UNIVERSITÀ DEGLI STUDI
DEL SANNIO Benevento
DING
DIPARTIMENTO DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Informatica

Progetto di Sicurezza delle Reti e dei Sistemi Software

PIATTAFORMA OPENCTI

Docente:
Corrado Aaron Visaggio

Supervisore:
Ing. Pietro Melillo

Gruppo 4:
Luigi Di Tuccio
Manuel Todesca

Anno Accademico 2021/2022

Indice

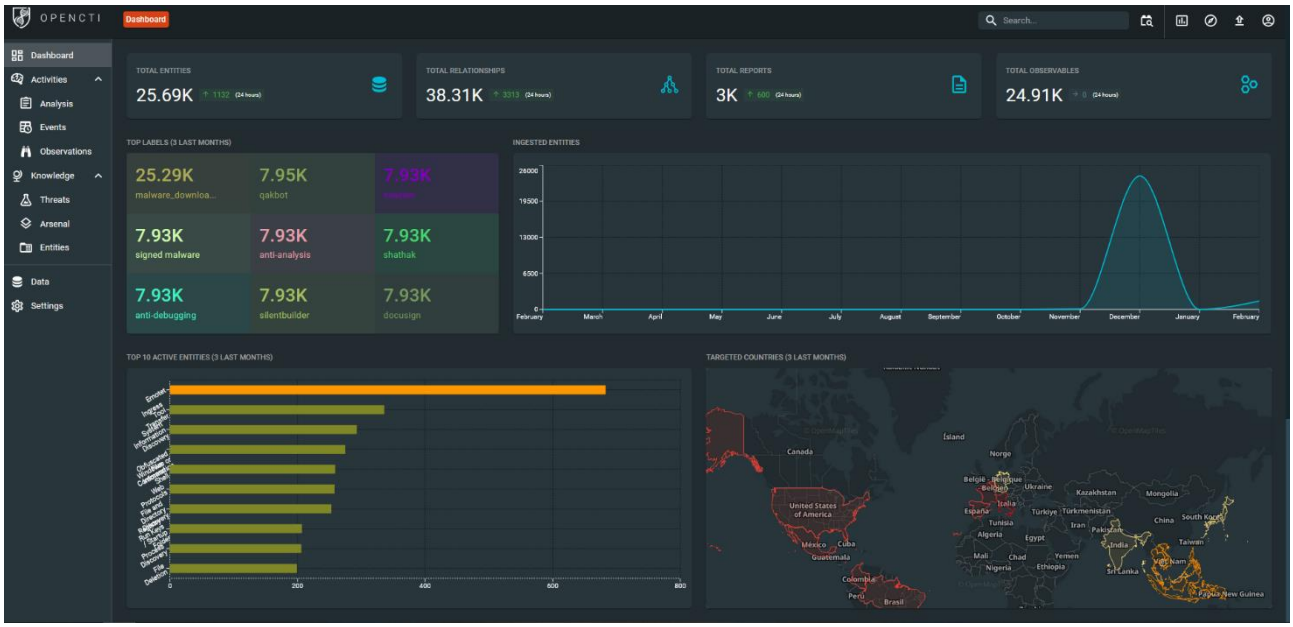
1 Introduzione 3
2. Architettura 4
2.1 Connettori 5
2.2 Requisiti Minimi e Configurazione della VM 6
3 Installazione dei connettori 7
4.Connetori custom 10
4.1 YOROI 10
4.2 Cluster25 13
5. Export dei dati 16

1 Introduzione

OpenCTI è una piattaforma open source di Cyber Threat Intelligence che fornisce un potente database di gestione delle conoscenze informatiche, permettendo di strutturare, archiviare e visualizzare diverse tipologie di informazioni: categorizzazione dell'evento, vittimologia, settore di attività e localizzazione.

I dati sono strutturati utilizzando uno schema di conoscenza basato sullo standard STIX2, possono essere integrati tramite connettori specifici verso piattaforme, servizi e repository esterni.

La piattaforma è stata realizzata come una moderna applicazione web che include un'API GraphQL e un frontend orientato alla User Experience.



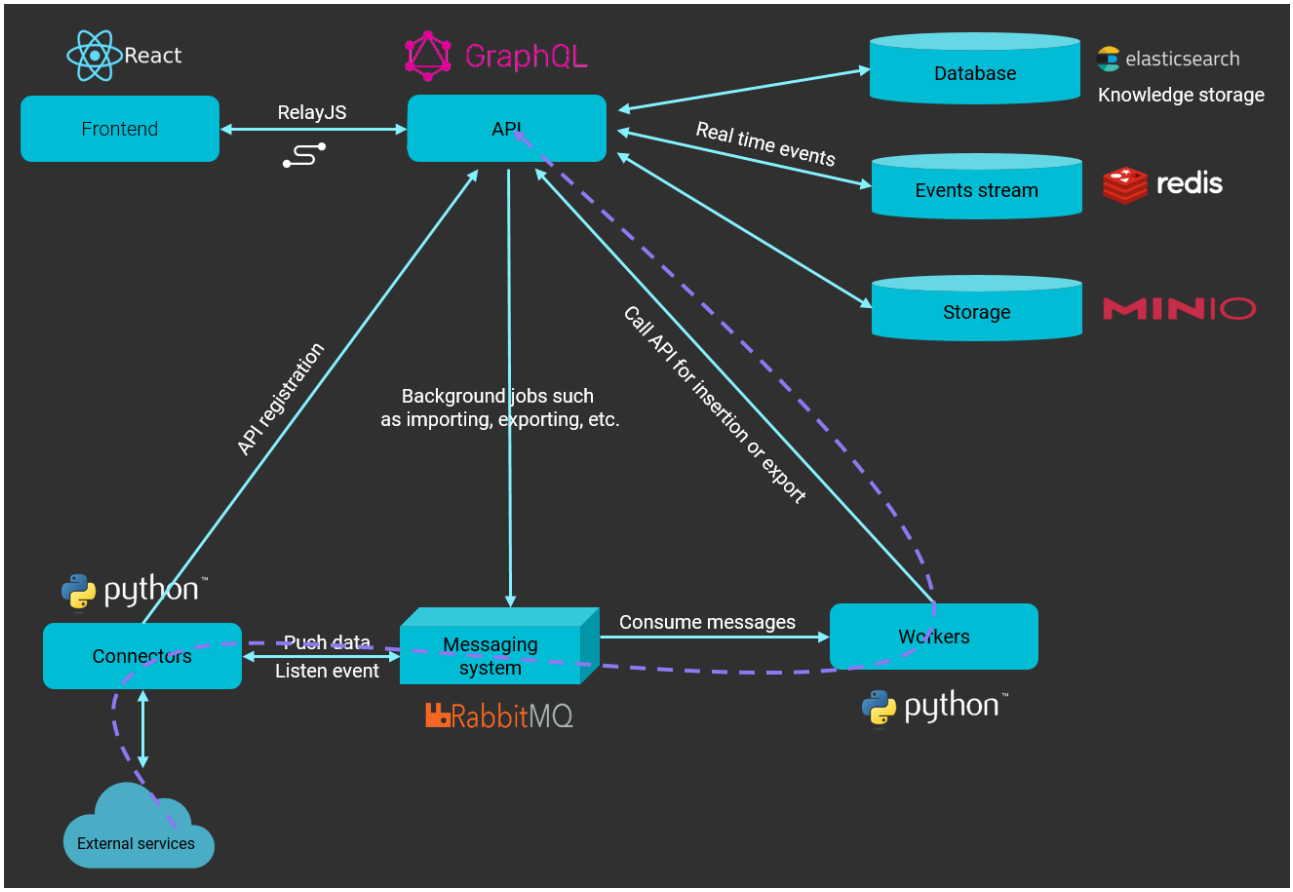
Obiettivo

L'obiettivo della piattaforma è creare uno tool completo che consenta agli utenti di integrare informazioni tecniche (come TTP e osservabili) e informazioni non tecniche (come vittimologia, settore di attività e localizzazione), collegando ogni informazione alla sua fonte (un report, un evento MISP, ecc.) e fornendo funzionalità come collegamenti tra ciascuna informazione, date di prima e ultima visualizzazione, livelli di confidenza e campi di descrizione. Infatti, attraverso questa piattaforma gli specialisti informatici sono in grado di migliorare le tattiche implementate quando si affrontano minacce legate alla sicurezza informatica.

OpenCTI è in grado di utilizzare il framework MITRE ATT&CK per strutturare i dati e permettendo all'utente di scegliere come implementare i propri dataset. Una volta che i dati sono stati integrati all'interno di OpenCTI, è possibile ricavare nuove relazioni e informazioni per facilitare la comprensione e la rappresentazione dei dati esistenti. OpenCTI consente non solo l'importazione di dati, ma anche l'esportazione di essi in diversi formati (CSV, bundle STIX2, ecc.).

2. Architettura

La piattaforma OpenCTI si basa su diversi database e servizi esterni per funzionare.



L'API è la parte centrale della piattaforma Opencti, permettendo ai client (compreso il frontend) di interagire con il database e il broker del sistema di messaggistica queue oriented. Costruito in NodeJS, implementa il linguaggio di query Graphql.

Il database serve per archiviare informazioni e conoscenze acquisite. I workers sono processi Python autonomi che consumano messaggi dal broker per eseguire query di scrittura asincrona. È possibile avviare tutti i workers di cui si necessita per incrementare le scritture parallele e quindi il throughput. Ovviamente, questo è possibile finché non si raggiungerà il punto di saturazione rappresentato dal throughput del database.

2.1 Connettori

I connettori sono processi Python che inviano dati alla piattaforma in un formato ad essa comprensibile. Consentono di importare, arricchire o esportare facilmente dati sulla piattaforma. Esistono diverse tipologie di connettori in base alle loro funzionalità:

- **EXTERNAL IMPORT**: Recupera informazioni da un'entità o un servizio esterno, le converte in un formato STIX2 e le importa in Opencti.
- **INTERNAL IMPORT FILE**: Estrae dati dai file caricati sulla piattaforma attraverso le API.
- **INTERNAL EXPORT FILE**: Estrae le informazioni memorizzate sulla piattaforma in diversi formati (.csv o .json).
- **INTERNAL ENRICHMENT**: Arricchisce i report o la conoscenza di un oggetto estraendo dati da risorse esterne.
- **STREAM:** Consuma il flusso di dati dalla piattaforma.

Tutti i connettori per poter accedere all'API OpenCTI hanno bisogno di due parametri di configurazione obbligatori, OPENCTI_URL e OPENCTI_TOKEN. Oltre a questi due, i connettori hanno altri parametri che devono essere impostati per farli funzionare. La configurazione può essere fatta attraverso un "*docker-compose.yml*"; al suo interno si devono specificare oltre ai due parametri sopracitati: l'ID del connettore (valore UUIDv4), la sua tipologia, il nome, lo scope e il livello di confidenza.

Esempio:

OPENCTI_URL	http://localhost
OPENCTI_TOKEN	ChangeMe
CONNECTOR_ID	ChangeMe
CONNECTOR_TYPE	Template_Type
CONNECTOR_NAME	Template
CONNECTOR_SCOPE	Template_Scope
CONNECTOR_CONFIDENCE_LEVEL	100
CONNECTOR_LOG_LEVEL	info

2.2 Requisiti Minimi e Configurazione della VM

I requisiti hardware minimi sono:

NUMERO CORE CPU	1
RAM	16 GB
SPAZIO DI ARCHIVIAZIONE	64GB

Il file OVA preconfigurato con i requisiti minimi, importabile in VirtualBox per ospitare il back-end di OpenCTI, è disponibile al seguente link:

https://drive.google.com/drive/folders/1bvB6RmdQNHMW_3h-88KbAit9GRZlL5Bj

Per consentire l'accesso alla piattaforma dai dispositivi connessi in locale è necessario modificare la configurazione della scheda di rete da NAT a BRIDGE. Una volta avviata la VM è necessario effettuare il login fornendo le credenziali.

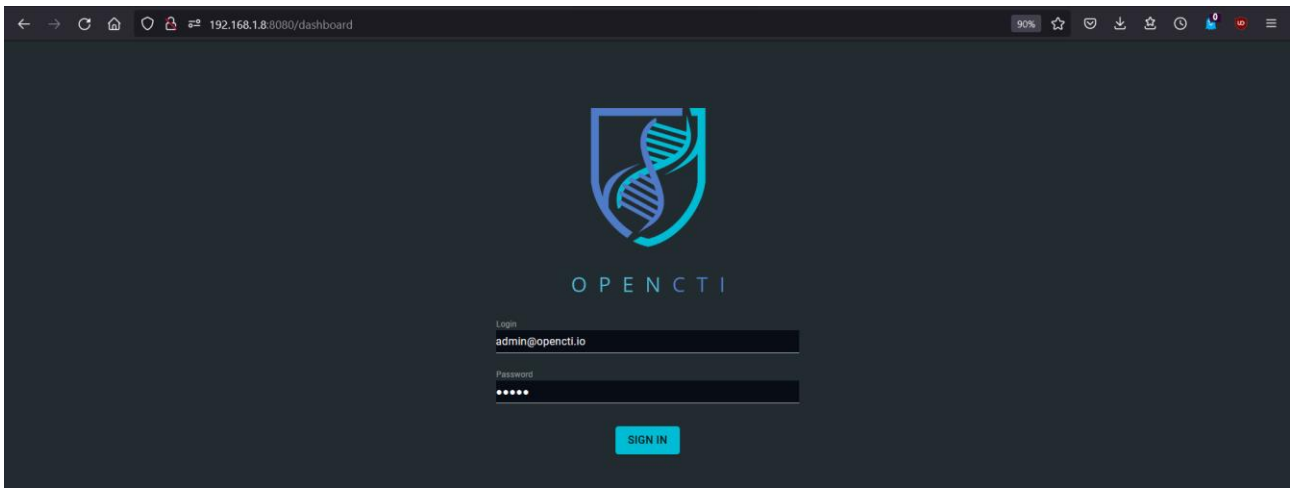
Default Username: opencti

Default Password: opencti

Dopo aver effettuato il login, per accedere alla piattaforma bisogna attendere dai 3 ai 5 minuti. Quest'ultima operazione può essere completata utilizzando l'URL http://{IP_ADDRESS}:8080 e le credenziali:

Username: admin@opencti.io

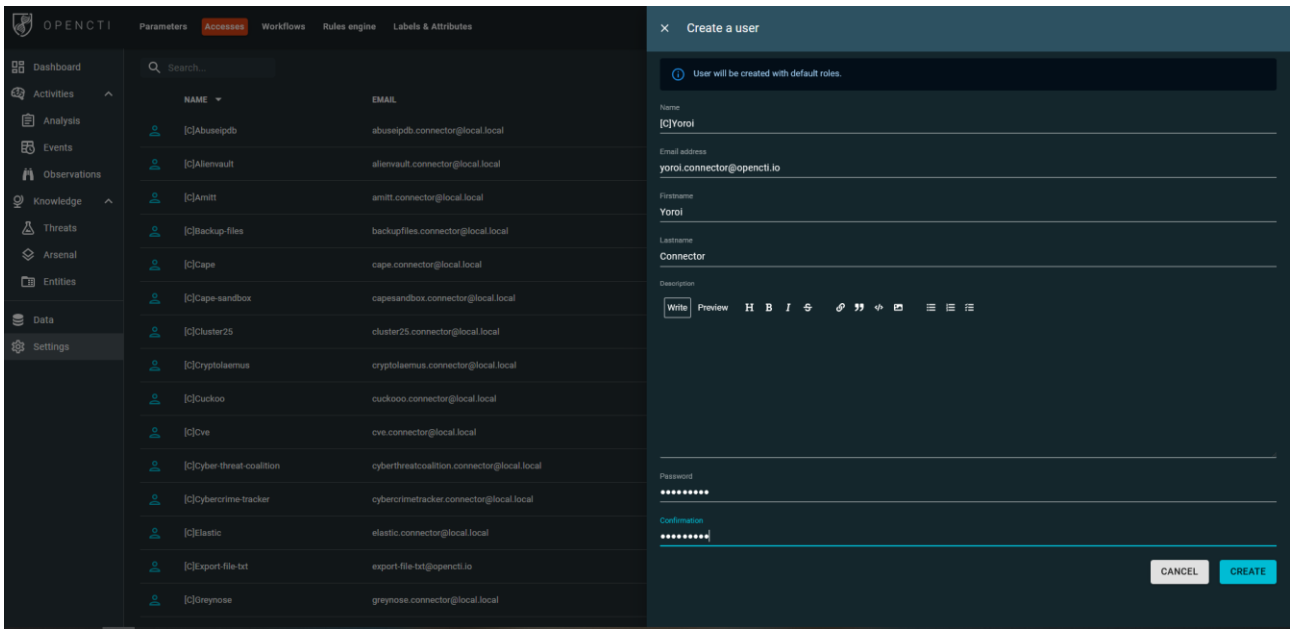
Password: admin



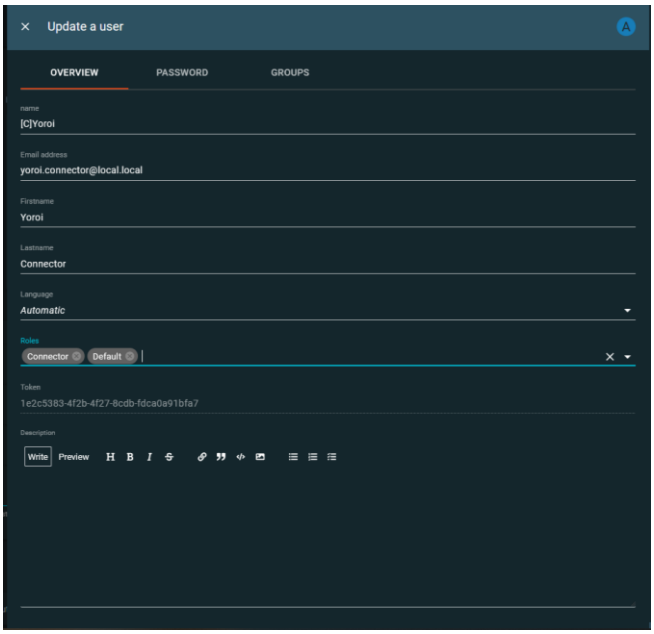
3 Installazione dei connettori

Come già accennato in precedenza per far comunicare i connettori con la piattaforma bisogna settare due parametri obbligatori OPENCTI_URL e OPENCTI_TOKEN. Il primo è settato su http://opencti:8080, invece, il secondo è legato all'utente che bisogna creare per ogni connettore.

Creazione dell'utente: Setting→ Accesses→Users→Create a user



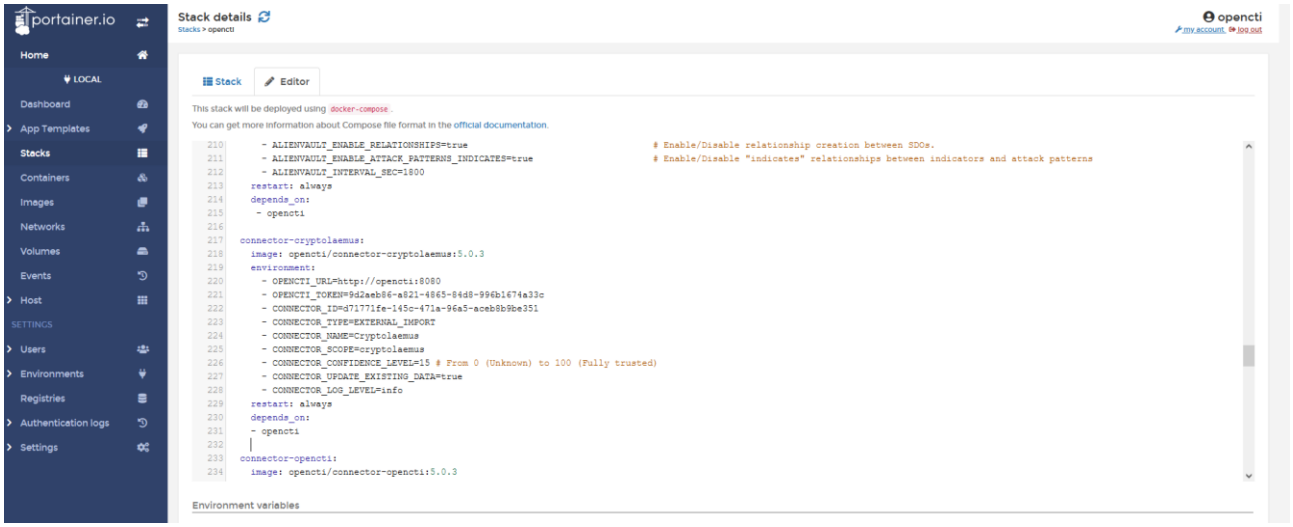
Dopo la creazione dell'utente bisogna assegnargli un ruolo per aver accesso alle informazioni classificate di un gruppo specifico.



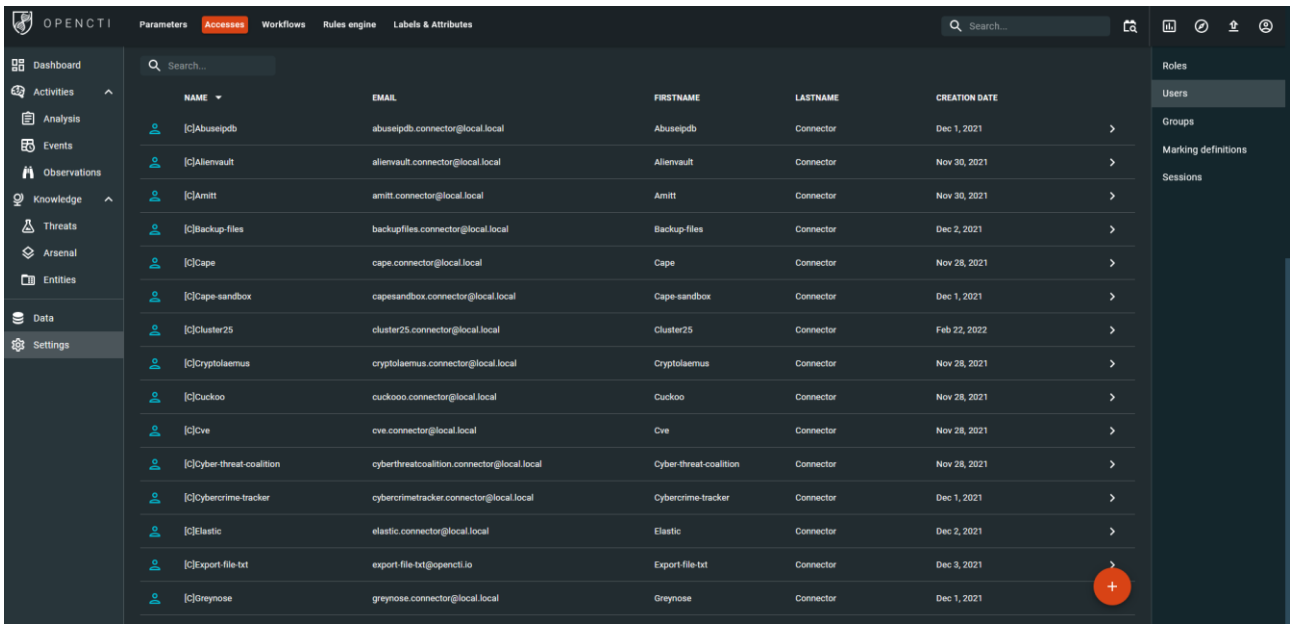
Per ogni connettore installato sono state modificate le configurazioni del docker-compose.yaml. Dove richiesto sono state inserire delle API key legate a ciascun provider.
Il dispiegamento del connettore è stato effettuato usando Portainer.io. A cui è possibile accedere utilizzando lo stesso indirizzo di OpenCTI e la porta 9443.

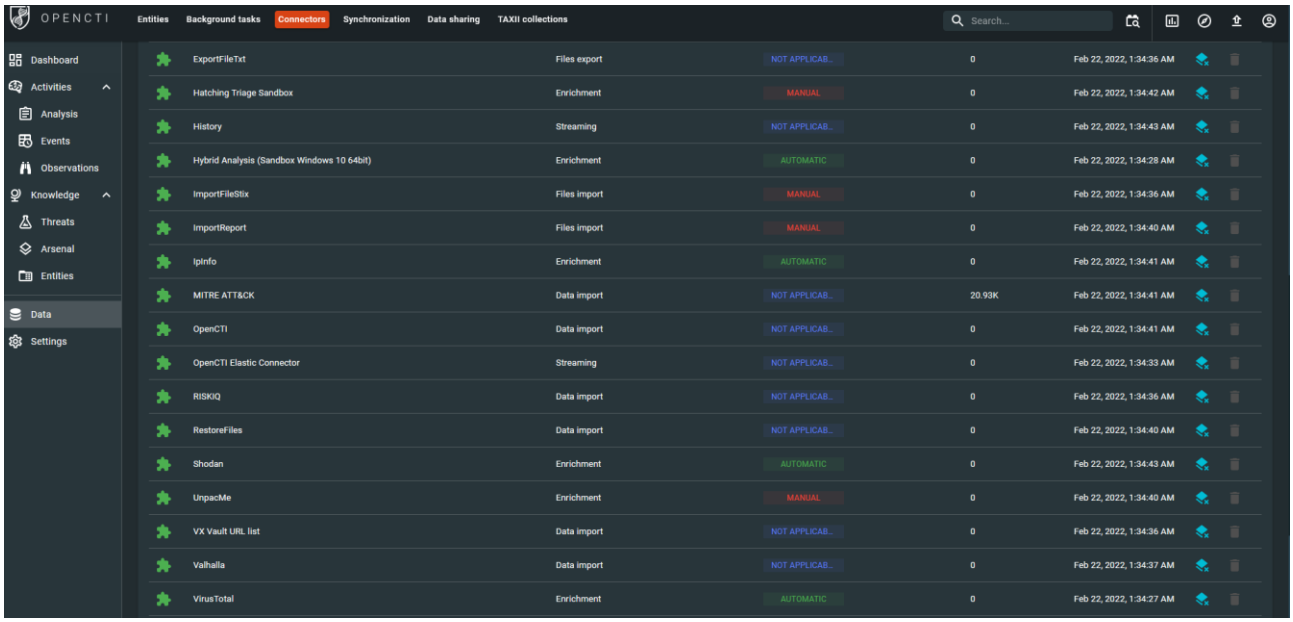
Default username: opencti

Default login: openctiopencti



Le immagini di tutti i connettori sono state recuperate da Docker-HUB.





La maggior parte dei connettori è stata installata con successo (specificati nel dettaglio nell'allegato "connettori.xlsx"). Ma per alcuni sono state riscontrate delle problematiche, legate al rifiuto di concedere una licenza gratuita oppure non si è ricevuta risposta alla richiesta di una demo, li riportiamo di seguito:

- Crowdstrike
- Kaspersky
- Lastinfosec
- Mandiant
- Riskiq
- Sekoia
- Taxii2
- The Hive
- Threatmatch
- Malbeacon
- Splunk

4.Connetori custom

Nel repository GitHub è disponibile la directory Template, contenente tutti i file necessari per sviluppare connettori custom. Tra questi troviamo:

- README.md
- docker-compose.yml
- entrypoint.sh
- Dockerfile
- src/main.py
- src/config.yml.sample

Configurabili opportunamente seguendo le direttive fornite nella documentazione OpenCTI.

4.1 YOROI

https://yoroi.company/blog/

Yoroi è un fornitore di servizi di sicurezza. Fornisce soluzioni di sicurezza informatica contro lo spionaggio industriale, le minacce interne e gli attacchi mirati avanzati. É una delle principali realtà italiana in ambito cybersecurity ed è riconosciuta a livello mondiale per l'eccellenza nel settore.

Ci si è concentrati sul blog di Yoroi, contenente diversi articoli rilevanti e costantemente aggiornato sulle minacce e gli accadimenti di rilievo in ambito cybersecurity.

Per l'esportazione dei vari articoli, in particolare titolo, descrizione, labels significative, si è analizzando il blog ed è stato effettuato lo scraping della pagina web:

```
html_text = requests.get("https://yoroi.company/blog/").text
soup = BeautifulSoup(html_text, 'html.parser')
max_page = int(soup.find_all('a', class_='page-numbers')[-2].text)

for pg in range(1, max_page+1):
    page = requests.get("https://yoroi.company/blog/page/" + str(pg) + "/").text
    soup = BeautifulSoup(page, 'html.parser')

    reports = soup.find_all('div', class_='oxy-post')
    for report in reports:
        labels = []
        title = report.find('h4', class_='yoroi-post__title').text
        tags = report.find('div', class_='yoroi-post__tags')
        for t in tags.find_all('div', class_='yoroi-post__tag'):
            labels.append(t.text.strip())
        date = report.find('p', class_='yoroi-post__date').text
        date = datetime.strptime(date, '%m/%d/%Y').strftime('%Y-%m-%dT%H:%M:%SZ')
        desc = report.find('div', class_='yoroi-post__description yoroi-post__description--collapsed').text
        pre_link = report.find('a', class_='button button--redshadow')
        link = str(pre_link.get('href'))
```

Yoroi è stato sviluppato come connettore di External Import. In particolare, gli oggetti STIX2 creati per questo connettore sono i report, che permettono di raccogliere informazioni sulle minacce incentrate su uno o più argomenti.

Oggetto Report formato STIX2:

Property Name	Type	
type	string	Required
name	string	Required
description	string	Optional
report_types	List of tupe openvocab	Optional
labels	list of type string	Optional
publisced	timestamp	Required
object_refs	list of type identifier	Required
created_by_ref	identifier	Optional

Il file docker-compose.yml contiene la configurazione del connettore:

```yaml
version: '3'
services:
  connector-yoroi:
    image: luigidituccio/opencti-connector-yoroi:latest
    environment:
      - OPENCTI_URL=http://opencti:8080
      - OPENCTI_TOKEN=${CONNECTOR_YOROI_TOKEN}
      - CONNECTOR_ID=${CONNECTOR_YOROI_ID}
      - CONNECTOR_TYPE=EXTERNAL_IMPORT
      - CONNECTOR_NAME=Yoroi
      - CONNECTOR_SCOPE=report # MIME type or Stix Object
      - CONNECTOR_CONFIDENCE_LEVEL=100 # From 0 (Unknown) to 100 (Fully trusted)
      - CONNECTOR_LOG_LEVEL=info
      - YOROI_INTERVAL_SEC=120
    restart: always
```

I requisiti di installazione sono contenuti nel file requirements.txt:

```
pycti==5.1.2
beautifulsoup4==4.10.0
```

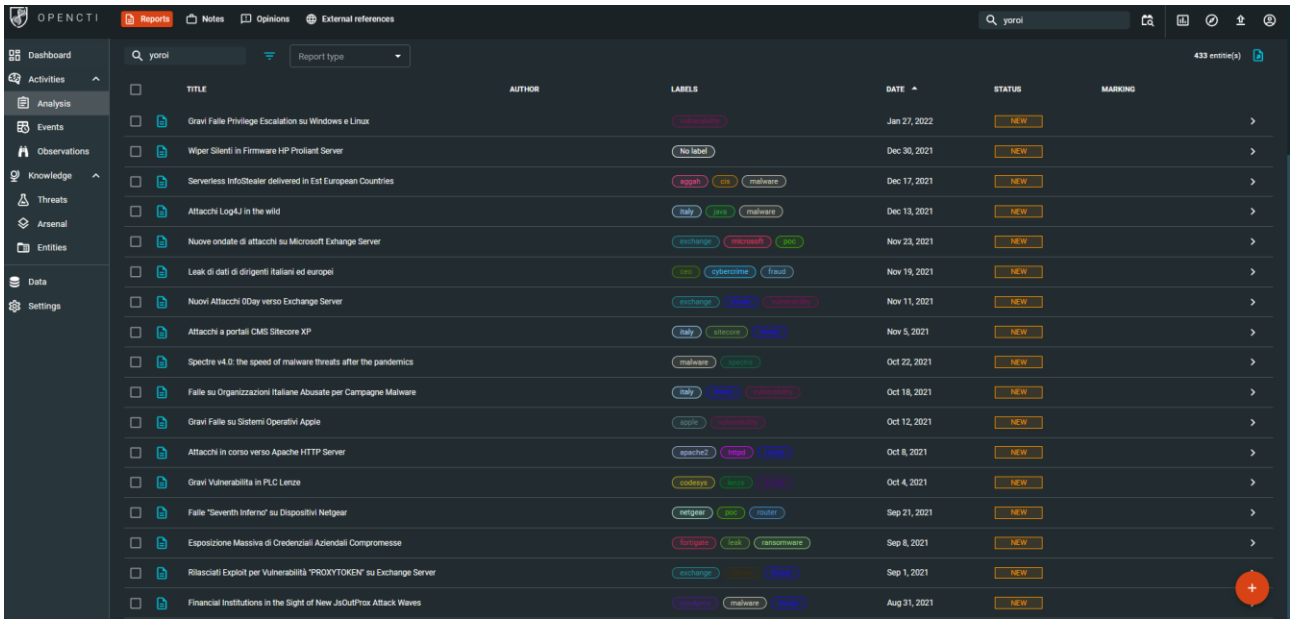
I report generati sono aggiunti ad un Bundle ed inviati ad OpenCTI utilizzando la funzione send_stix2_bundle().

```
bundleObjects.append(Report(type="report",
                            report_types="threat-report",
                            spec_version="2.1",
                            id=OpenCTIStix2Utils.generate_random_stix_id("report"),
                            created=date,
                            name=title,
                            modified=date,
                            description=desc,
                            object_refs=[self.author],
                            labels=labels,
                            published=date,
                            external_references=[er],
                            ))
```

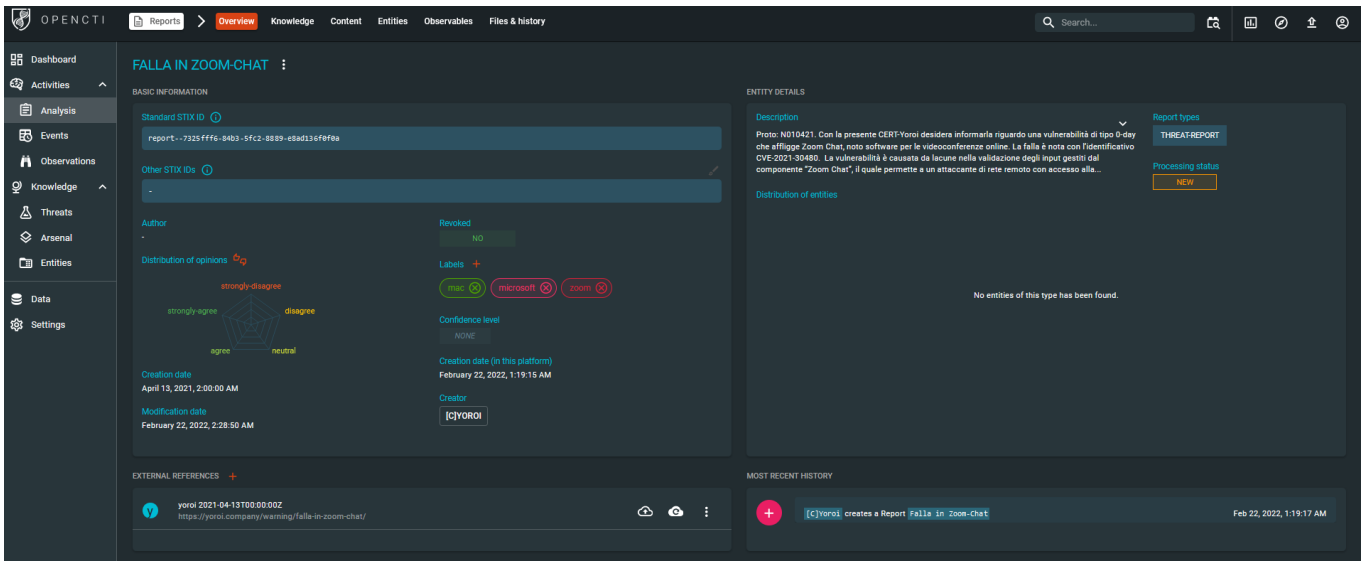
Dopo aver recuperato i vari articoli del blog Yoroi, si estrae da ciascuno il relativo URL e viene inserito come attributo dell'External Reference immessa all'interno dell'apposito Report da inviare ad OpenCTI.

```
er = ExternalReference(
    source_name="yoroi " + date, url=link
)
```

É possibile osservare i report importati sulla piattaforma:



Struttura di un report Yoroi in dettaglio:



4.2 Cluster25

https://cluster25.io/

Cluster25 è la divisione di ricerca interna sulla sicurezza informatica di un'azienda tecnologica mondiale. Cluster25 è specializzata nella caccia e nella raccolta di minacce informatiche, analisi e processi di reverse engineering. Infatti, in Cluster25 vengono sviluppate internamente tecnologie e strumenti per le pratiche di attribuzione, classificazione e categorizzazione di artefatti dannosi spesso prima che questi vengano utilizzati negli interventi informatici.

L'obiettivo di Cluster25 è quello di analizzare, contestualizzare e convalidare i dati di cyber threat intelligence provenienti da diverse fonti (OSINT, CLOSINT, terze parti commerciali, partner, programmi di condivisione delle informazioni ecc.ecc.) e produrre report dettagliati.

Analizzando il sito si evince che è principalmente un blog, contenente diversi articoli rilevanti sulle minacce e gli accadimenti di rilievo in ambito cybersecurity.

Per l'esportazione dei vari articoli, in particolare titolo, descrizione, labels significative, si è analizzando il blog ed è stato effettuato lo scraping della pagina web:

```
html_text = requests.get("https://cluster25.io/blog/").text
soup = BeautifulSoup(html_text, 'html.parser')
reports = soup.find_all('div', class_='firwl-post__content')
for report in reports:
    labels = []
    pre_link = report.find('h3', class_='firwl-post__title')
    title = pre_link.text
    pdf_link = pre_link.find('a').get('href')
    tg_list = self.get_tags(pdf_link)
    if pdf_link is not None:
        pdf_link = self.get_pdf_link(pdf_link)
    tags = report.find('span', class_='firwl-p-catz')
    if tags:
        for t in tags.find_all('a'):
            labels.append(t.text.strip().replace("\t", "").replace("\n", "").replace("+", ""))

    tag_list = tg_list + labels
    date = report.find('a', class_='firwl-p-auth').text.replace("/", "").strip()
    date = datetime.strptime(date, '%B %d, %Y').strftime('%Y-%m-%dT%H:%M:%SZ')
    desc = report.find('div', class_='firwl-excerpt').text.strip()
```

```
def get_pdf_link(self, link):
    html_text = requests.get(link).text
    soup = BeautifulSoup(html_text, 'html.parser')
    pdf = soup.find('div',
                    class_='elementor-element elementor-element-48deb1c elementor-widget elementor-widget-image')
    if pdf:
        pdf = pdf.find('a').get('href')
    else:
        pdf = "Not found"
    return pdf
```

```python
def get_tags(self, link):
    html_text = requests.get(link).text
    soup = BeautifulSoup(html_text, 'html.parser')
    tags = soup.find('p', class_='firwl-tags')
    tag_list = []
    if tags is not None:
        for t in tags:
            tag_list.append(t.text.replace('Tagged as:', "").replace(',',"").replace('.',"").replace('#',"").strip())
    else:
        tags= 'No tags found'
    tag_list = [value for value in tag_list if value != ""]
    return tag_list
```

Cluster25 è stato sviluppato come connettore di External Import. In particolare, gli oggetti STIX2 creati per questo connettore sono i report, che permettono di raccogliere informazioni sulle minacce incentrate su uno o più argomenti.

Oggetto Report formato STIX2:

Property Name	Type	
type	string	Required
name	string	Required
description	string	Optional
report_types	List of tupe openvocab	Optional
labels	list of type string	Optional
publisced	timestamp	Required
object_refs	list of type identifier	Required
created_by_ref	identifier	Optional

Il file docker-compose.yml contiene la configurazione del connettore:

```yaml
version: '3'
services:
  connector-cluster25:
    image: luigidituccio/opencti-connector-cluster25:latest
    environment:
      - OPENCTI_URL=http://opencti:8080
      - OPENCTI_TOKEN=${CONNECTOR_CLUSTER25_TOKEN}
      - CONNECTOR_ID=${CONNECTOR_CLUSTER25_ID}
      - CONNECTOR_TYPE=EXTERNAL_IMPORT
      - CONNECTOR_NAME=Cluster25
      - CONNECTOR_SCOPE=report # MIME type or Stix Object
      - CONNECTOR_CONFIDENCE_LEVEL=100 # From 0 (Unknown) to 100 (Fully trusted)
      - CONNECTOR_LOG_LEVEL=info
      - CLUSTER25_INTERVAL_SEC=120
    restart: always
```

I requisiti di installazione sono contenuti nel file requirements.txt:

```
pycti==5.1.2
beautifulsoup4==4.10.0
```

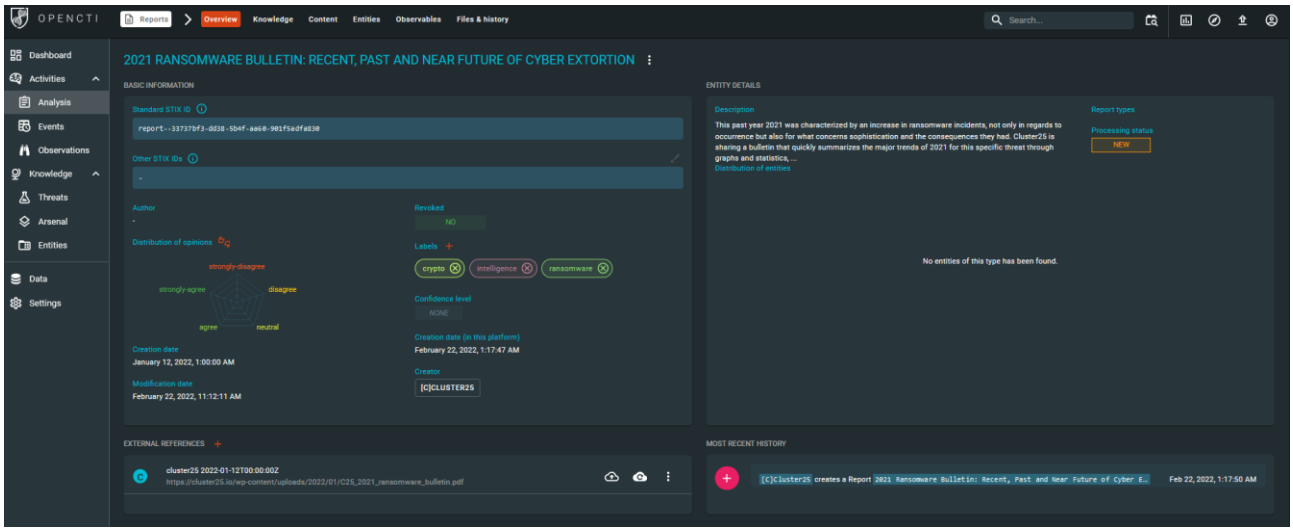
I report generati sono aggiunti ad un Bundle ed inviati ad OpenCTI utilizzando la funzione send_stix2_bundle().

```
bundleObjects.append(Report(type="report",
                            spec_version="2.1",
                            id=OpenCTIStix2Utils.generate_random_stix_id("report"),
                            created=date,
                            name=title,
                            description=desc,
                            object_refs=self.author,
                            external_references=[er],
                            published=date,
                            labels=tag_list
                            ))
```

Dopo aver recuperato i vari articoli del blog Cluster25, si estrae da ciascuno il relativo URL e viene inserito come attributo dell'External Reference immessa all'interno dell'apposito Report da inviare ad OpenCTI.

```
er = ExternalReference(
    source_name="cluster25 " + date,
    url=pdf_link
)
```

Struttura di un report di Cluster25 in dettaglio:



Le immagini docker dei due connettori sono disponibili su Docker-Hub al repository:

https://hub.docker.com/repository/docker/luigidituccio/opencti-connector-yoroi

https://hub.docker.com/repository/docker/luigidituccio/opencti-connector-cluster25

5. Export dei dati

È possibile effettuare l'export delle informazioni (osservabili, minacce, eventi, etc.) nei seguenti formati:

- Json
- Csv