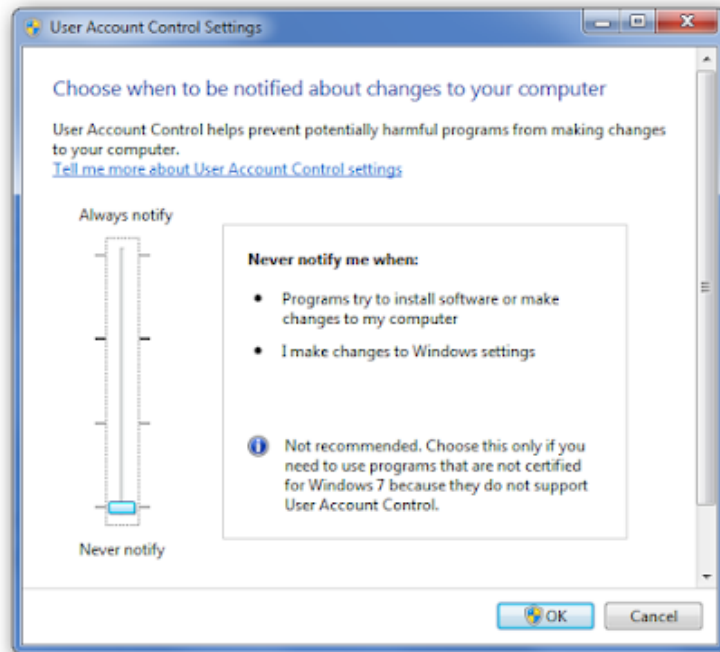
```
pip3 install Pillow==9.0.0
```

Install additional software

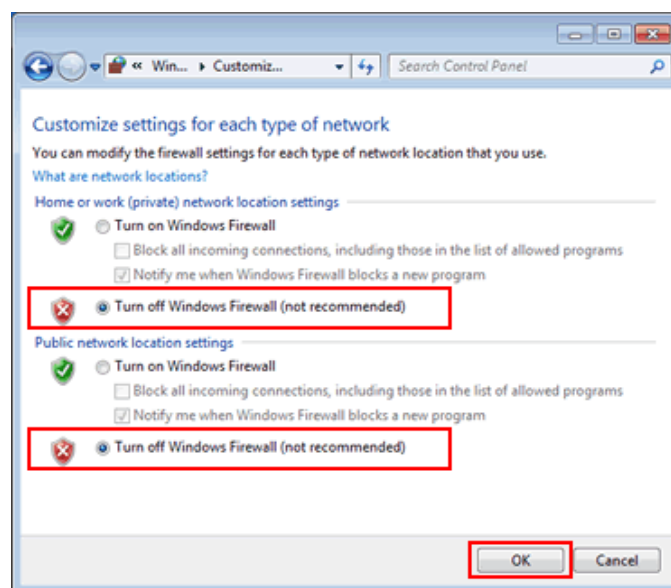
We might want to install additional software such as browsers, PDF readers, office suites, etc for fully functional features. Remember to disable the “auto-update” or “check for updates” feature of any additional software.

Disable Windows Protections

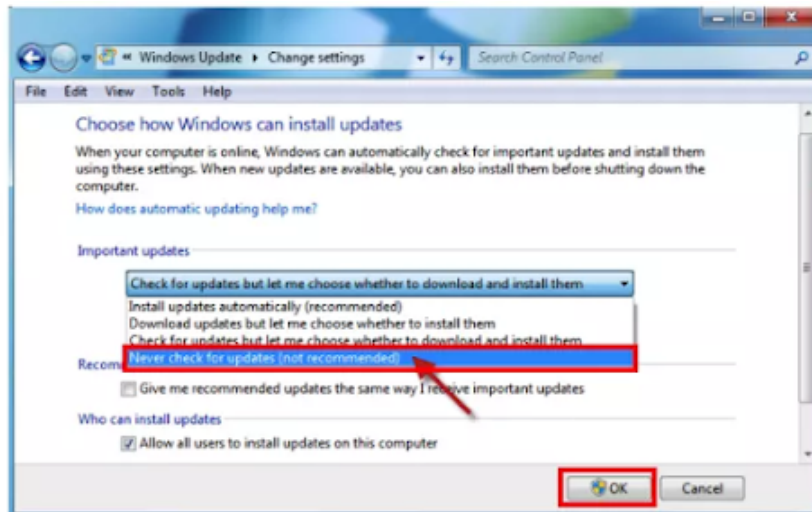
- **Disable UAC** : Head into Control Panel and type UAC into the search box, or do it from the start menu. Then drag the slider down to the bottom.



- **Disable Firewall**



- **Disable Windows AutoUpdate**

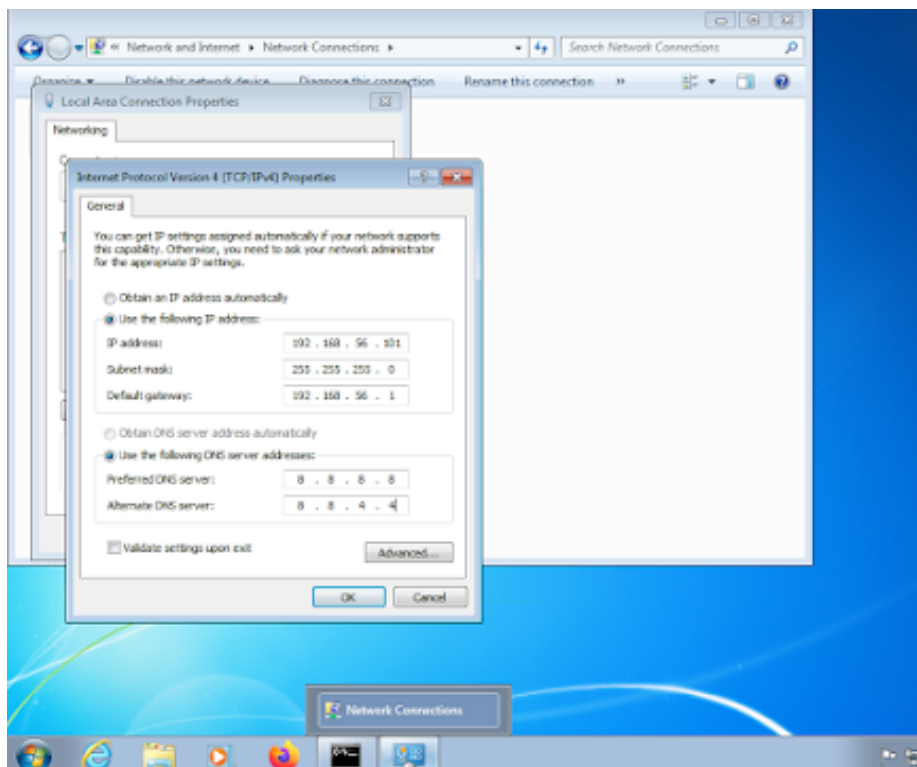


Network Configuration

It's recommended to set the VM to host-only network but we didn't experimented this. You can find some leads [here](#).

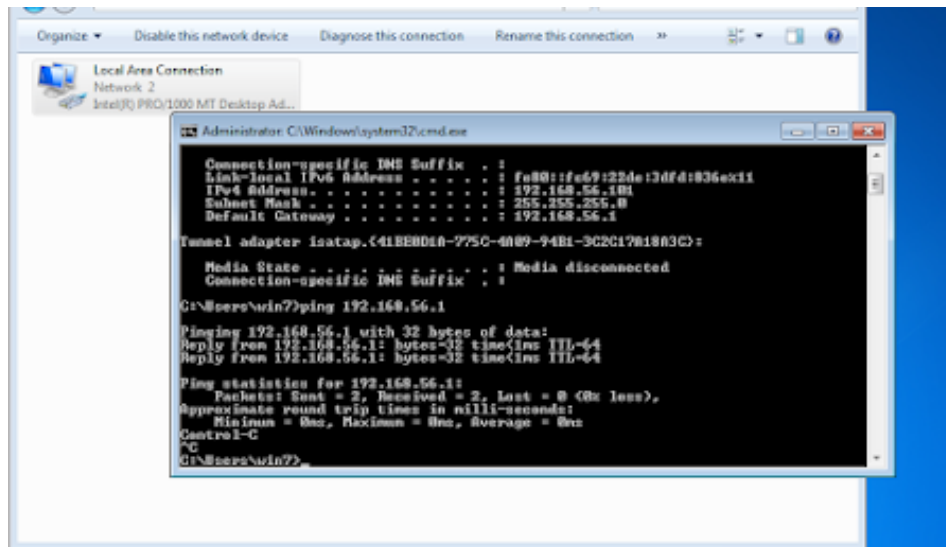
Configure the network with your own IP address and gateway, here for example we have:

- IP address: 192.168.56.101
- Subnet mask: 255.255.255.0
- Default gateway: 192.168.56.1
- Preferred DNS server: 8.8.8.8
- Alternate DNS server: 8.8.4.4



Make sure both the guest (Win7) and host (Ubuntu) can be ping each other :

```
fafa@ubuntu:~$ ping 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.
64 bytes from 192.168.56.101: icmp_seq=1 ttl=128 time=0.416 ms
64 bytes from 192.168.56.101: icmp_seq=2 ttl=128 time=0.347 ms
^C
--- 192.168.56.101 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1005ms
rtt min/avg/max/mdev = 0.347/0.381/0.416/0.034 ms
```



Disable Noisy Network Services

Teredo

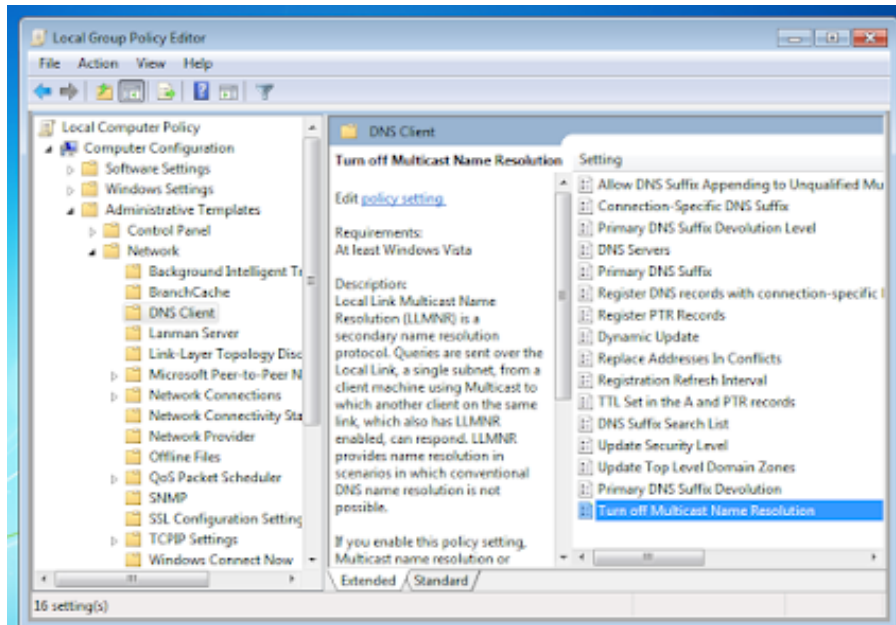
Open a command prompt as Administrator, and run :

```
netsh interface teredo set state disabled
```

Link-Local Multicast Name Resolution (LLMNR)

Open the Group Policy editor. Then navigate to Computer Configuration > Administrative Templates > Network > DNS Client, and open Turn off Multicast Name Resolution.

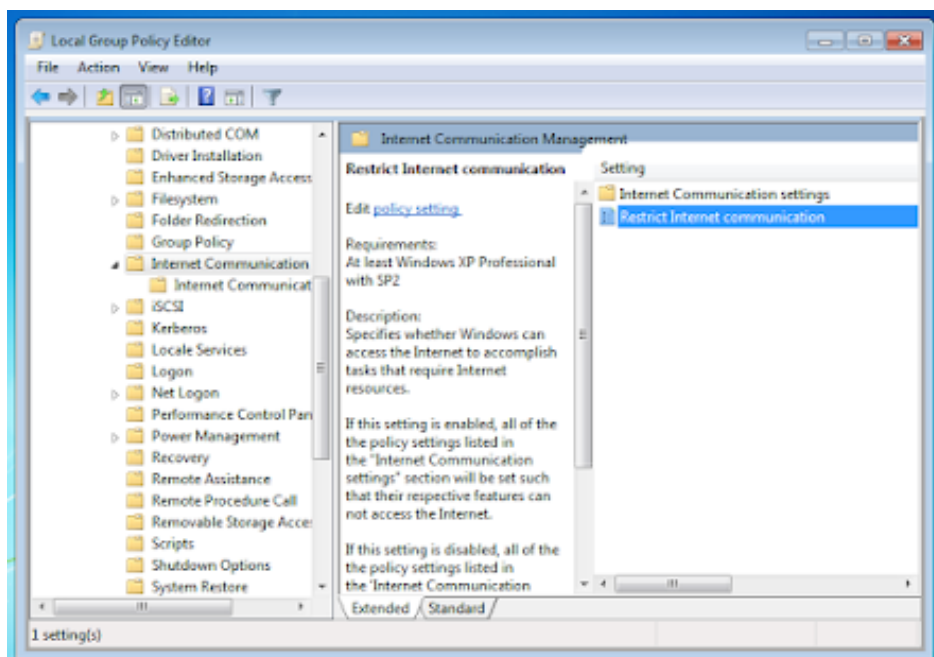
Set the policy to enabled :



Network Connectivity Status Indicator, Error Reporting, etc

Open the Group Policy. Then navigate to Computer Configuration > Administrative Templates > System > Internet Communication Management, and open Restrict Internet Communication.

Set the policy to enabled :



Install and run the CAPE Agent

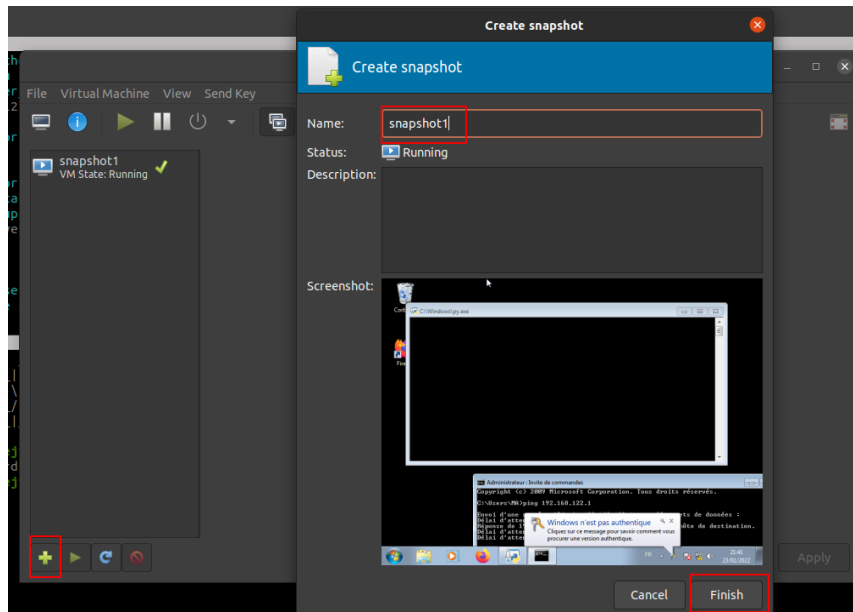
Download the agent [here](#). Copy the file into the Win7 VM.

Running (double click) the agent.py will launch the HTTP server which will be listening for connections.

If you want the script to be launched at Windows' boot, just place the file in the Startup folder. The All Users startup folder should be `C:\ProgramData\Microsoft\windows\Start Menu\Programs\Startup`.

Take VM snapshot

After start agent.py and minimize it, create a snapshot with the name "Snapshot1".



CAPE Configurations

Don't forget to read the [configurations documentation](#) to understand the configurations as yours might be different with my configuration. Important files to be configured in `/opt/CAPEv2/conf/`:

- cuckoo.conf
 - Change the IP address

```
# Delta in days from current time to set the guest clocks to for file analyses
# A negative value sets the clock back, a positive value sets it forward.
# The default of 0 disables this option
# Note that this can still be overridden by the per-analysis clock setting
# and it is not performed by default for URL analysis as it will generally
# result in SSL errors
daydelta = 0

# Path to the unix socket for running root commands.
rooter = /tmp/cuckoo-rooter

[resultserver]
# The Result Server is used to receive in real time the behavioral logs
# produced by the analyzer.
# Specify the IP address of the host. The analysis machines should be able
# to contact the host through such address, so make sure it's valid.
# NOTE: if you set resultserver IP to 0.0.0.0 you have to set the option
# 'resultserver_ip' for all your virtual machines in machinery configuration.
ip = 192.168.56.1

# Specify a port number to bind the result server on.
port = 2042
```

- auxiliary.conf
 - Configure with your own requirement

- kvm.conf
 - Change label, ip, snapshot name

```

# 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 = cuckoo1

interface = virbr0

[cuckoo1]
# Specify the label name of the current machine as specified in your
# libvirt configuration.
label = win7

# Specify the operating system platform used by current machine
# [windows/darwin/linux].
platform = windows

# Specify the IP address of the current virtual machine. Make sure that the
# IP address is valid and that the host machine is able to reach it. If not,
# the analysis will fail. You may want to configure your network settings in
# /etc/libvirt/<hypervisor>/networks/
ip = 192.168.122.165

```

- memory.conf
 - Change the guest_profile to your VM name , here win7

```

# Volatility configuration

# Basic settings
[basic]
guest_profile = win7
# Delete memory dump after volatility processing.
dostrings = yes
strings_nullterminated_only = no
strings_minchars = 5
delete_memdump = no
# Delete memory dump in the event of a volatility exception
delete_memdump_on_exception = no

```

- reporting.conf
 - Configure with your own requirement
- web.conf
 - If you'd like you can enable scoring :

```

rps = 1/rps
rpm = 5/rpm

# Show submit to all VMs on webgui
[all_vms]
enabled = no

[admin]
enabled = no

[comments]
enabled = no

#enable linux fields on webgui
[linux]
#For advanced users only, can be buggy, linux analysis is work in progress for fun
enabled = no

[malscore]
enabled = yes

[vtupload]
enabled = no

```

- Or configure with your own requirement

Run CAPE and Webserver

Run Cuckoo

First you have to make last installations in order to allow the replay of HTTP and HTTPS requests :

```
$ cd /opt/CAPEv2/utls
$ sudo python3 community.py -cr
$ sudo pip3 install -U git+https://github.com/CAPEsandbox/httpreplay
```

Then we just have to run cuckoo :

```
$ cd /opt/CAPEv2/
$ sudo python3 cuckoo.py
```

Run Webserver

For web, in a new tab, run these command :

```
$ cd /opt/CAPEv2/web
$ sudo python3 manage.py migrate
$ sudo python3 manage.py runserver 0.0.0.0:8090
```

Run Processing

If you want CAPE to process the information you should run :

```
sudo python3 process.py -p7 auto
```

If get an error on this command, try to lower the number of parallels threads used with the flag `-px` and X equal to the number of threads used.

OpenCTI

You can choose the way you want to install OpenCTI on the [official wiki](#).

It is recommended to use **Docker**, you can follow the official OpenCTI docker [installation instructions](#).

MISP

Docker

MISP developed a Docker container, you can deploy it following the Github [installation instructions](#).

Manual deployment

If you wish to install MISP on your machine, you can follow the official [installation instructions](#) of MISP.

Connect Instances

CAPE ↔ OpenCTI

We will use the [CAPE connector](#) developed by OpenCTI team. Add the code of the *Connector docker-compose.yml* content to your *OpenCTI docker-compose*.

! Warning :

You will probably encounter network problem saying `API is not reachable`. Indeed your CAPE instance is running on your machine and your OpenCTI instance on docker. This docker has his own internal network. In order to bypass those network problem, you should add `host.gateway` parameter to your docker-compose file. You can change the *host.docker.internal* to whatever you'd like.

```
connector-cape:
  image: opencti/connector-cape:rolling
  environment:
    - OPENCTI_URL=http://opencti:8081
    - OPENCTI_TOKEN=
    - CONNECTOR_ID=2
    - CONNECTOR_TYPE=EXTERNAL_IMPORT
    - CONNECTOR_NAME=CAPE
    - CONNECTOR_CONFIDENCE_LEVEL=15 # From 0 (Unknown) to 100 (Fully trusted)
    - CONNECTOR_UPDATE_EXISTING_DATA=true
    - CONNECTOR_LOG_LEVEL=info
    - CAPE_CREATE_INDICATORS=true
    - CAPE_ENABLE_NETWORK_TRAFFIC=false # enable creation of net Traffic (Very Loud)
    - CAPE_ENABLE_REGISTRY_KEYS=false # enable creation of Created registry Keys (Very Loud)
    - CAPE_API_URL=http://host.docker.internal:8090/api/v2/ # CAPE API EP
    - CAPE_BASE_URL=http://host.docker.internal:8090 # CAPE Web UI URL
    - CAPE_INTERVAL=30 # In Min
    - CAPE_START_TASK_ID=0 # In Min
    - CAPE_REPORT_SCORE=7
  extra_hosts:
    - "host.docker.internal:host-gateway"
  restart: always
  depends_on:
    - opencti
```

CAPE ↔ MISP

The functionality to transfer data to MISP is already built into CAPE. There is a MISP section in the [reporting.conf](#) file that allows you to enter the parameters of the MISP instance to be connected.

```
[Misp]
enabled = yes
apikey = 30pTMMgn9wtsRC6aZAKj5BCBMCjB2QcWron40lnI
url = https://localhost/
Make event published after creation?
published = no
minimal malscore, by default all
min_malscore = 0
by default 5 threads
threads =
this will retrieve information for Iocs
and activate MISP report download from webgui
extend_context = yes
upload Iocs from cuckoo to MISP
upload Iocs = yes
distribution = 0
threat_level_id = 2
analysis = 2
Sections to report
Analysis ID will be appended, change
title = Iocs from cuckoo analysis:
network = yes
ids_files = yes
ropped = yes
registry = yes
textes = yes
```

MISP ↔ OpenCTI

Like CAPE, we will use [MISP official connector](#) developed by OpenCTI.

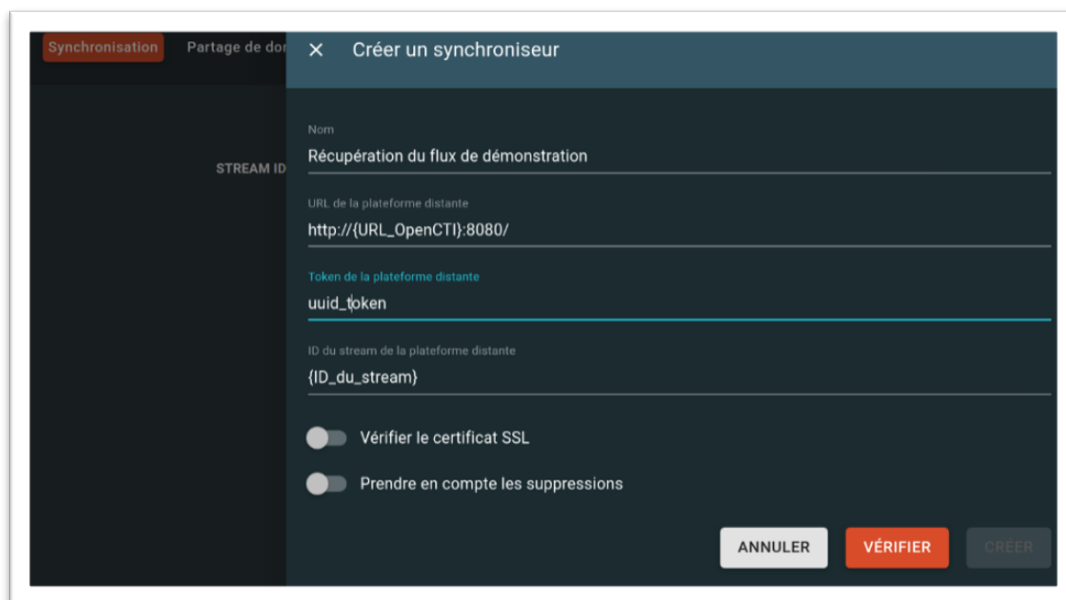
As mentioned in the github connector README :

Enabling this connector could be done by launching the Python process directly after providing the correct configuration in the config.yml file or within a Docker with the image opencti/connector-misp:latest. We provide an example of docker-compose.yml file that could be used independently or integrated to the global docker-compose.yml file of OpenCTI.

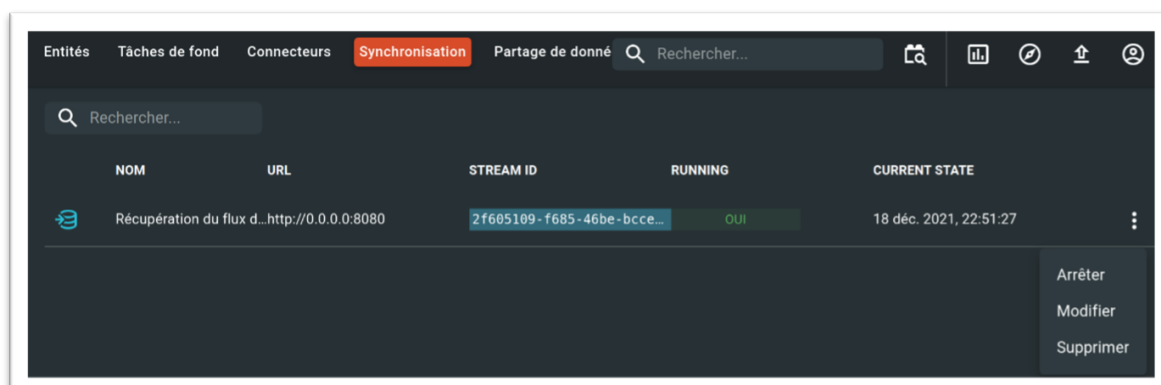
You could encounter the [same network problem](#) as for Cape Connexion. Be sure your docker container are on the same network if your use MISP with docker. Otherwise if you had manually deployed MISP add the `host.gateway` argument in your *docker-compose.yml*.

OpenCTI ↔ OpenCTI

It is possible to retrieve this stream by creating a synchroniser. All you need is the URL of the OpenCTI instance, an authentication token and the ID of the stream you want to retrieve.



Once the stream has been created, all that remains is to start or stop it.



NOM	URL	STREAM ID	RUNNING	CURRENT STATE
Récupération du flux d...	http://0.0.0.0:8080	2f605109-f685-46be-bcce...	OUI	18 déc. 2021, 22:51:27

There are others methods, please refer to [this article](#) on data sharing with OpenCTI.

Sources

- [CAPE v2 installation](#) by NetBySec
- [CAPE v2 official documentation](#)
- [CAPE installation tutorial](#) by CTechmat
- [OpenCTI wiki](#)
- [OpenCTI data sharing article](#) by Luatix