SOC

Le centre d'opération de cyberdéfense est là pour vous protéger !

- [SOC] Outils de cyber défense
- [SOC] Afficher l'empreinte d'un fichier
- [SOC] YARA
- [SOC] Dettect
- [SOC] Graylog
- [SOC] Cyberchef

[SOC] Outils de cyber défense

Introduction

Cette page répertorie une base d'outils de cyber défense utile à un SOC ou même à un administrateur aguéri et conscient des enjeux cyber.

Scanner d'URL

Une multitude d'outils existent pour scanner les URL grâce à leur réputation.

Voici une liste d'outils :

- VirusTotal

- MetaDefender Cloud - OPSWAT
- URLscan.io
- URLhaus
- WhoIs
- Cisco Talos Intelligence
- ThreatMiner
- Brightcloud

Scanner d'adresse IP

Voici une liste d'outils en ligne pour analyser la réputation des adresses IP :

- AbuseIPDB
- MetaDefender Cloud - OPSWAT
- Cisco Talos Intelligence
- Feodo Tracker
- Threatbook
- Pulsedive
- Shodan
- XForceIBM
- Alienvault
- GreyNoise

Scanner de fichier et de hash

Voici une liste d'outils en ligne pour analyser des fichiers et des hashs de fichiers :

- VirusTotal
- MetaDefender Cloud - OPSWAT
- MalwareBazaar
- Malshare
- Cisco Talos Intelligence
- HybridAnalysis

Règles de détection

- SOC Prime Threat Detection Marketplace

Blacklist d'empreintes de certificats SSL

- SSL Blacklist

Scanner de mail

Pour se protéger du phishing, il peut être utile d'utiliser des outils qui vont décortiquer le mail afin de potentiellement trouver des IOC.

- PhishTool

Sandbox

Les sandboxs permettent de tester le comportement de fichier ou de site web afin de détecter un comportement malveillant dans un environnement protégé.

- Browserling : Accéder à un site web dans une sandbox.
- AnyRun : Lancer un exécutable Windows dans une sandbox.
- Wannabrowser : Accéder à un site web dans une sandbox.

Matrice MITRE ATT&CK

Framework permettant notamment de lister les TTPs et les APT.

- MITRE ATT&CK

Liste d'IOC multiples

- ThreatFox : MISP events, Suricata IDS Ruleset, Domain Host files, DNS Response Policy Zone, JSON files and CSV files.

PS - Obtenir le hash d'un fichier

```
Get-FileHash <FILE> -Algorithm MD5
```

[SOC] Afficher l'empreinte d'un fichier

Introduction

Dans le cadre du travail d'analyste, il peut être intéressant d'obtenir l'empreinte d'un fichier sous ses différentes formes (**MD5**, **SHA-1** ou **SHA-256**).

Ce tutoriel traitera la méthodologie à suivre sous Linux exclusivement.

Manuel

MD5

```
md5sum <FILE>
```

SHA-1

```
sha1sum <FILE>
```

SHA-256

```
sha256sum <FILE>
```

Changer l'empreinte d'un fichier

Il est possible de changer l'empreinte d'un fichier sans (presque) altérer son contenu en ajoutant un **null byte** à la fin de celui-ci :

```
echo -n -e "\x00" >> file.txt
```

> Cette technique est très pratique pour échapper à la détection basée sur la signature mais permet aussi d'envoyer des échantillons de malware sur des plateformes publiques sans que celui-ci soit retrouvé par l'éditeur.

[SOC] YARA

Introduction

Le **YARA** est un langage pour écrire des règles de détection de malware.

Il s'agit d'un langage simple et descriptif qui est adopté par le grand public.

Il est découpé par section qui ont chacune leur utilité.

Source

- TryHackMe - Yara
- Cuckoosandbox - Sandbox pour tester vos règles Yara
- PEfile - Scan les exécutables Windows PE

Annexes

- Github - Awesome Yara
- Valhalla - Base de règles Yara Opensource

Anatomie d'une règle

Anatomy of a yara Rule

Yara is a tool used to identify file, based on textual or binary pattern.

A rule consists of a set of strings and conditions that determine its logic.

Rules can be compiled with "yarac" to increase the speed of multiple Yara scans.

1 Import Module

Yara modules allow you to extend its functionality. The PE module can be used to match specific data from a PE:

- pe.number_of_exports
- pe.sections[0].name
- pe.imphash()
- pe.imports("kernel32.dll")
- pe.is_dll()

List of modules: pe, elf, hash, math, cuckoo, dotnet, time

2 Rule Name

The rule name identifies your Yara rule. It is recommended to add a meaningful name. There are different types of rules:

- Global rules: applies for all your rules in the file.
- Private rules: can be called in a condition of a rule but not reported.
- Rule tags: used to filter yara's output.

3 Metadata

Rules can also have a metadata section where you can put additional information about your rule.

- Author
- Date
- Description
- Etc...

```
import "pe"

rule demo_rule : Tag1 Demo
{
    meta:
        author = "Thomas Roccia"
        description = "demo"
        hash = ""

    strings:
        $string0 = "hello" nocase wide
        $string1 = "world" fullword ascii
        $hex1 = { 01 23 45 ?? 89 ab cd ef }
        $re1 = /md5: [0-9a-zA-Z]{32}/

    condition:
        uint16(0) == 0x5A4D and filesize < 2000KB
        or pe.number_of_sections == 1 and
        any of ($string*) and (not $hex1 or $re1)
}
```

4 Strings

The field strings is used to define the strings that should match your rule. It exists 3 type of strings:

- Text strings
- Hexadecimal strings
- Regex

Text Strings

Text strings can be used with modifiers:

- nocase: case insensitive
- wide: encoded strings with 2 bytes per character
- fullword: non alphanumeric
- xor(0x01-0xff): look for xor encryption
- base64: base64 encoding

Hexadecimal

Hex strings can be used to match piece of code:

- Wild-cards: { 00 ?2 A? }
- Jump: { 3B [2-4] B4 }
- Alternatives: { F4 (B4 | 56) }

Regex

Regular expression can also be used and defined as text strings but enclosed in forward slash.

5 Condition

Conditions are Boolean expressions used to match the defined pattern.

- Boolean operators:
  - and, or, not
  - <=, >=, ==, <, >, !=
- Arithmetic operators:
  - +, -, *, \, %
- Bitwise operators:
  - &, |, <<, >>, ^, ~
- Counting strings:
  - #string0 == 5
- Strings offset:
  - $string1 at 100

Advanced Condition

- Accessing data at a given position: uint16(0) == 0x5A4D
- Check the size of the file: filesize < 2000KB
- Set of strings: any of ($string0, $hex1)
- Same condition to many strings: for all of them : (# > 3)
- Scan entry point: $value at pe.entry_point
- Match length: !re1[1] == 32
- Search within a range of offsets: $value in (0..100)

@FrØgger_
Thomas Roccia

Manuel

Installation

```
apt install -y yara
```

Meta

Cette section permet de donner des informations complémentaires qui ne seront pas interprêtés comme le ferait un commentaire dans du code.

Par exemple on peut utiliser le mot-clé **desc**, pour donner une description à notre règle afin qu'elle soit plus explicite pour les utilisateurs.

Strings

Cette section permet de détecter des chaînes de caractères présente dans les fichiers.

Voici un exemple d'utilisation :

```
rule helloworld_checker{
strings:
$hello_world = "Hello World!"

condition:
$hello_world
}
```

On peut aussi détecter des chaînes multiples :

```
rule helloworld_checker{
strings:
$hello_world = "Hello World!"
$hello_world_lowercase = "hello world"
$hello_world_uppercase = "HELLO WORLD"

condition:
any of them
}
```

Opérateurs

Comme dans les langages de programmation traditionnels, on peut utiliser des opérateurs pour nos conditions :

```
rule helloworld_checker{
strings:
$hello_world = "Hello World!"

condition:
    #hello_world <= 10
}
```

Opérateurs	Descriptions
<=	Plus petit ou égal
>=	Plus grand ou égal
!=	Différent de

Combinaisons

On peut utiliser les mot-clés suivants pour combiner nos conditions :

Mot-clés	Descriptions
and	Les deux conditions doivent être valides
or	Au moins l'une des deux conditions doit être valide
not	Inverse la condition (true devient false et false devient true)

Voici un exemple pour vérifier si la chaîne est présente et si la taille du fichier est inférieure à 10KB :

```
rule helloworld_checker{
strings:
$hello_world = "Hello World!"

    condition:
     $hello_world and filesize < 10KB
}
```

Lancer le scan

```
yara <RULE>.yar <FILE_TO_SCAN>
```

> Si la règle match, la commande renverra le nom de la règle qui a matchée ainsi que le nom du fichier qui a matché.

Scanners d'IOC basés sur Yara

- Loki
- THOR
- Fenrir
- YAYA

Loki

Mettre à jour la base de signature

```
python loki.py --update
```

Lancer un scan d'un dossier

```
python loki.py -p <DIR>
```

YarGen

Cet outil permet de créer une règle Yara à partir d'un ou plusieurs fichiers connus pour être malveillants.

Il va se baser sur les chaînes de caractères et les informations pour générer la règle qui va détecter le ou les fichiers.

Téléchargement

- Github - YarGen

Mettre à jour l'outil

```
python3 yarGen.py --update
```

> Cela va mettre à jour la base avec les chaînes et les opcodes.

Créer une règle

```
python3 yarGen.py -m <FILE_PATH> --excludegood -o <OUTPUT.yar>
```

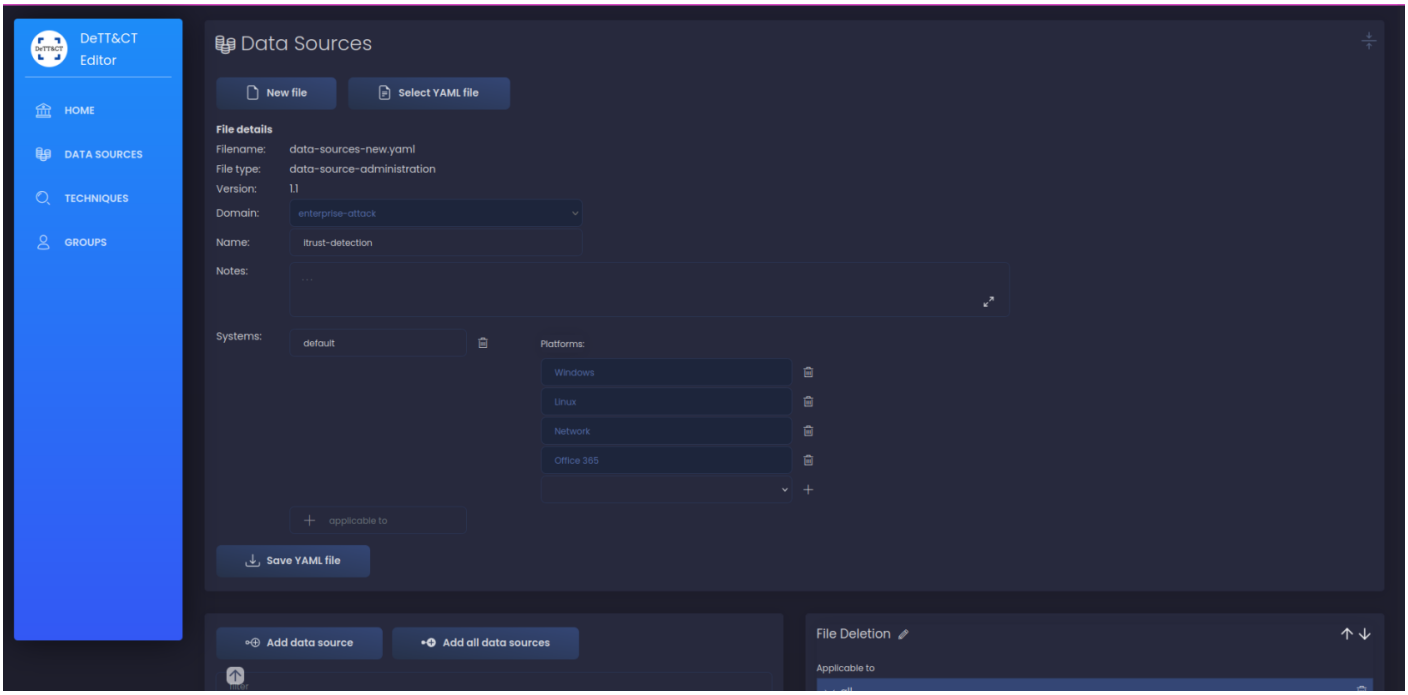
> Bien que l'outil soit fonctionnel, il est recommandé d'éditer la règle pour supprimer les chaînes qui pourraient lever des faux-positifs.

[SOC] Dettect

Introduction

Le projet Dettect a pour objectif d'identifier les TTPs couvertes (et non-couvertes) par vos règles de détection.

Le projet vous aidera à générer un fichier avec vos datasources couvertes et à convertir ce fichier en un fichier importable dans le MITRE Navigator afin d'afficher les TTPs.



DeTT&CT

Voici le lien du projet :

- https://github.com/rabobank-cdc/DeTTECT

Lancez le conteneur :

```
docker run -p 8080:8080 -v $(pwd)/output:/opt/DeTTECT/output -v $(pwd)/input:/opt/DeTTECT/input --name dettect -it rabobankcdc/dettect:latest /bin/bash
```

Puis lancez le serveur web en écoute :

```
python3 dettect.py e
```

> Vous pouvez ouvrir un navigateur web et vous rendre sur http://localhost:8080 .

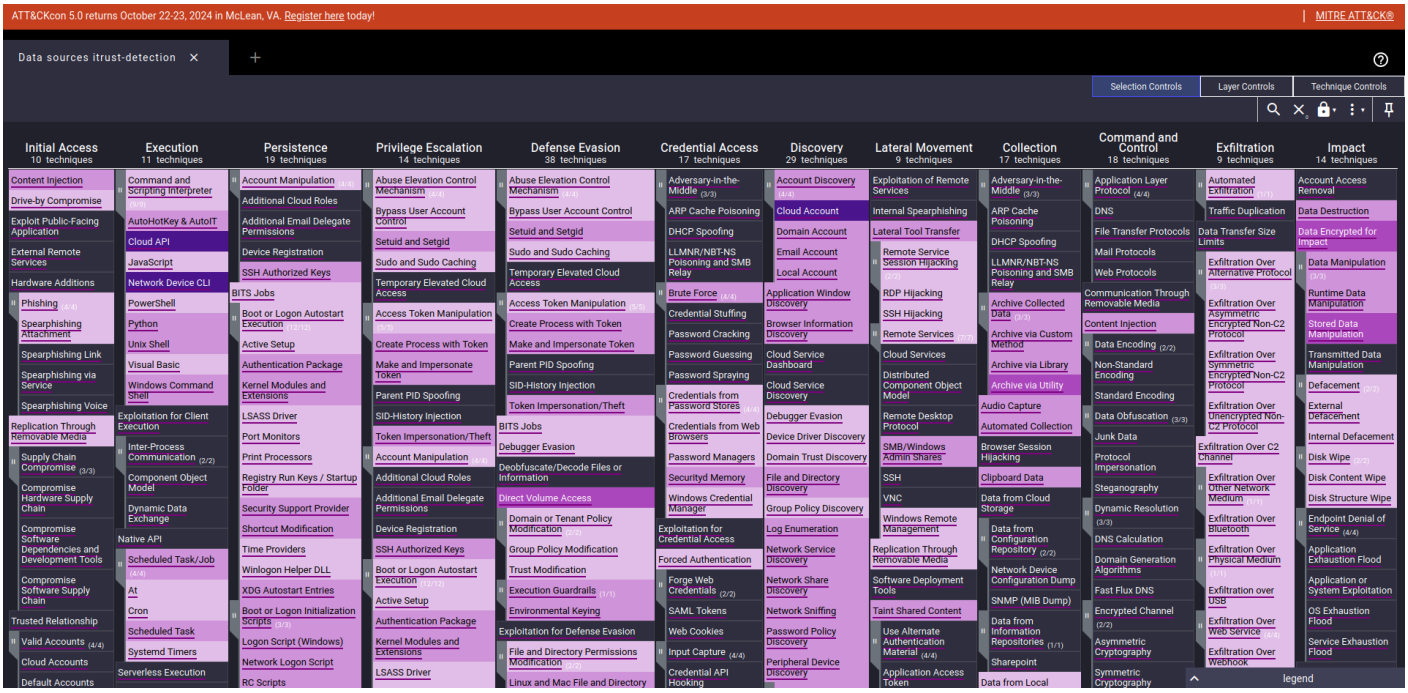
- Après avoir créer vos datasources et télécharger votre configuration, déplacez-le dans le dossier input du projet.
- Ensuite ouvrez un shell dans le conteneur :

```
docker exec -it dettect bash
```

Puis convertissez pour obtenir un fichier de configuration importable dans le MITRE Navigator :

```
python3 dettect.py ds -fd input/data-sources-new.yaml -l
```

Et importez dans le MITRE Navigator :



[SOC] Graylog

Introduction

Graylog est un SIEM qui permet de centraliser vos logs et de faire des queries pour faire des analyses de threat hunting notamment.

Sources

- https://www.it-connect.fr/tuto-graylog-sur-debian-centraliser-et-analyser-logs/
- https://www.it-connect.fr/envoyer-les-logs-windows-vers-graylog-avec-nxlog/

Installation

Serveur (Debian 12)

Tout d'abord, configurez correctement la timezone ou le serveur de temps :

```
timedatectl set-timezone Europe/Paris
```

Installez quelques outils de base dont nous auront besoin pour la suite :

```
apt update && apt install -y install curl lsb-release ca-certificates gnupg2 pwgen
```

MongoDB

```
curl -fsSL https://www.mongodb.org/static/pgp/server-6.0.asc | sudo gpg -o /usr/share/keyrings/mongodb-server-6.0.gpg --dearmor && echo "deb [ signed-by=/usr/share/keyrings/mongodb-server-6.0.gpg] http://repo.mongodb.org/apt/debian bullseye/mongodb-org/6.0 main" | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list && apt update && apt-get install -y mongodb-org
```

> Vous allez obtenir une erreur par rapport à **libssl** à ce stade et c'est normal, passez à la suite.

Rendez-vous sur le site suivant pour trouver la version la plus récente de libssl et copiez le lien :

- http://archive.ubuntu.com/ubuntu/pool/main/o/openssl/

> Faite CTRL+F et cherchez "**libssl1.1_1.1.1f-1ubuntu**2." puis sélectionnez la version deb amd64.

Une fois l'URL récupérée, vous devriez faire quelque chose comme ça :

```
wget http://archive.ubuntu.com/ubuntu/pool/main/o/openssl/libssl1.1_1.1.1f-1ubuntu2.23_amd64.deb && dpkg -i libssl1.1_1.1.1f-1ubuntu2.23_amd64.deb
```

Puis réinstallez le paquet **mongodb-org** :

```
apt install -y mongodb-org
```

Et lancez le service :

```
sudo systemctl daemon-reload && sudo systemctl enable --now mongod.service
```

> Si l'installation de MongoDB s'est mal passée c'est que vous n'avez pas téléchargé le bon paquet sur le site d'Ubuntu.

> Si l'installation de MongoDB s'est bien passée mais que le service ne démarre pas, assurez-vous de passer le CPU en mode host s'il s'agit d'une VM.

Opensearch

```
curl -o- https://artifacts.opensearch.org/publickeys/opensearch.pgp | sudo gpg --dearmor --batch --yes -o /usr/share/keyrings/opensearch-keyring && echo "deb [signed-by=/usr/share/keyrings/opensearch-keyring] https://artifacts.opensearch.org/releases/bundle/opensearch/2.x/apt stable main" | sudo tee /etc/apt/sources.list.d/opensearch-2.x.list && apt update
```

```
env OPENSEARCH_INITIAL_ADMIN_PASSWORD=<PASSWORD> apt-get install opensearch
```

> Choisissez un mot de passe d'au moins 8 caractères avec minuscule, majuscule, chiffre et caractère spécial sinon l'installation d'opensearch échouera.

Maintenant ouvrez le fichier de configuration **/etc/opensearch/opensearch.yml** et ajustez la configuration avec les éléments suivants :

```
cluster.name: graylog
node.name: ${HOSTNAME}
path.data: /var/lib/opensearch
path.logs: /var/log/opensearch
discovery.type: single-node
network.host: 127.0.0.1
action.auto_create_index: false
plugins.security.disabled: true
```

Java

Éditez le fichier de configuration **/etc/opensearch/jvm.options** pour définir la ram utilisée par Java (minimum 4G) :

```
-Xms4g
-Xmx4g
```

Vérifiez que le max_map_count est définit à **262144** :

```
cat /proc/sys/vm/max_map_count
```

Si ce n'est pas le cas (uniquement) :

```
sysctl -w vm.max_map_count=262144
```

Une fois que java est configuré, lancez et activez le service opensearch :

```
systemctl daemon-reload && systemctl enable --now opensearch
```

Graylog

```
wget https://packages.graylog2.org/repo/packages/graylog-6.1-repository_latest.deb && dpkg -i graylog-6.1-repository_latest.deb && apt update && apt install -y graylog-server
```

Avant de lancez Graylog il faut configurer le **password_secret** que vous pouvez générer avec la commande suivante :

```
pwgen -N 1 -s 96
```

Renseignez le à l'endroit adéquat dans le fichier **/etc/graylog/server/server.conf** .

Ensuite il faut configurer le mot de passe admin de Graylog. Pour cela on doit calculer son hash :

```
echo -n "<PASSWORD>" | shasum -a 256
```

Et renseignez le dans le fichier **/etc/graylog/server/server.conf** au niveau du champ **root_password_sha2** .

Toujours dans le fichier de configuration, définissez les paramètres suivants :

```
http_bind_address = 0.0.0.0:9000
elasticsearch_hosts = http://127.0.0.1:9200
```

Et enfin, activez et démarrez Graylog :

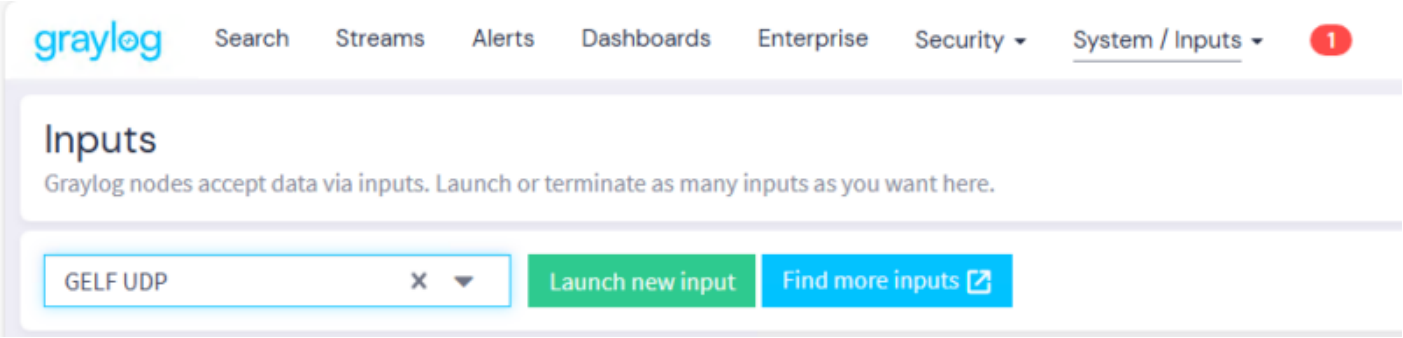
```
systemctl enable --now graylog-server
```

> Graylog est désormais accessible via **http://<IP>:9000** .

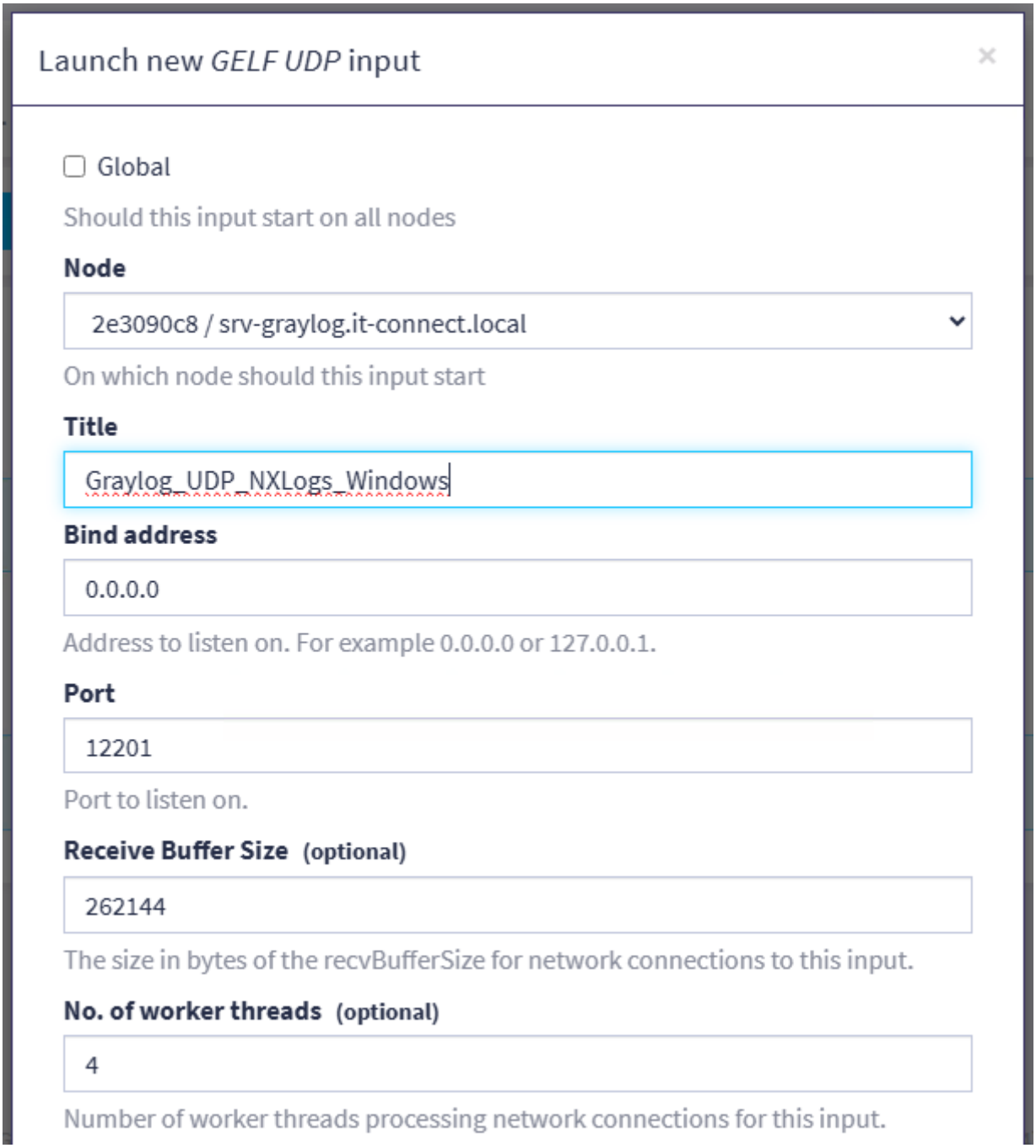
Agent Windows

Tout d'abord il faut configurer Graylog pour recevoir les logs.

Pour cela rendez-vous dans **System > Inputs** et sélectionnez **GELF UDP** comme ci et cliquez sur **Launch new input** :



Configurez de la sorte (sauf pour le titre, mettez ce que vous souhaitez) :



Vous devriez voir quelque chose comme ça :

Local inputs 1 configured

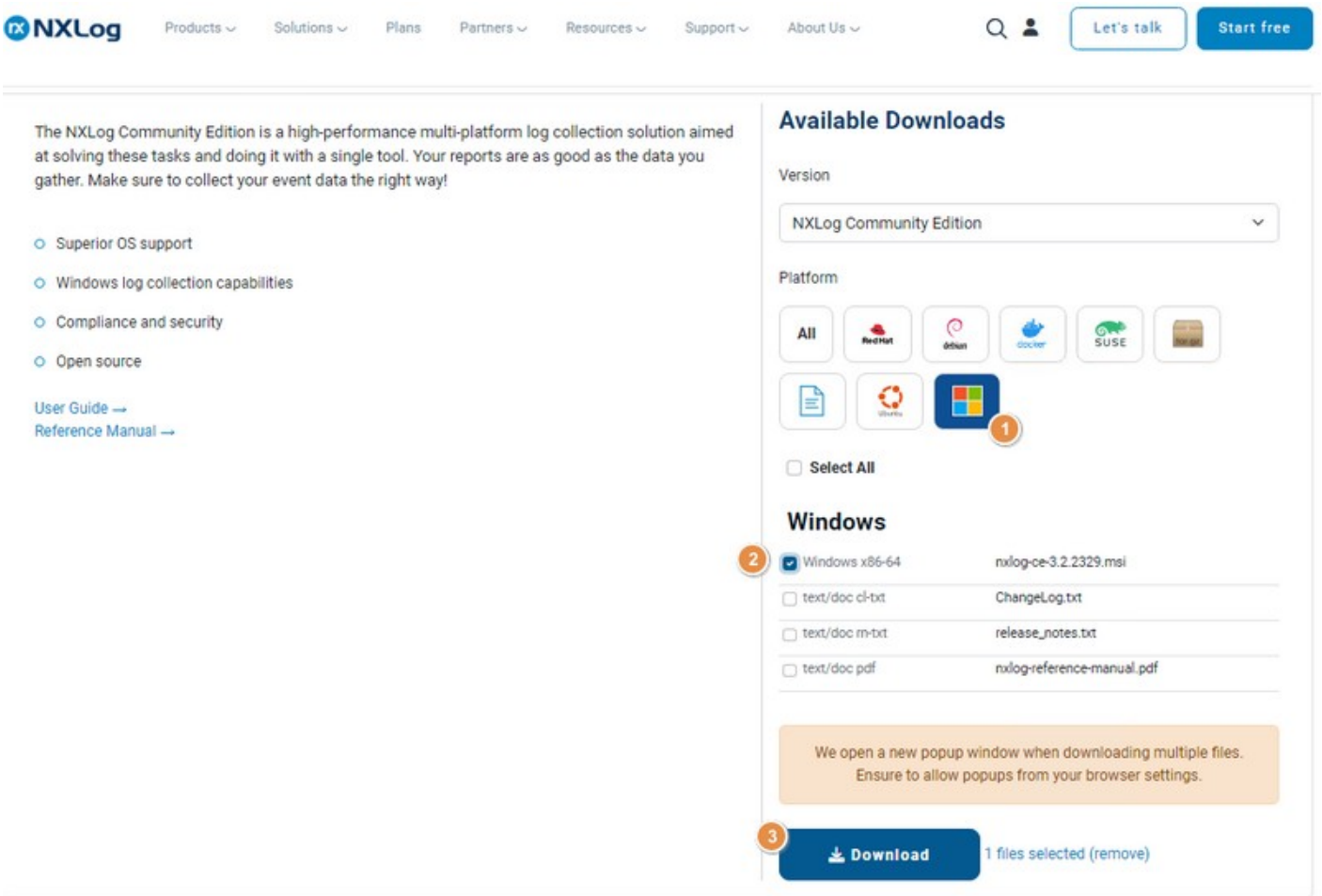
Graylog_UDP_NXLogs_Windows GELF UDP (6721f8280ac6780828d1f9ad) RUNNING

On node ⭐ 2e3090c8 / srv-graylog.it-connect.local

```
bind_address: 0.0.0.0
charset_name: UTF-8
decompress_size_limit: 8388608
number_worker_threads: 4
override_source: <empty>
port: 12201
recv_buffer_size: 262144
```

Maintenant téléchargez l'agent **NXLog** sur la machine qui va envoyer les logs :

- https://nxlog.co/downloads/nxlog-ce#nxlog-community-edition



Saisissez la configuration suivante dans le fichier **C:\Program Files\nxlog\conf\nxlog.conf** :

```
# Récupérer les journaux de l'observateur d'événements
<Input in>
```

```
   Module      im_msvistalog
</Input>

# Déclarer le serveur Graylog (selon input)
<Extension gelf>
   Module        xm_gelf
</Extension>

<Output graylog_udp>
   Module        om_udp
   Host          192.168.10.220
   Port  12201
   OutputType    GELF_UDP
</Output>

# Routage des flux in vers out
 <Route 1>
    Path        in => graylog_udp
 </Route>
```

> Celle-ci enverra tous les logs de la machine sur Graylog, ce qui n'est pas forcément nécessaire, on pourrait envoyer seulement les logs **Security** et appliquer la configuration suivante :

```
# Récupérer les journaux Security de l'observateur d'événements
<Input in>
   Module      im_msvistalog
   <QueryXML>
      <QueryList>
         <Query Id='1'>
            <Select Path='Security'>*</Select>
         </Query>
      </QueryList>
   </QueryXML>
</Input>

# Déclarer le serveur Graylog (selon input)
<Extension gelf>
   Module        xm_gelf
```

```
</Extension>

<Output graylog_udp>
   Module        om_udp
   Host          192.168.10.220
   Port  12201
   OutputType    GELF_UDP
</Output>

# Routage des flux in vers out
 <Route 1>
    Path        in => graylog_udp
 </Route>
```

Désormais relancez le service NXlog pour appliquer la configuration :

```
Restart-Service nxlog
```

> Le fichier de log de NXlog est le fichier **C:\Program Files\nxlog\data\nxlog.log** si vous devez debugger.

[SOC] Cyberchef

Introduction

Cyberchef est un outil pour décoder, formater etc très utile en CTF ou dans les tâches quotidiennes.

Sources

- https://gchq.github.io/CyberChef/
- https://kravensecurity.com/cyberchef-guide/#CyberChef

Cyberchef API

Un projet a été réalisé, permettant d'exécuter des recettes via des call API (très pratique pour de l'automatisation :

- https://github.com/gchq/CyberChef-server

Pour l'installer (docker requis) :

```
git clone https://github.com/gchq/CyberChef-server && cd CyberChef-server && docker build -t cyberchef-server .
```

Pour lancer le serveur :

```
docker run -it --rm --name=cyberchef-server -p 3000:3000 cyberchef-server
```

Ensuite, par exemple, vous pourriez extraire des IPv4 d'un texte brut avec une simple requête curl :

```
curl -X POST -H "Content-Type: application/json" -d '{"input":"120[.]89[.]71[.]226 120[.]89[.]71[.]226:9090 aa", "recipe":[{"op":"Fang URL", "args":[true, false]}, {"op":"Extract IP addresses", "args":["IPv4", false]}]}' http://localhost:3000/bake
```