

POLITECNICO DI TORINO

POLITECNICO DI TORINO

Master Degree course in Cybersecurity Engineering

Master Degree Thesis

The Cyber Threat Intelligence Cycle from the Collection Perspective

Supervisor
Prof. Fulvio Valenza

Candidate
Enrico Di Stasio

Academic Year 2025-2026

Abstract

Cyber Threat Intelligence (CTI) has become a core function within organizational security operations, structured around a cyclical process of direction, collection, processing, analysis, dissemination, and feedback.

This thesis examines the CTI Cycle as applied within Leonardo, a leading company in Italy's cybersecurity sector, using as a case study the design and implementation of a data feeder for CTIS, the company's existing CTI platform, using as source the European Repository of Cyber Incidents (EuRepoC), an open-access, multi-variable database of politically and technically significant cyber incidents. The study addresses the need for adding new, already curated, incidents from OSINT sources to CTIS.

Adopting a mixed-methods approach, the thesis describes the requirements of the EuRepoC coding methodology and details the design, implementation, and evaluation of a feeder pipeline that collects, normalizes, and pre-processes incident data from the EuRepoC's database and adapts it to the standard already used by the company's platform (STIX 2.1) as well as the processes in place for analysing and disseminating the obtained information.

The results show that besides the obvious automation of the incidents' inclusion, there is also an enrichment of already existing ones. In particular the feeder allowed approximately 5000 incidents to be added to the platform as well as new ones being added everyday.

Keywords: Cyber Threat Intelligence, CTI Cycle, EuRepoC, cyber incident database, threat data feeds, open-source intelligence (OSINT)

Contents

1 Introduction 9
- 1.1 Context . . . 9
- 1.2 Problem Statement . . . 10
- 1.3 Significance and Generalizability of the Feeder Architecture . . . 10
- 1.4 Thesis Structure . . . 11

2 System model 13
- 2.1 EuRepoC Project . . . 13
  - 2.1.1 APT Profiles . . . 16
- 2.2 The CTI Cycle . . . 17
  - 2.2.1 Planning and Direction . . . 17
  - 2.2.2 Collection . . . 18
  - 2.2.3 Processing and Exploitation . . . 19
  - 2.2.4 Analysis and Production . . . 19
  - 2.2.5 Dissemination . . . 20
- 2.3 The Diamond Model of Intrusion Analysis . . . 20
  - 2.3.1 Introduction . . . 20
  - 2.3.2 The Event as the Atomic Unit of Analysis . . . 21
  - 2.3.3 Core Features . . . 22
  - 2.3.4 Meta-Features . . . 24
  - 2.3.5 From Events to Activity Threads, Campaigns, and Activity Groups 27
  - 2.3.6 Analytic Pivoting . . . 28
  - 2.3.7 Applying the Diamond Model to the EuRepoC Data . . . 30
- 2.4 The STIX Standard . . . 30
  - 2.4.1 STIX Relationships . . . 33
  - 2.4.2 STIX Patterning . . . 34
  - 2.4.3 Common Data Types . . . 35
  - 2.4.4 General Concepts . . . 36
  - 2.4.5 STIX Domain Objects . . . 38
  - 2.4.6 STIX Meta Objects . . . 40
  - 2.4.7 STIX Bundle Object . . . 41
  - 2.4.8 STIX Vocabularies . . . 41

3 Leonardo's Cyber Threat Intelligence Platform (CTIS / C-THINK) 45
3.1 Overview and positioning . . . 45
3.2 Architecture: the CTI lifecycle . . . 46
3.3 Analytical frameworks and domain specialization . . . 47
3.4 Expansions to the STIX Standard . . . 47
3.4.1 Operation . . . 47
3.4.2 News . . . 47
3.5 Integration with LENS (EDR) . . . 48
3.6 Deployment models . . . 48
3.7 Vendor-Listed Capabilities . . . 48

4 The Feeder Description 51
4.1 Basic Information . . . 51
4.2 Technologies Involved . . . 51
4.2.1 Python . . . 51
4.2.2 Docker . . . 52
4.2.3 NocoDB . . . 53
4.3 The Feeder Structure . . . 53
4.4 Feeder Execution Flow . . . 55
4.4.1 Dataset Download . . . 55
4.4.2 Dataset Pre-processing . . . 55
4.4.3 Entities Extraction . . . 56
4.4.4 Global Mappings and Sentinel Values . . . 56
4.4.5 CTIS Entities Definition . . . 57

5 Feeder evaluation 71
5.1 Evaluation Methodology . . . 71
5.2 Numerical results . . . 72
5.2.1 Targeted Sectors . . . 72
5.2.2 Conflicts . . . 74
5.2.3 Cyber Conflicts . . . 75
5.2.4 EuRepoC Types . . . 76
5.2.5 Technical Impact Labels . . . 77
5.2.6 Legal and Political Labels . . . 78
5.2.7 Intrusion Set Labels . . . 79

6 Conclusion 87

Bibliography 89

Summary

Cyber Threat Intelligence (CTI) has become a core function within organizational security operations, structured around a cyclical process called the **CTI Cycle**. It adapts the traditional intelligence cycle used in military and national security contexts to the domain of cybersecurity. Intelligence, in this context, refers to the process of collecting, processing, and analyzing information about a competitive or hostile entity and its agents, in order to support an organization's or group's security and overall well-being. One of the most important phases is **Collection**, which involves researching sources of intelligence and collecting that intelligence to add it to one's knowledge base, so that it can later be used in the other steps of the cycle. Despite its centrality to the cycle, the Collection phase is often a manual bottleneck: analysts must repeatedly identify, evaluate, and reconcile new external data sources against the specific data model used by their own **Threat Intelligence Platform (TIP)**, a process that does not scale well as the number and diversity of available OSINT sources continues to grow. This thesis examines the Collection phase from the point of view of a CTI Analyst on a company's CTI team, tasked with integrating a new feeder for a TIP, in order to address this gap. The objective was to automate the collection of cyber operations from the **European Repository of Cyber Incidents (EuRepoC)**, a comprehensive, interdisciplinary, and continuously updated database of cyber incidents worldwide, consisting of around 5000 incidents dating from 2000 to today. The idea behind the feeder was to map information contained in the EuRepoC database onto the **STIX 2.1 standard**, the framework used to construct entities within CTIS (or C-THINK), the TIP of Leonardo, the company where I carried out my internship. CTIS is Leonardo's next-generation platform, designed to support analyst teams operating in high-confidentiality environments, such as national CERTs/CSIRTs, defense organizations, and critical infrastructure operators, and is explicitly built around the classical Cyber Threat Intelligence lifecycle. STIX, in turn, is an open standard language and serialization format used to exchange cyber threat intelligence. It enables the clear representation of all aspects of suspicion, compromise, and attribution through a set of objects and the descriptive relationships that connect them. The challenge of mapping EuRepoC to STIX was tackled by drawing on several structured frameworks, including the **Cyber Kill Chain** and **MITRE ATT&CK**, with the most important being the **Diamond Model**. The **Diamond Model of Intrusion Analysis**, introduced by Caltagirone, Pendergast, and Betz, is one of the most widely adopted conceptual tools for organizing and reasoning about cyber intrusions. Unlike purely descriptive taxonomies, the Diamond Model provides both a formal representation of malicious activity and a methodology for exploiting that representation

analytically. At its core, the model rests on a single claim: every malicious act, or event, can be described as an **adversary** deploying a **capability** over some **infrastructure** against a **victim**. These four elements, the model's **core features**, are represented as the vertices of a diamond, with the edges between them capturing the natural relationships that link adversary operations together. On the technical side, the feeder was built as a **Python** application running in a **Dockerized** environment. The choice of language was motivated by the existing **CTIS API Library**, which facilitated queries to the TIP, while Docker was used to improve environment isolation and ensure the application's reproducibility across hardware migrations. The feeder was designed to run every day to retrieve new cyber incidents, with scheduling handled via **Crontab**. Since access to EuRepoC's dedicated API could not be granted by the consortium, the feeder instead performs a daily scrape of EuRepoC's table view of the database. This constraint led to some difficulties in parsing the downloaded .xlsx dataset, which were addressed by writing dedicated methods to decouple the various attributes composing an incident. A single EuRepoC incident cannot be represented as one STIX object, since it simultaneously carries information about the operation itself, the actors involved, the parties affected, and the broader legal and political response it triggered, each of which corresponds to a conceptually distinct STIX entity type. The feeder therefore decomposes every incident into several interrelated entities, mirroring the multi-object structure of STIX and allowing each dimension of an incident to be queried and enriched independently on CTIS. The main entities obtained by mapping each EuRepoC incident were:

- **Operation**, the entity used to track the incident, with most attributes mapped as labels. It also serves as the starting point for relationships with the other main entities
- **News**, used to mirror the **Operation** entity, with some differences in terms of labels
- **Attribution**, which tracks who attributed the incident to whom; mapped as an entity of type News due to certain restrictions of the CTIS platform
- **Initiator**, which tracks the actual **Intrusion Set** or **Threat Actor** responsible for the attack. It is mapped to an existing CTIS entity if one is found on the platform; otherwise, it is added to a **NocoDB** table for manual mapping by an analyst later
- **Receiver**, the victim of the incident, tracked both as an **Identity** and as a **Sector**
- **Legal and Political Response**, the response to the incident carried out by states or organizations
- **Attack Patterns**, the techniques used by the Initiators, as described by MITRE ATT&CK

At the time of writing, approximately 5000 Operations had been ingested, with new ones being added every day. The feeder's evaluation focused not on functional correctness, but on the substantive content it extracts from the EuRepoC dataset, examining what the resulting distributions reveal about both the dataset and the feeder's mapping

logic. This was done by aggregating feeder output into frequency dictionaries across several categories of labels, allowing patterns to emerge across the entire processed dataset. Across the label categories examined, i.e. **Sectors Targeted**, **Conflicts**, **Cyber Conflict Issues**, **EuRepoC Types**, **Technical Impact Labels**, **Legal and Political Labels**, and **Intrusion Set Labels**, a consistent interpretive theme emerges: the observed distributions are best understood as artifacts of EuRepoC's coding methodology and inclusion criteria, rather than as unmediated reflections of the global cyber threat landscape. The dominance of **National Government** as a targeted sector, for instance, follows directly from EuRepoC's requirement that included incidents possess a demonstrable political dimension, while the prominence of **Russia-Ukraine** among **Conflict** labels reflects the scale of the underlying offline conflict rather than a standalone measure of cyber activity. Similarly, the prevalence of labels such as **Hijacking with misuse** or **Sovereignty** among **International Law Breach** labels results from the structural and evidentiary logic underpinning EuRepoC's coding scheme, rather than indicating a dominant attacker motive or legal reality independent of that scheme. In general, a consistent concept demonstrated by the findings is that the feeder's output cannot be interpreted in isolation from the coding methodology and platform context that shape it. With this caveat in mind, the evaluation concludes that the feeder's output faithfully represents the dataset it was built to process. Grounded in the CTI Cycle and the Diamond Model, the feeder addresses the Collection phase of threat intelligence from the perspective of an analyst seeking to identify and integrate new OSINT sources. The feeder met this objective, both by demonstrating robust technical translation between formats and by enabling richer qualitative analysis of EuRepoC's incidents, Intrusion Sets, and targeted Sectors than either the source dataset or the existing TIP could offer on its own. It does, however, have some limitations. The most significant of these is its dependence on EuRepoC's public-facing table view rather than a dedicated API, which leaves the preprocessing pipeline brittle to formatting changes and introduces data loss through cell truncation; resolving this by obtaining API access from the EuRepoC consortium would therefore be the natural priority for any continuation of this work, as it would remove the root cause of several of the feeder's remaining limitations at once. The underlying mapping logic, by contrast, is general enough to extend to other sources, and is not affected by this constraint. Beyond securing API access, further directions for future work include enriching the CTIS knowledge base with additional OSINT sources, and generalizing the mapping logic into a reusable framework for onboarding future feeds. Automating the distinction between Intrusion Set and Threat Actor entities is also a particularly promising but difficult challenge, one likely to require continued analyst oversight even with the aid of **NLP techniques**. Ultimately, this work demonstrates that OSINT sources like EuRepoC can be reliably integrated into operational TIPs, and that this thesis offers not just a working implementation, but a reusable methodology for scaling the Collection phase of the CTI Cycle in the face of growing threat data volume and diversity. In doing so, it turns Collection from the largely manual bottleneck described at the outset into a more scalable, semi-automated process, better equipped to keep pace with the CTI landscape it is meant to serve.

Chapter 1

Introduction

1.1 Context

Cyber incidents affecting states, critical infrastructure, and private organizations have grown markedly in frequency, sophistication, and political significance over the past decade. What were once treated as isolated technical failures are now routinely framed as instruments of geopolitical competition, coercion, and even armed conflict, as seen in the cyber operations accompanying the war in **Ukraine**, ransomware campaigns against hospitals and pipelines, and state-sponsored espionage targeting critical infrastructure. This shift has elevated cyber incidents from a purely technical concern to an object of foreign policy, national security strategy, and international law.

In response, organizations have increasingly formalized **Cyber Threat Intelligence (CTI)** functions to anticipate, detect, and respond to these threats. CTI practice is commonly structured around the so-called '**Intelligence Cycle**': a recurring process of **Direction**, **Collection**, **Processing**, **Analysis**, **Dissemination**, and **Feedback**, adapted from traditional intelligence studies.

A persistent challenge for CTI is the fragmentation and inconsistency of data on cyber incidents. Attribution claims, severity assessments, and legal-political classifications vary widely across vendors, media reporting, and government disclosures, complicating efforts to build cumulative, comparable evidence about the cyber threat landscape. In response, several research consortia have developed open, structured repositories of cyber incidents to support evidence-based policymaking and academic research. One such initiative is the **European Repository of Cyber Incidents (EuRepoC)**, an independent research consortium that codes cyber incidents against a **standardized**, **multi-disciplinary** codebook covering **technical**, **legal**, and **political** dimensions. By combining open-source data collection with structured, expert-driven coding, EuRepoC and comparable initiatives aim to bridge the gap between the granular, often proprietary data produced within the CTI intelligence cycle and the aggregated, comparable evidence base needed for policy analysis and academic research.

1.2 Problem Statement

This project originated from **Leonardo's CTI team**'s need for new OSINT sources on cyber operations to enrich the incidents already tracked on their **Threat Intelligence Platform (TIP) CTIS**. EuRepoC was selected for its methodology in selecting the operations it covers, as well as the depth and quality of the information the consortium tracks. In particular, EuRepoC tracks both **political** and **legal responses** to cyber incidents, shifting the focus from the purely technical aspects of cyber operations to their broader geopolitical implications.

The main advantage of this feeder lies in the automation of the **Collection** phase of the **CTI Cycle**, reducing the manual effort typically required to gather and structure relevant threat intelligence.

The main challenges encountered during implementation stemmed primarily from two areas. The first was the *translation* of the information provided by EuRepoC into **STIX entities** and the corresponding **relationships** between them, a process that required carefully mapping heterogeneous, human-readable data into a structured and semantically consistent format.

The second challenge concerned obtaining the EuRepoC information directly. Although the consortium provides an **API** for systematically querying their database, I was not able to obtain access to it. As a result, the required information had to be retrieved through scraping of the *table view* available on their website, which presented additional difficulties due to its underlying structure. These challenges will be explored in more detail later in this work.

1.3 Significance and Generalizability of the Feeder Architecture

Although the feeder was designed specifically to ingest and structure information originating from the EuRepoC repository, its underlying architecture is not inherently tied to that data source. The central design effort of this work is not the ingestion logic itself, but the **pre-planning** phase: the deliberate definition of a **coherent entity model and the relationships connecting those entities**. This modelling decision, rather than any EuRepoC-specific parsing logic, constitutes the core intellectual contribution of the feeder component.

By decoupling the conceptual data model from the specifics of any single source, the feeder establishes a template that can, in principle, be reused or adapted for other cyber incident datasets and research repositories beyond EuRepoC. A future project seeking to integrate a different data source into the same CTI platform would not need to redesign the underlying entity-relationship structure from scratch; it would instead need to implement a source-specific adapter that maps new data onto the existing schema.

This has two practical implications. First, it supports the long-term maintainability of the CTI platform: as new sources of cyber incident data emerge or as EuRepoC's own codebook evolves, the platform can absorb this information without fragmenting its internal structure or requiring parallel, incompatible data models for each source.

Second, it strengthens the platform's analytical value, since a coherent and consistent entity-relationship structure is a precondition for meaningful cross-source analysis like, for example, correlating actors, incidents, and techniques reported by EuRepoC with equivalent information from other repositories or threat intelligence feeds.

More broadly, this contribution reflects a recurring challenge in operationalizing cyber threat intelligence: the value of any given data source is constrained not only by its content, but by how well that content can be structured, related, and integrated into a broader analytical environment. By treating schema design as a first-class problem rather than an incidental byproduct of ingesting EuRepoC data, this work aims to provide a foundation that outlives the specific dataset it was built to support.

1.4 Thesis Structure

This thesis is structured to move from conceptual foundations, through case-study context, to design, implementation, and evaluation, before returning to the broader CTI Cycle framework in its conclusion.

Chapter 2 reviews the relevant literature, covering the **CTI Cycle** as a conceptual framework for structuring intelligence work, the **Diamond Model** to help with the correct mapping of EuRepoC data onto CTIS, the **STIX** standard for representing and exchanging structured threat intelligence, and the landscape of open cyber incident repositories, with particular attention to **EuRepoC**'s data collection and coding methodology. Together, these four strands establish the theoretical and technical vocabulary used throughout the rest of the thesis.

Chapter 3 introduces the case study context in which the feeder was developed, describing Leonardo's existing CTI platform (**CTIS**) and the operational processes it supports. This chapter grounds the thesis in an organizational setting that clarifies the practical constraints and requirements that shaped the feeder's design.

Chapter 4 details the design and implementation of the feeder itself, including the entity-relationship model developed to structure incoming EuRepoC data and the engineering choices made to integrate this data coherently into CTIS.

Chapter 5 presents the evaluation of the feeder based on its use in production, assessing its performance and its practical impact on CTI workflows.

Chapter 6 discusses these findings in light of the CTI Cycle framework introduced in Chapter 2, reflecting on how the feeder's contribution maps onto stages of the intelligence process. The chapter concludes with a discussion of the work's limitations and recommendations for future research.

Chapter 2

System model

In this chapter, the topics related to the thesis will be described in depth. In particular, the chapter is organized into the following sections:

- **The EuRepoC project and its methodology for collecting information**, an overview of the project's goals, data sources, and the methodology used to gather and structure cyber incident data.
- **The theory behind the Cyber Threat Intelligence (CTI) Cycle**: a description of the CTI lifecycle, its phases, and its relevance to threat intelligence analysis.
- **The Diamond Model**: a theoretical overview of the **four core features** of the model as well as the its **Meta-Features**.
- **The STIX standard**: an introduction to the Structured Threat Information eXpression (STIX) standard, its data model, and its role in enabling structured threat information sharing.

2.1 EuRepoC Project

The **Eu**ropean **Repo**sitory of **C**yber Incidents (**EuRepoC**) provides a comprehensive, inter-disciplinary and continuously updated database of cyber incidents worldwide:

- Covering incidents from 2000 up to today
- Around 3,500 articles from 221 sources scanned and curated daily
- Each cyber incident is coded across 60 variables

Data is gathered, processed and analysed from public reports of cyber incidents through a combination of data science tools and expert knowledge. The Data Pipeline is composed of:

1. **Data Collection**: Data is collected daily from over 200 sources, including IT reports, government reports, media articles and social media content.

2. **Automated Pre-processing**: The collected data is filtered using a machine-learning model trained to detect reports of cyber incidents.
3. **Inter-disciplinary coding**: Researchers code each reported cyber incident using an online interface along three dimensions: technical, political and legal.
4. **Updating**: All cyber incidents are reopened and recoded as new information emerges.

with its output feeding into:

- **Database**: All coded cyber incidents are saved in a PostgreSQL database.
- **Dashboard**: The interactive dashboard queries the database and is refreshed daily.
- **Research papers**: The data is analysed by researchers to identify key trends in cyber conflicts.

The scientific coding of cyber incidents is an ongoing, iterative process which necessitates continuous refinement and recalibration. As new data is obtained, it is continuously incorporated into the existing dataset, which in turn updates the interactive dashboard.

A standardized codebook [2] is used to assess data on cyber incidents worldwide against a set of 60 criteria. For each incident, the main categories of information recorded are:

- **ID**, **name**, and **description** of the incident
- **Start** and **end date**
- **Inclusion criteria**
- **Source of incident detection/disclosure**: whether the disclosure of the incident originates from the victim, from a self-attribution by the actor or from a third party
- **Incident type**: the category of the incident (disruption, data theft, etc.)
- Information regarding the **receiver(s)**, including: name, country, region, and category (sector)
- Information regarding the **initiator(s)** of the incident, including: name, country, and category
- Information regarding the **attribution(s)**, including: attribution ID, date, type, and the basis for the attribution
- Information regarding the **attributing actor(s)**, including: name, IT company (if available), and country
- Information regarding the **attributed initiator(s)**, including: name, country, and category

- **Cyber conflict issue**, **offline conflict issue**, and its **intensity** according to the intensity levels of the **Conflict Barometer by the Heidelberg Institute for International Conflict Research (HIIK)** [3]. Each incident is coded for its relevance to broader cyber conflict issues and, where applicable, associated offline (kinetic or diplomatic) conflict issues, situating cyber incidents within their geopolitical context.
- **Political response(s)** to the incident, including: date, type, country, and actor
- Information on whether the incident exploited a **zero-day vulnerability**, **required user interaction**, and which **techniques of the MITRE initial access and MITRE impact tactics** it can be mapped to
- Whether the incident caused **disruption**, **data theft**, and/or **hijacking**
- **Spatial and temporal physical effects**, which, together with the **target multiplier** of the receiver, form the **cyber intensity score**
- Indicators of the **impact of the incident**, including: **functional**, **intelligence**, **political** (based on affected countries), and **economic** impacts
- **State responsibility indicator**: the degree of involvement of a state
- The area of **international law (IL) breached** (if any) and the **evidence used to justify sanctions** (if present)
- **Legal response(s)** to the incident, including: date, type, country, actor
- **Number of casualties** (if applicable)
- Sources for all the above information

The **Scope** mandates that only cyber incidents fitting the following criteria are included in the database:

- Cyber incidents in violation of the **CIA** triad of information security (confidentiality, integrity, availability).
- Cyber incidents that have been **publicly reported**; this excludes a significant number of unreported cases due to non-detection or non-disclosure.
- Cyber incidents with a **political dimension**. In practice, this includes cyber incidents that:
  - have affected political or state actors/institutions;
  - have been associated with state actors as the actual "mastermind," or exhibit a political motivation;
  - have been publicly politicized, regardless of the affected target.

- Cyber incidents **against critical infrastructure**, regardless of the attributed initiator, due to the elevated threat they pose.

The **Cyber Intensity Indicator** measures the severity of a cyber incident by evaluating the duration of the incident itself, the duration of its effects, and the criticality of the targets involved.

2.1.1 APT Profiles

The EuRepoC database presents the most prominent **Advanced Persistent Threat (APT)** groups in the form of short, standardized and regularly updated profiles, which cover:

- key aspects of the group's conflict activity (quantitative & qualitative);
- its characterization within the framework of political, technical and legal attribution processes;
- countermeasures already initiated against it.

The **Threat Level Index** assesses the overall intensity, frequency, and scope of the attacks carried out by specific APT groups. The index is a composite indicator compiling **five individual sub-indicators** into a single score. Each sub-indicator is calculated using data from the EuRepoC database of cyber incidents; only attacks that have been publicly attributed to the APT group during its period of activity, and which meet the specific EuRepoC criteria for inclusion, are considered. The five sub-indicators are:

1. **Intensity of the APT's attacks**: represents the average Weighted Cyber Intensity score from the EuRepoC codebook for all attacks attributed to the APT during its period of activity. The average is then converted into a five-level scale.
2. **Sectorial scope of the APT's attacks**: this indicator is calculated in two steps:
   (a) First, the average number of unique targeted sectors per attack attributed to the APT is calculated. This average is then converted to a four-level scale.
   (b) Multipliers are introduced to account for the broader societal impact of the APT's targets, weighing targets critical to the functioning of society more heavily. Multipliers range from 0.5 to 2 over a four-level scale.
3. **Geographical scope of the APT's attacks**: this indicator is calculated in three steps:
   (a) Each type of targeted entity is attributed a code (1, 2, 5, 10, 15).
   (b) For each incident attributed to an APT group, the codes for all targeted geographical entities are summed. The average across all incidents is then calculated to obtain a single score.
   (c) The averages obtained are converted to a four-level scale.

4. **Frequency of the APT's attacks**: this indicator is calculated by dividing the total number of attacks attributed to the APT group within the EuRepoC database by the number of years the group has been active. The resulting scores are converted to a four-level scale.

5. **Use of zero-days**: this indicator calculates the percentage of attacks attributed to the APT that make use of one or more zero-day vulnerabilities. The resulting scores are converted to a three-level scale.

The overall Threat Level Index is computed as the sum of all converted scores obtained from the five indicators (maximum 24 points):

Intensity+Sectorial scope+Geographical scope+Frequency of attacks+Use of zero-days

This sum is then converted to a four-level scale to enable meaningful interpretation of the score through an assigned qualitative label.

2.2 The CTI Cycle

The **Cyber Threat Intelligence (CTI) cycle** [6] adapts the traditional intelligence cycle used in military and national security contexts to the domain of cybersecurity. Intelligence, in this context, refers to the process of collecting, processing, and analysing information about a competitive or hostile entity and its agents, in order to support an organization's or group's security and overall well-being. Crucially, intelligence is best understood as both a process (the cycle itself) and a product (the resulting analysis delivered to a consumer). Within this framework, CTI is more narrowly defined as analysed information concerning the intent, capability, and opportunity of an adversary, tailored to satisfy a specific requirement, with particular emphasis on the human threat actor behind the activity, a point that distinguishes CTI from purely technical indicator-sharing.

2.2.1 Planning and Direction

The CTI cycle begins with **Planning and Direction**, in which the objectives and scope of the intelligence function are established. A CTI team's mandate typically spans three functions:

- **Preventative function**, such as Security Operations Center (SOC) support, alerting, and triage
- **Response function**, such as Incident Response support
- **Strategic support function** (mainly informing business decisions and resource prioritization)

Where a CTI team sits organizationally has practical consequences, since proximity to one part of the organization can bias its output toward that unit's concerns; a centrally

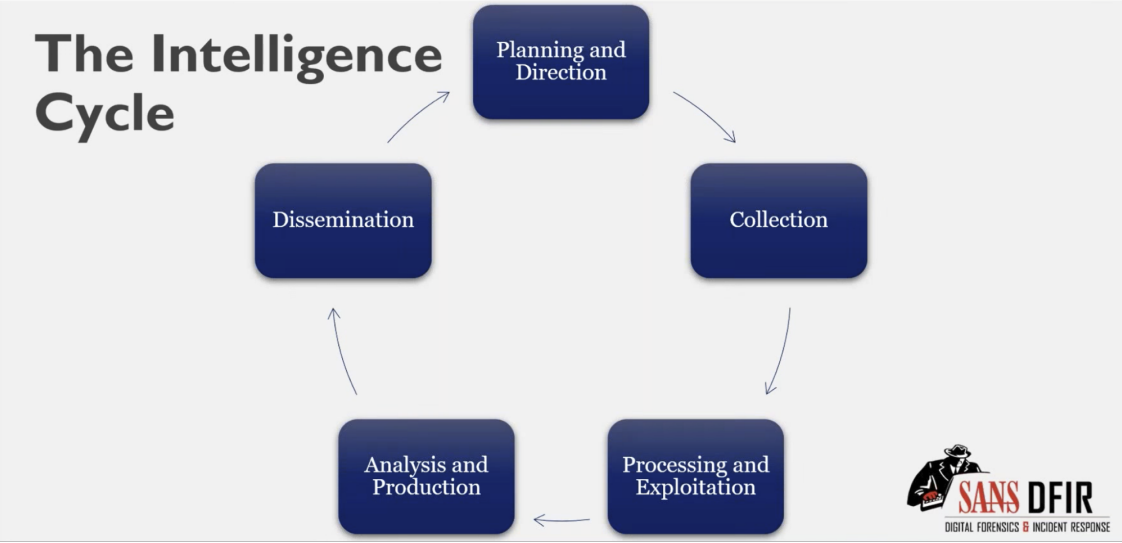


Figure 2.1. The CTI Cycle from [6]

positioned team, supported by diversity of background, culture, and thought, is generally considered best practice for mitigating this risk. This phase is operationalized through three components:

- **Intelligence Requirements (IRs)** [10]: objectives that analyses seek to satisfy through the intelligence process. They should be clearly articulated and available to the intelligence team and its consumers (within the company). IRs are generated by seeking input from intelligence consumers [9] (other organization members) and translating that input into a requirement, while avoiding the "self-licking ice cream cone" problem (i.e., producing useless intelligence). IRs can exist at the strategic, operational, or tactical level.

- **Threat modelling**: this is about knowing both ourselves and the adversary, in particular what assets we hold that an adversary might be interested in (e.g., financial data, intellectual property, or system availability).

- **Collection Management Framework**: a view of the available data sources, what information is available within that data, and how it is processed and exploited. Essentially, it identifies which requirements the analysts can fulfil using the available data.

2.2.2 Collection

The **Collection** phase encompasses the range of sources from which raw data is gathered. **Intrusion analysis**, formalized through models such as the **Lockheed Martin Kill Chain**, draws on internal telemetry to reconstruct the stages of a single intrusion.

Malware collection remains a central source of data; public malware sandboxes ("malware zoos" such as VirusTotal, Hybrid-Analysis, and Joe Sandbox) provide accessible repositories, though analysts should be aware that adversaries can access the same data. **Domain-based data pivoting** allows analysts to expand their picture of adversary infrastructure, but carries the risk of the "*Kevin Bacon effect*", whereby increasingly distant and tenuous connections are drawn between unrelated domains; as a rule, the closer the pivot, the more reliable the resulting link. **External threat data feeds**, typically composed of IP addresses, file hashes, filenames, and other atomic or computed indicators can be integrated directly into an organization's security architecture, provided analysts critically assess the data's provenance, relevance to the organization's threat profile, and intended use. Finally, **TLS certificate data**, aggregated by services such as Censys.io, Scans.io, Circl.lu, and PassiveTotal, can help identify command-and-control (C2) infrastructure, since the re-use of certificates across ostensibly distinct infrastructures can reveal shared tooling or overlapping campaigns.

2.2.3 Processing and Exploitation

Once collected, raw data must be structured to support analysis. Structured models (including the **Kill Chain** [4], the **Diamond Model**, **MITRE ATT&CK** [8], and **VERIS**) allow analysts to categorize data, enabling abstraction and the identification of recurring patterns. The **Diamond Model**, in particular, formalizes an intrusion event around **four core features** (adversary, infrastructure, capability, and victim), while **MITRE ATT&CK** provides a widely adopted taxonomy of **adversary tactics and techniques**, facilitating consistent documentation across incidents and organizations. Processed intelligence must then be stored in a format that supports rapid retrieval, typically within a **Threat Intelligence Platform (TIP)**. Open-source TIPs (e.g., CRITS, MISP, Threat_Note, YETI) offer cost-free deployment and access to open sharing communities, but can be comparatively difficult to implement and maintain; commercial TIPs, by contrast, offer vendor support, easier installation, and integrated analytics, at the cost of licensing fees and potential misalignment with existing workflows.

2.2.4 Analysis and Production

The **analytic** phase is where raw and processed data are transformed into assessments. This phase is particularly vulnerable to cognitive bias, as analysts' differing backgrounds and cultural frames of reference can lead to distorted interpretations of the same evidence. Confirmation bias is a paradigmatic example, whereby analysts selectively seek evidence supporting a favoured hypothesis while discounting disconfirming evidence. **Structured Analytic Techniques (SATs)** such as the **Analysis of Competing Hypotheses**, **Devil's Advocacy**, **Team A/Team B exercises**, **structured brainstorming**, and **Red Team analysis** are designed to counteract these biases by imposing methodological rigor and distancing the analyst from unexamined assumptions. Effective analysis also requires deliberately varying analytic approaches rather than defaulting to habitual or preferred methods. A core analytic task is the correlation of activity into clusters, variously labelled as **threat actors**, **activity groups**, **campaigns**, or **intrusion sets**,

depending on the methodology employed. Activity groups, for instance, are defined as clusters of intrusions linked through an analyst's or team's confidence-weighted judgment; a common shortcut for establishing such links is the Diamond Model's "*Rule of 2*," whereby analysts look for overlaps between any two of the model's four vertices across separate intrusions as grounds for clustering them together.

2.2.5 Dissemination

The final phase, **Dissemination**, concerns the effective delivery of intelligence products to their intended consumers. Because different audiences have different needs in terms of both content and format, **understanding the audience** is the single most important factor in effective dissemination. The intended audience and its objectives determine whether the resulting intelligence is **strategic**, **operational**, or **tactical** in nature. To ensure analytic rigour and transparency, CTI assessments are typically communicated using a structured formula combining confidence level, analysis, evidence, and sources (e.g., "We assess with [confidence] that [assessment], because of [evidence] from [sources]"). Confidence levels are themselves standardized:

- **High Confidence** reflects a preponderance of supporting evidence with no significant contradicting evidence.
- **Moderate Confidence** indicates that meaningful evidentiary gaps remain, such that new information could alter the assessment.
- **Low Confidence**: other equally plausible hypotheses exist, and little evidence is available to support the assessment; nonetheless, it is still better than nothing.

```
ASSESSMENT = CONFIDENCE + ANALYSIS + EVIDENCE + SOURCES
```

2.3 The Diamond Model of Intrusion Analysis

2.3.1 Introduction

Cyber threat intelligence relies on structured frameworks to make sense of otherwise fragmented and noisy indicators of malicious activity. Among these, the Diamond Model of Intrusion Analysis, introduced by Caltagirone, Pendergast, and Betz [1], has become one of the most widely adopted conceptual tools for organizing and reasoning about cyber intrusions. Unlike purely descriptive taxonomies, the Diamond Model provides both a formal representation of malicious activity and a methodology for exploiting that representation analytically, a property that distinguishes it from adjacent frameworks such as the Cyber Kill Chain or the MITRE ATT&CK matrix, with which it is often used in combination rather than in competition.

At its core, the model rests on a simple but fundamental claim: every malicious act, or *event*, can be described as an *adversary* deploying a *capability* over some *infrastructure* against a *victim*. These four elements, the model's **core features**, are represented as the vertices of a diamond, with the edges between them capturing the natural relationships

that link adversary operations together. Analysts populate these vertices as new events are discovered, and by systematically exploring the connections between vertices, a technique the model formalizes as *pivoting*, they can progressively reconstruct a fuller picture of an adversary's operations from a single, partial observation.

This section presents the theoretical foundations of the Diamond Model in detail. It first introduces the notion of an event as the atomic unit of analysis and its formal representation as a labelled tuple. It then examines each of the model's core and meta-features individually, before turning to how events are aggregated into activity threads, campaigns, and activity groups. It then discusses analytic pivoting and the six feature-centered analytical approaches the model enables. Finally, Section 2.3.7 motivates the use of the Diamond Model as the organizing structure adopted in this thesis to map and relate the incident-level information provided by the **European Repository of Cyber Incidents (EuRepoC)** [2].

2.3.2 The Event as the Atomic Unit of Analysis

The Diamond Model treats the **event** as its fundamental, atomic unit of analysis. An event is defined as a discrete, time-bound activity, occurring within a specific *phase* of an intrusion, in which an *adversary*, drawing on external *resources*, employs a *capability* and a *methodology* over some piece of *infrastructure* against a *victim*, producing a given *result*. Formally, an event E is represented as a labelled n-tuple, in which each element pairs a knowledge feature with an independent confidence value:

$$E = \langle (f_1, c_1), (f_2, c_2), \ldots, (f_n, c_n) \rangle \tag{2.1}$$

Any individual feature within this tuple may itself be expanded into a nested sub-tuple where finer-grained detail is required or available, allowing the model to scale from a coarse, high-level description of an event down to a highly granular technical account, without altering its underlying structure.

Because most features of an event are unknown or only partially known at the moment of initial discovery, the model does not require complete knowledge to be useful. Instead, an event's tuple is expected to be populated incrementally as an investigation matures and new evidence emerges. This progressive character is reflected directly in the model's treatment of confidence, described in Section 2.3.2, and is one of the reasons the Diamond Model is well suited to the iterative, evidence-driven nature of real-world intrusion analysis.

An event can equally be represented graphically, as a diamond-shaped structure in which the four core features occupy the vertices and the edges denote the natural analytic relationships between them (Figure 2.2). This dual representation, tuple and graph, is not incidental: the graph form is what makes the model's central analytic technique, pivoting, tractable, since it allows an analyst to visualize which features are reachable from a given known feature.

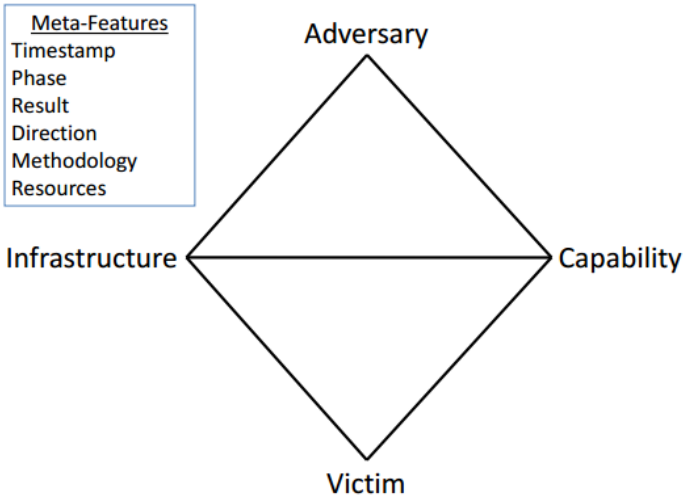


Figure 2.2. Graphical representation of an event as a diamond, adapted from Caltagirone et al. [1].

Confidence Values

Every feature of an event, whether core or meta, carries an associated **confidence value**. The model deliberately leaves the semantics of this value undefined, since different organizations and analytic contexts may require different notions of confidence, a qualitative scale (e.g., low/medium/high), a probabilistic estimate, or a more elaborate scoring scheme. Because confidence is frequently the product of several underlying considerations (e.g., source reliability, corroboration, recency), it too may be itemized as a sub-tuple, allowing its individual determinants to be tracked and reasoned about separately rather than collapsed into a single opaque score.

2.3.3 Core Features

Adversary

The **adversary** is the actor or organization responsible for employing a capability against a victim in pursuit of some intent. Adversary knowledge is typically the most elusive of the four core features and is often absent, or held with very low confidence, at the time an event is first discovered, attribution, where it is possible at all, is usually the product of extensive pivoting across many events rather than a starting point.

The model distinguishes two roles within the adversary feature, which need not be occupied by the same entity:

- **Adversary Operator**, the individual actually conducting the intrusion activity.
- **Adversary Customer**, the entity that stands to benefit from the intrusion, which may be the operator itself or a separate sponsoring person or organization (e.g., in

cybercrime-as-a-service or state-sponsored operations where operators act on behalf of a client).

This distinction is analytically important: conflating operator and customer can lead to misattribution, particularly in cases of contracted cybercrime or proxy operations where the technical origin of an intrusion diverges from its ultimate beneficiary.

Capability

The **capability** feature describes the tools and techniques an adversary uses to carry out an event. The model does not prescribe a fixed level of granularity for describing capability; an analyst may describe it broadly (e.g., "remote access trojan") or with considerable technical specificity (e.g., a particular malware family, version, and configuration), depending on the fidelity of available evidence and analytic need.

Two auxiliary concepts refine the capability feature:

- **Capability Capacity**, the full set of vulnerabilities and exposures that a given capability can exploit, independent of any specific victim. This represents the theoretical reach of a tool or technique.
- **Adversary's Arsenal**, the complete set of capabilities available to an adversary, and correspondingly the union of the capacities of those individual capabilities. Where capability capacity is known, it should be documented as a sub-tuple of the capability feature, and can additionally inform the construction of activity-attack graphs mapping potential future adversary paths.

Command and Control (C2), borrowed conceptually from military doctrine as the exercise of authority and direction over assets, is treated within the Diamond Model as a *sub-feature of capability* rather than a standalone core feature. In the context of intrusion analysis, C2 refers to the channels, communication structures, protocols, and content used by an adversary to direct and receive effect from a capability already deployed. Because C2 is ultimately determined by the capability in use, analysts pivot over it primarily by discovering communication patterns between infrastructure and victims, rather than treating it as an independent analytic axis.

Infrastructure

The **infrastructure** feature describes the physical or logical communication structures an adversary uses to deliver a capability, maintain control over it, and extract results from a victim. The model distinguishes three roles infrastructure can play:

- **Type 1 Infrastructure**, infrastructure fully controlled by the adversary, or over which they have direct physical proximity (e.g., adversary-owned servers).
- **Type 2 Infrastructure**, infrastructure controlled by a witting or unwitting intermediary. This is typically what a victim directly observes as "the adversary," and it functions to obfuscate the true origin of an operation and complicate attribution

(e.g., compromised "zombie" hosts, malware staging servers, or malicious domains registered under false identities).

- **Service Providers**, organizations that, wittingly or unwittingly, supply services critical to the availability of Type 1 and Type 2 infrastructure, such as Internet Service Providers, hosting companies, or domain registrars.

This distinction is analytically significant because it separates infrastructure an analyst can meaningfully attribute to the adversary from infrastructure that merely intermediates or supports the operation, collapsing the two risks both misattribution and wasted investigative effort directed at innocent third parties.

Victim

The **victim** is the target of the adversary, the entity against whom vulnerabilities and exposures are exploited and capabilities are directed. The model separates the victim into two analytically distinct sub-features:

- **Victim Persona**, the people or organizations being targeted (e.g., an organization's name, or the identity of a targeted individual). Victim personae are particularly useful for non-technical analysis, such as cyber-victimology or socio-political approaches to threat assessment.
- **Victim Asset**, the technical attack surface associated with the victim: networks, systems, hosts, email addresses, IP addresses, and similar resources against which the adversary directs their capability. Victim assets may lie inside or outside the persona's direct control or visibility while still being reachable by an adversary (for instance, third-party webmail accounts or cloud storage used by an organization's employees). Notably, a victim asset compromised in one event may subsequently be repurposed as infrastructure in a later event, for example, a compromised host used as a staging server for further attacks.

Associated with the victim is the concept of **Victim Susceptibility**, the set of vulnerabilities and exposures that render the victim liable to exploitation. This is expressed as a sub-tuple of the victim feature and provides the technical basis on which capability-victim interactions in an event can be assessed.

2.3.4 Meta-Features

Beyond the four core features, the Diamond Model defines a set of **meta-features** that, while not essential to defining an event, substantially enrich the model's descriptive and analytic power and are used to order events within an activity thread.

Timestamp

The **timestamp** records the date and/or time at which an event occurred. Timestamps are fundamental to grouping malicious activity chronologically and to establishing periodicity or behavioural patterns in adversary operations. They also play an epistemic role:

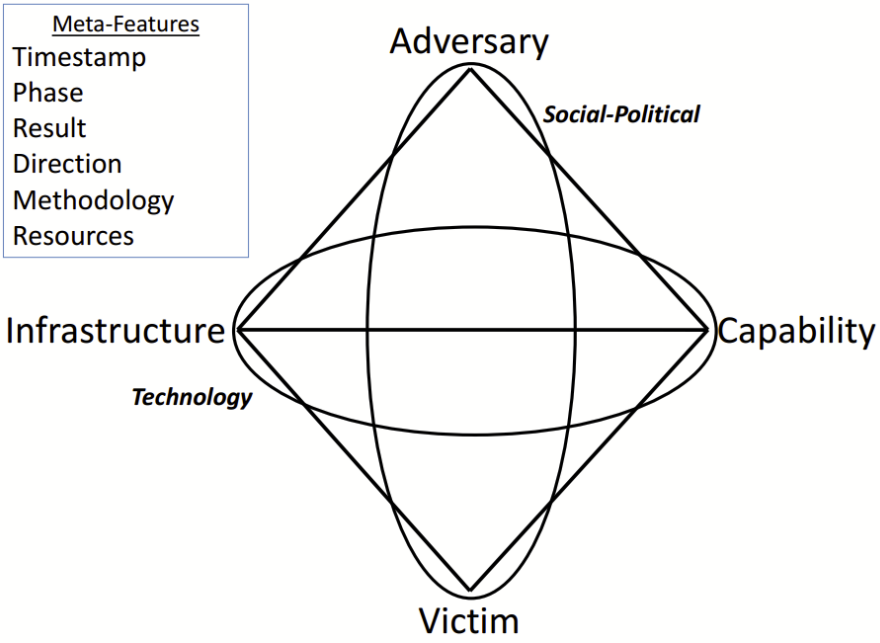


Figure 2.3. Metafeatures of the Diamond Model

as time elapses since an event, the confidence an analyst can reasonably hold in other features tends to decrease, since the likelihood of underlying adversary infrastructure, personnel, or tooling having changed increases correspondingly.

Phase

Malicious activity rarely consists of a single, isolated event; rather, real-world intrusions typically unfold as a chain of events, often employing different capabilities and infrastructure at different stages. The Diamond Model formalizes this progression through **phases**. Formally, the phases of an intrusion are defined as an ordered n-tuple

$$P = (p_1, p_2, \ldots, p_n), \tag{2.2}$$

where n is the number of phases an organization deems necessary and sufficient to describe the full space of possible events, and where:

- $n \geq 2$ (an intrusion is modeled as consisting of at least two phases);
- p_1 denotes the first phase of an adversary's operations;
- p_{i+1} denotes a phase executed subsequent to p_i;
- every event's phase is one, and only one, element of P.

Each event is thus assigned to exactly one phase, and the ordered sequence of phases across an adversary's events constitutes an activity thread (Section 2.3.5). This construction is deliberately compatible with, but not dependent on, external phase models such as the Cyber Kill Chain, which an organization may adopt as its concrete instantiation of P.

Result

The **result** meta-feature captures the post-conditions of an event, that is, what the adversary actually achieved. Results are frequently unknown, or known with low confidence, but are nonetheless valuable to capture wherever possible, since aggregating results across many events allows an analyst to estimate an adversary's success rate with particular capabilities or against particular classes of victim, and can more broadly inform assessments of adversary intent.

The model does not mandate a single scheme for documenting results; three approaches are commonly used:

1. A simple three-value tuple: *Success*, *Failure*, *Unknown*.

2. A classification by compromised security property: *Confidentiality Compromised*, *Integrity Compromised*, *Availability Compromised*.

3. A full enumeration of the specific post-conditions resulting from the event (e.g., exfiltration of targeting information, or a password later reused in a masquerade attack).

Direction

Directionality describes the direction in which an event's activity flows, and is particularly relevant when evaluating mitigation options or the optimal placement of detection capabilities. It is most naturally applied to network-based events, though it can equally describe host-based activity. The model defines seven possible values: Victim-to-Infrastructure, Infrastructure-to-Victim, Infrastructure-to-Infrastructure, Adversary-to-Infrastructure, Infrastructure-to-Adversary, Bidirectional, and Unknown.

Methodology

The **methodology** feature allows an analyst to characterize the general class of activity involved in an event, for instance, a spear-phishing email, a content-delivery attack, a SYN flood, or a port scan. Unlike most other features, methodology explicitly permits more than one simultaneous value per event where the activity spans multiple recognizable techniques. By abstracting away from specific indicators, methodology enables indicator-independent comparison of events, both within a single adversary's operations and across distinct adversaries, supporting activity grouping and mitigation planning.

Resources

The **resources** meta-feature enumerates the external resources an event depends upon in order to be carried out, broadly, any supporting element without which the event could not occur. Typical categories include *software*, *hardware*, and *knowledge*, though an organization need only document the categories relevant to its own analytic and operational needs.

Additional Meta-Features

The Diamond Model is explicitly extensible: organizations may define additional meta-features tailored to their own analytic requirements. Commonly proposed candidates include:

- **Data Source**, the origin of the data used to capture or detect the event;
- **Author**, the analyst responsible for documenting the event;
- **Detection Method**, the tool, technique, or capability that detected the event;
- **Detection Signature**, the specific signature or heuristic responsible for detection.

The inclusion of such additional meta-features allows analysts and organizations to preserve operationally useful context for future reference without altering the model's core structure.

2.3.5 From Events to Activity Threads, Campaigns, and Activity Groups

Individual events acquire greater analytic value once they are related to one another. The Diamond Model achieves this through a layered aggregation process:

1. **Activity Threads**, events sharing a common adversary-victim pair are ordered by phase into an activity thread, representing the temporal flow of a single adversary's operation against a single victim.
2. **Campaigns**, once activity threads are established, events can be correlated *across* threads to identify broader adversary campaigns, revealing operations that span multiple victims or intrusion attempts over time.
3. **Activity Groups**, events sharing common features, whether or not they belong to the same campaign, can be coalesced into activity groups, which capture threats exhibiting similar behaviour, tooling, or targeting.

Activity groups serve two complementary purposes: they support the automated correlation of newly discovered events against known adversary behaviour, and they provide a basis for gaming and planning, that is, for developing and evaluating strategic mitigation options intended to counter an adversary's broader pattern of operations, rather than merely responding to individual incidents in isolation.

2.3.6 Analytic Pivoting

Pivoting is the analytic technique of extracting a known data element from an event and exploiting it, together with available data sources, to discover related elements not yet known. It is, at its core, an exercise in hypothesis testing: each feature within an event generates hypotheses about the other features to which it is connected, and further evidence is then sought to strengthen, weaken, or revise those hypotheses. The success of pivoting depends both on the analyst's understanding of the relationships between features and on their practical ability to exploit relevant data elements and data sources.

The Diamond Model was, in fact, originally developed through the exploration of pivoting scenarios, and its core features are deliberately arranged as a diamond precisely so that the edges connecting them highlight the pivot opportunities available to an analyst. A representative pivot chain illustrates the technique:

1. A victim discovers malware on their network.
2. The malware is reverse-engineered, exposing its command-and-control (C2) domain.
3. The C2 domain is resolved, exposing the underlying IP address hosting the controller.
4. Firewall logs are examined, revealing other compromised hosts within the victim's network communicating with the now-identified controller IP address.
5. The IP address's registration records are examined, potentially revealing adversary details and offering a path toward attribution.

This chain illustrates how a single initial observation, malware discovered on one host, can, through successive pivots across the model's connected features, expand into a substantially fuller picture of the underlying adversary operation. The Diamond Model formalizes six general approaches to pivoting, each anchored on a different core feature or contextual dimension.

Victim-Centered Approach

This is the approach most commonly exercised by organizations through routine network and host monitoring, detection, and defensive operations. Analysis of data associated with a potential victim reveals the other connected elements of the diamond, namely, the malicious capabilities and infrastructure involved. Large-scale deployments such as honeypot projects, and more targeted monitoring efforts directed at specific user populations, exemplify this approach.

Capability-Centered Approach

This approach exploits the features of a known capability to uncover the other elements connected to it in adversary operations: victims, the infrastructure supporting the capability, the underlying technology enabling it, clues pointing to other related capabilities, and potentially the adversary itself. Antivirus-vendor research exemplifies this approach

particularly well, for instance, the technical analyses by Symantec and CrySyS that linked the Stuxnet and Duqu malware families, or Kaspersky's investigation of the "Red October" campaign.

Infrastructure-Centered Approach

This approach begins from the adversary's malicious infrastructure and, from that starting point, uncovers other connected elements: victims observed communicating with the infrastructure, capabilities delivered or controlled through it, related infrastructure, and possibly clues regarding the adversary or intermediary controlling it. The investigation conducted by the Command Five research team into the so-called "SK Hack" incident illustrates the practical application of this approach.

Adversary-Centered Approach

Arguably the most difficult of the six approaches, given that it is fundamentally constrained by the analyst's access to the adversary, this approach is also potentially the most fruitful when feasible. It involves direct monitoring of an adversary in order to discover their infrastructure and capabilities, as exemplified by the FBI's monitoring of the "Phonemasters" hacking group.

Social-Political-Centered Approach

This approach is distinctive in that it does not directly yield new technical elements or indicators on its own. Instead, it exploits an expected relationship between adversary and victim, grounded in socio-political context, to generate hypotheses about who might plausibly be a victim and who their corresponding adversary might be, or vice versa. These hypotheses can then be pursued using the adversary-centered or victim-centered approaches to obtain concrete tactical detail. Notable historical examples include Cheswick's correlation, as early as 1990, between intrusion activity observed against his network and the 1990–91 Gulf War, and, more recently, the correlation drawn between the 2008 Georgia DDoS attacks and sustained attacks against pro-Tibet groups with contemporaneous political events. This approach, however, demands particular methodological caution, since the well-known dictum that *correlation is not causation* applies with special force here.

Technology-Centered Approach

This final approach targets potential misuse or anomalous use of a given technology in order to discover previously unknown infrastructure and capabilities that exploit it. Monitoring and anomaly detection within the Domain Name System (DNS) has proven a particularly popular and productive instantiation of this approach; researchers have similarly explored anomalies in packet headers observed on backbone networks as a further avenue for discovering malicious activity.

2.3.7 Applying the Diamond Model to the EuRepoC Data

The preceding sections presented the Diamond Model in its general, abstract form. This section motivates its adoption as the organizing structure for the empirical work of this thesis, namely the mapping of incident-level information published by the European Repository of Cyber Incidents (EuRepoC) [2] onto a coherent relational framework.

EuRepoC's codebook already records, for each cataloged incident, information that maps naturally onto several of the Diamond Model's core and meta-features: attributed or alleged threat actors correspond to the *adversary* feature; targeted states, sectors, and organizations correspond to *victim* personae and, where available, specific victim assets; the type of cyber operation and any documented tooling correspond to *capability*; and incident dates and observed effects correspond respectively to *timestamp* and *result*. What the raw EuRepoC dataset does not provide directly, however, is an explicit *relational* structure linking these dimensions together at the level of an individual, atomic incident, nor a principled way of aggregating related incidents into broader adversary campaigns or activity groups.

The Diamond Model is adopted in this thesis to support the mapping of EuRepoC information. While the primary standard used in CTIS is STIX 2.1, the Diamond Model has heavily influenced the composition of CTIS entities as visible in 2.4. By re-expressing each EuRepoC-recorded incident as a combination of STIX entities, populating the four core features of the Diamond Model with the corresponding EuRepoC fields, and layering the relevant meta-features (timestamp, phase, result, methodology) on top, it becomes possible to:

1. represent the internal structure of a single incident explicitly, including the confidence with which each attributed feature (particularly adversary attribution) is held;

2. exploit the pivoting techniques described in Section 2.3.5 to identify when multiple EuRepoC incidents plausibly belong to the same activity thread or campaign, even where EuRepoC's own categorization does not make that link explicit; and

3. construct activity groups across the dataset, supporting higher-level analysis of recurring adversary behaviour, targeting patterns, and capability reuse across the repository as a whole.

The specific mapping adopted between EuRepoC codebook fields and CTIS entity features, together with the methodological choices this mapping entails, is presented in full in Chapter 4.4.5.

2.4 The STIX Standard

Structured Threat Information Expression (STIX) [7] is an open-source language and serialization format used to exchange cyber threat intelligence (CTI). It enables the clear representation of all aspects of suspicion, compromise, and attribution through a set of objects and the descriptive relationships that connect them. STIX information can

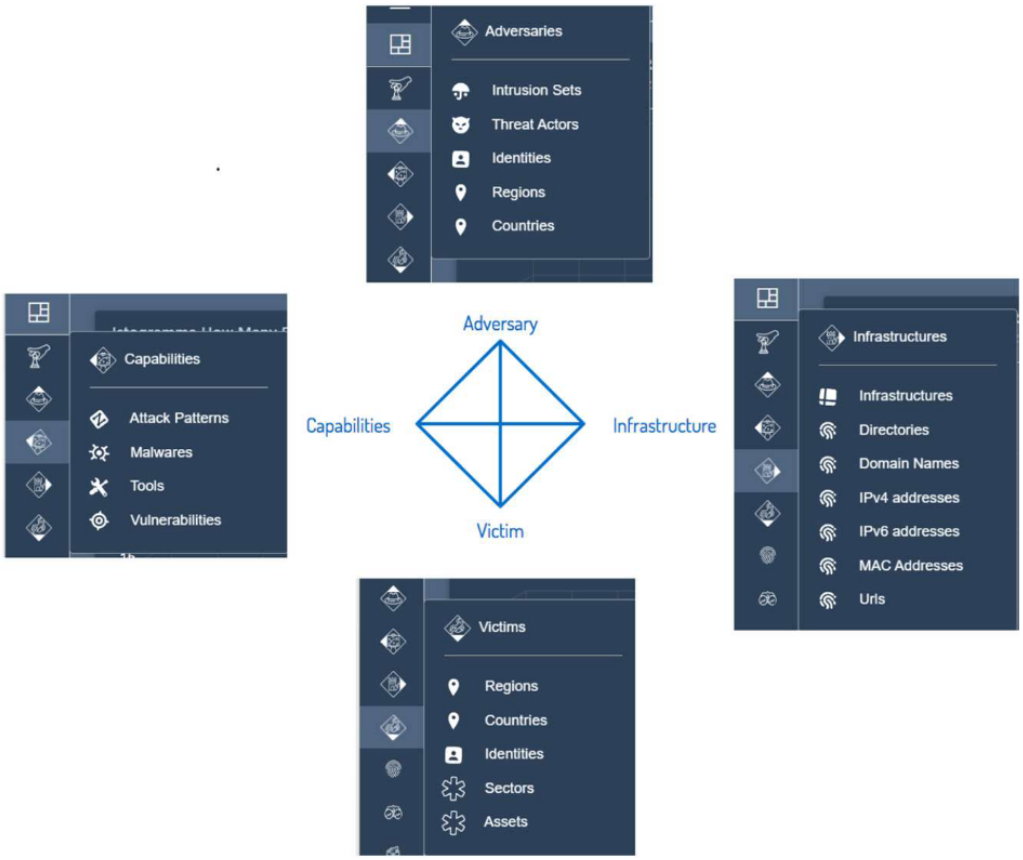


Figure 2.4. Diamond Model in CTIS

be visually represented for human analysts or stored in JSON format for fast, automated machine processing. It is the language on which **CTIS**, the **TIP** (Threat Intelligence Platform) used at **Leonardo**, is based.

The example below shows how a STIX object type (in this case, attack-pattern) is formally defined using a JSON Schema:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "attack-pattern",
  "description": "Attack Patterns are a type of TTP that describe ways that
  adversaries attempt to compromise targets. ",
  "type": "object",
  "allOf": [
    {
      "$ref": "../common/core.json"
    },
    {
```

```
      "properties": {
        "type": {
          "type": "string",
          "description": "The type of this object, which MUST be the
          literal ‘attack-pattern‘.",
          "enum": [
            "attack-pattern"
          ]
        },
        "id": {
          "title": "id",
          "pattern": "^attack-pattern--"
        },
        "name": {
          "type": "string",
          "description": "The name used to identify the Attack Pattern."
        },
        "description": {
          "type": "string",
          "description": "A description that provides more details
          and context about the Attack Pattern, potentially including
          its purpose and its key characteristics."
        },
        "kill\_chain\_phases": {
          "type": "array",
          "description": "The list of kill chain phases for which
          this attack pattern is used.",
          "items": {
            "$ref": "../common/kill-chain-phase.json"
          },
          "minItems": 1
        }
      }
    }
  ],
  "required": [
    "name"
  ]
}
```

STIX is a schema that defines a taxonomy of cyber threat intelligence, represented through the following set of objects:

- **STIX Core Objects**: any SDO, SCO, or SRO.
- **STIX Domain Objects (SDOs)**: Higher-level intelligence objects that represent

the behaviours and constructs threat analysts typically create or work with while understanding the threat landscape. The defined set of objects comprises: Attack Pattern, Campaign, Course of Action, Grouping, Identity, Indicator, Infrastructure, Intrusion Set, Location, Malware, Malware Analysis, Note, Observed Data, Opinion, Report, Threat Actor, Tool, and Vulnerability. Each of these objects corresponds to a concept commonly used in CTI.

- **STIX Cyber-observable Objects (SCOs)**: Objects that represent observed facts about a network or host, which may be used and related to higher-level intelligence to form a more complete understanding of the threat landscape. SCOs are used by various SDOs to provide supporting context.

- **STIX Relationship Objects (SROs)**: Objects that connect STIX Domain Objects to one another, STIX Cyber-observable Objects to one another, and STIX Domain Objects to STIX Cyber-observable Objects, forming a more complete understanding of the threat landscape.

- **STIX Meta Objects (SMOs)**: Objects that provide the necessary "glue" and associated metadata to enrich or extend STIX Core Objects, supporting user and system workflows.

- **STIX Bundle Object**: An object that provides a wrapper mechanism for packaging arbitrary STIX content together.

2.4.1 STIX Relationships

A relationship is a link between SDOs, SCOs, or between an SDO and an SCO that describes the way in which the objects are related. Relationships can be represented using an SRO or, in some cases, through certain properties that store an identifier reference, which constitutes an embedded relationship (e.g., the *created_by_ref* property). The generic **STIX Relationship Object (SRO)** is one of two SRO types. It contains a property called *relationship_type* that describes more specifically what the relationship represents. The specification defines a set of known terms to use for the *relationship_type* property between SDOs of specific types, but also allows for user-defined ones.

The only other SRO is the **Sighting SRO**, an object used to capture cases where an entity has "seen" an SDO. The Sighting object contains additional properties, such as *count*, that are only applicable to Sighting relationships. STIX also uses ID references to represent embedded relationships, where a reference to one object is embedded as a property within another. Embedded relationships are used when the property is an inherent part of the object, rather than something a third party might add, or something that might require the inclusion of a confidence score. An example is the entity that created a STIX Object: this is an inherent, factual part of that object, and the embedded relationship to it is captured in the *created_by_ref* property.

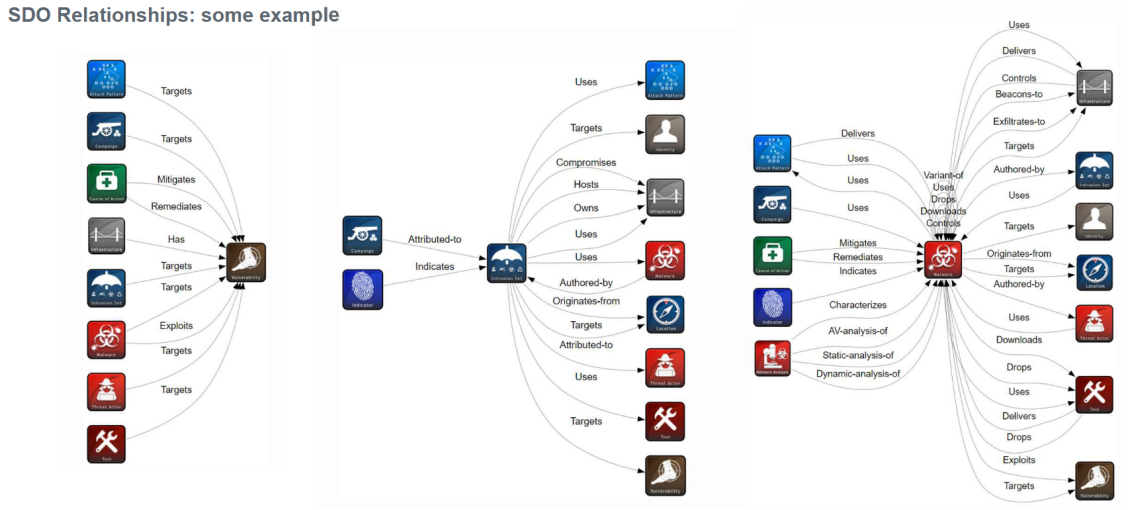


Figure 2.5. Examples of STIX Relationships

2.4.2 STIX Patterning

STIX Patterning is a language that enables the detection of activity on networks and endpoints by matching against timestamped cyber-observable data collected by a threat intelligence platform or similar system. Currently used only by the **STIX Indicator object**, which it was primarily designed to support, the language focuses on a common set of use cases and therefore allows for the expression of an initial set of patterns that producers and consumers of STIX can utilize.

SDOs and SROs all share a common set of properties that provide core capabilities such as versioning and data marking. All SCOs likewise share a common set of properties applicable to them, while SMOs use some, but not all, of the common properties.

Some STIX properties are defined using **open vocabularies** or **enumerations**; these help avoid different entities using different strings for the same concept. STIX vocabularies have predefined values, whose use is strongly encouraged, though values outside them are permitted if necessary. In the case of STIX Enumerations, however, only the predefined values are permitted.

While STIX is independent of any specific storage mechanism or serialization, all STIX-conformant tools must implement support for JSON, though they may also implement support for other serializations.

STIX is transport-agnostic, since its structures and serializations do not rely on any specific transport mechanism. The **TAXII** specification is designed specifically to transport STIX Objects, but STIX also provides a Bundle as a container for STIX Objects, allowing for the transportation of bulk STIX data (especially over non-TAXII communication mechanisms).

2.4.3 Common Data Types

The common data types used throughout STIX for all STIX Objects are as follows:

- **binary**: A data type that represents a sequence of bytes. In order to allow pattern matching on custom objects, for all properties that use the *binary* type, the property name must end with *_bin*.
- **boolean**: A value of either *true* or *false*.
- **dictionary**: Captures an arbitrary set of key/value pairs; empty dictionaries are prohibited in STIX. Values **MUST** be valid property base types.
- **enum**: A hardcoded list of terms, represented as a *string*. For properties that use this type, a defined list of values is identified in the definition of that property.
- **external-reference**: A reference type used to describe pointers to information represented outside of STIX (for example, the ID of a malware sample in another, external database). Its properties are:
  - *source_name* (required, *string*)
  - *description* (optional, *string*)
  - *url* (optional, *string*)
  - *hashes* (optional, *hashes*), specifies a dictionary of hashes for the contents of the *url*
  - *external_id* (optional, *string*)

  In addition to *source_name*, at least one of *description*, *url*, or *external_id* must be present.
- **float**: A double-precision number; *+/- infinity* and *NaN* are not valid values in STIX.
- **hashes**: Represents one or more cryptographic hashes, as a special set of key/value pairs. **SHA-256** should be used whenever possible.
- **hex**: An array of octets (8-bit bytes) represented as hexadecimal. All properties that use the *hex* type must end with *_hex*.
- **identifier**: Uniquely identifies a STIX Object, and may do so in a deterministic manner depending on the object type. All *identifiers* must follow the form *object-type-UUID*, where *object-type* is the exact value from the *type* property. The *UUID* portion must be unique across all objects produced by a given producer, regardless of the object type identified by the *object-type* prefix.
- **integer**: A data type that represents a whole number. All integers must be capable of being represented as a signed 53-bit value:

$$[-(2^{53}) + 1, \quad (2^{53}) - 1]$$

- **kill-chain-phase**: Represents a phase in a kill chain, which describes the various phases an attacker may undertake in order to achieve their objectives. Its properties are:
  - *kill_chain_name* (required, *string*); when referencing the Lockheed Martin Kill Chain, this property must be set to *lockheed-martin-cyber-kill-chain*
  - *phase_name* (required, *string*)
- **list**: Defines a sequence of values, ordered according to the sequence in which they appear in the list. The phrasing "*list* of type *«type»*" is used to indicate that all values within the list must conform to the specified type. Empty lists are prohibited.
- **open-vocab**: Represented as a *string*. A list of suggested values (a *suggested vocabulary*) is provided, from which the value of the property can be chosen, but any other *string* may be used. A consumer that receives STIX content with one or more *open-vocab* terms not defined in the suggested vocabulary may ignore those values.
- **string**: Represents a finite-length string of valid characters from the Unicode coded character set.
- **timestamp**: Defines how dates and times are represented in STIX. It must be of the format *YYYY-MM-DDTHH:mm:ssZ*, where "s+" represents one or more optional sub-second values. Timestamps must be expressed in the **UTC timezone** (indicated by the **"Z"** designation).

2.4.4 General Concepts

All type names, property names, and literals must be expressed in lowercase, except when referencing canonical names defined by another standard (for example, values taken from an **IANA** registry, which retain their original casing). Type names and property names must begin with a letter character. Words within property names must be separated with an underscore (_), while words within type names and string enumerations must be separated with a hyphen (-). Dictionary keys and hash algorithm names may use either underscores (_) or hyphens (-). Finally, all type names, property names, object names, and vocabulary terms must be between three and 250 characters in length.

These naming conventions were introduced by the STIX 2.1 specification to guarantee unambiguous parsing across implementations and to make STIX content interoperable regardless of the producing or consuming tool. Enforcing a single, predictable casing and separator scheme also simplifies serialization to and from JSON, since property names double as JSON object keys.

There is a set of **Common Properties** that may appear on STIX Objects. While some STIX Objects use all of these common properties, not every object type does; each type of STIX Object explicitly defines which common properties are required, optional, or not applicable to it. This mechanism allows the specification to maintain a consistent, predictable core structure across otherwise heterogeneous object types (STIX Domain

Objects, STIX Relationship Objects, STIX Cyber-observable Objects, and Meta Objects), while still letting each type diverge where needed.

- `type` (`string`) — identifies the type of STIX Object; the value must correspond to one of the defined STIX object types (e.g. `indicator`, `malware`, `relationship`).
- `spec_version` (`string`) — indicates the version of the STIX specification used to represent the object (e.g. `2.1`); this allows tools to correctly interpret objects created under different specification versions.
- `id` (`identifier`) — a globally unique identifier for the object, composed of the object's type followed by a UUID (e.g. `indicator-`*UUID*); it is immutable for the lifetime of the object, even across modifications.
- `created_by_ref` (`identifier`) — an identifier reference to the `identity` object that describes the entity that created this object.
- `created` (`timestamp`) — the time at which the object was originally created; this value does not change when the object is later modified.
- `modified` (`timestamp`) — the time at which the object was last modified; for objects that support versioning, a new version is created whenever this timestamp changes.
- `revoked` (`boolean`) — used only by STIX Objects that support versioning; indicates whether the object has been revoked and should therefore no longer be considered valid or actionable by consumers.
- `labels` (`list` of type `string`) — a set of terms used to describe the object, allowing producers to add descriptive or categorical tags without extending the schema.
- `confidence` (`integer`, ranging from 0 to 100) — expresses the level of confidence the object's creator has in the correctness of the information it contains; higher values indicate greater confidence.
- `lang` (`string`) — specifies the language in which the object's textual content is expressed; if absent, the language is assumed to be `en` (English).
- `external_references` (`list` of type `external-reference`) — points to external sources of information related to the object, such as reports, articles, or entries in external databases (e.g. CVE, CAPEC).
- `object_marking_refs` (`list` of type `identifier`) — a list of `id` properties of `marking-definition` objects that apply to this object, used to convey handling restrictions (e.g. TLP markings).
- `granular_markings` (`list` of type `granular-marking`) — allows data markings to be applied to specific properties or list items within the object, rather than to the object as a whole, enabling fine-grained control over information sharing.

- `defanged` (`boolean`) — indicates whether the data contained within the object has been defanged (i.e. altered to prevent accidental execution or resolution, such as replacing `http` with `hxxp`); the default value is `false`.

- `extensions` (`dictionary`) — specifies any extensions applied to the object, allowing producers to add custom properties or behaviour to standard STIX types without breaking compatibility with consumers that do not recognize the extension.

There are a number of Common Relationship types defined in the STIX 2.1 specification that can be used between any SDOs (STIX Domain Objects) and SCOs (STIX Cyber-observable Objects), regardless of their more specific, object-type-defined relationships. These common relationship types exist to cover generic connections that recur across many object types, so that the specification does not need to redefine an equivalent relationship separately for every possible pairing of object types.

- `derived-from` — used between SDOs or SCOs of the same type; asserts that the information contained in the target object is based on, or was derived from, information contained in the source object. This relationship is typically used to preserve provenance when one object is generated through analysis, transformation, or enrichment of another (e.g. an `indicator` derived from a `malware` analysis report).

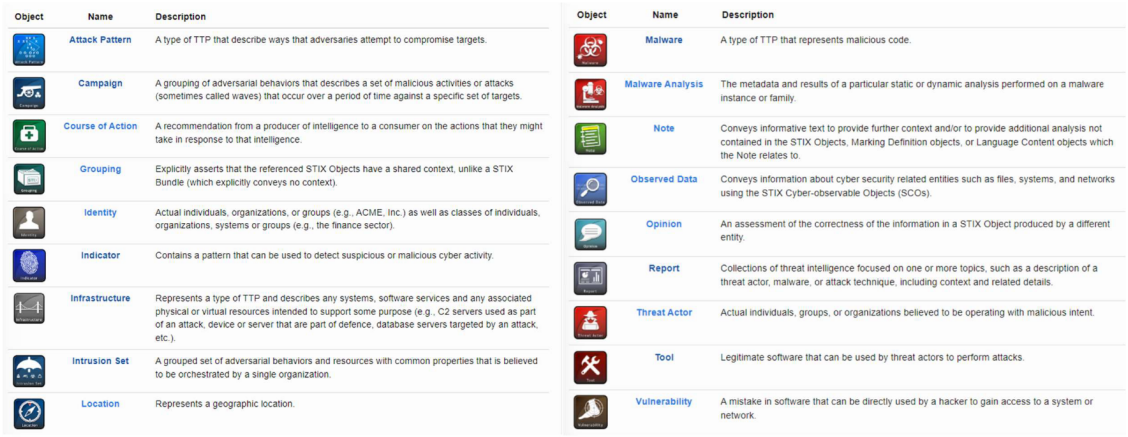
- `duplicate-of` — used between SDOs or SCOs of the same type; asserts that the source and target objects are semantically equivalent, i.e. they represent the same real-world entity or observation, even though they may have been created independently (for example, by two different producers). This relationship allows consumers to reconcile duplicate data without discarding either object outright.

- `related-to` — used between SDOs or SCOs of any type; asserts a generic, non-specific relationship between two objects. It is intended to be used only when none of the predefined, type-specific relationships apply, and when defining a new custom relationship type is not necessary or desirable. Because it carries no semantic precision beyond "these two objects are related," its use is generally discouraged when a more specific relationship type exists, as it reduces the analytical value of the resulting graph.

Unlike the type-specific relationships defined for individual SDOs (e.g. `indicates`, `uses`, `targets`), these common relationships are intentionally generic and apply uniformly across the entire object model. This uniformity is what allows any two STIX Objects to be connected in a meaningful way regardless of type.

2.4.5 STIX Domain Objects

This section describes the main STIX Domain Objects (SDOs). While other SDOs exist in the specification, the focus here is restricted to those used in the construction of the feeder.



Object	Name	Description
	Attack Pattern	A type of TTP that describe ways that adversaries attempt to compromise targets.
	Campaign	A grouping of adversarial behaviors that describes a set of malicious activities or attacks (sometimes called waves) that occur over a period of time against a specific set of targets.
	Course of Action	A recommendation from a producer of intelligence to a consumer on the actions that they might take in response to that intelligence.
	Grouping	Explicitly asserts that the referenced STIX Objects have a shared context, unlike a STIX Bundle (which explicitly conveys no context).
	Identity	Actual individuals, organizations, or groups (e.g., ACME, Inc.) as well as classes of individuals, organizations, systems or groups (e.g., the finance sector).
	Indicator	Contains a pattern that can be used to detect suspicious or malicious cyber activity.
	Infrastructure	Represents a type of TTP and describes any systems, software services and any associated physical or virtual resources intended to support some purpose (e.g., C2 servers used as part of an attack, device or server that are part of defence, database servers targeted by an attack, etc.).
	Intrusion Set	A grouped set of adversarial behaviors and resources with common properties that is believed to be orchestrated by a single organization.
	Location	Represents a geographic location.

Object	Name	Description
	Malware	A type of TTP that represents malicious code.
	Malware Analysis	The metadata and results of a particular static or dynamic analysis performed on a malware instance or family.
	Note	Conveys informative text to provide further context and/or to provide additional analysis not contained in the STIX Objects, Marking Definition objects, or Language Content objects which the Note relates to.
	Observed Data	Conveys information about cyber security related entities such as files, systems, and networks using the STIX Cyber-observable Objects (SCOs).
	Opinion	An assessment of the correctness of the information in a STIX Object produced by a different entity.
	Report	Collections of threat intelligence focused on one or more topics, such as a description of a threat actor, malware, or attack technique, including context and related details.
	Threat Actor	Actual individuals, groups, or organizations believed to be operating with malicious intent.
	Tool	Legitimate software that can be used by threat actors to perform attacks.
	Vulnerability	A mistake in software that can be directly used by a hacker to gain access to a system or network.

Figure 2.6. STIX Domain Objects

Attack Pattern

Attack Patterns are a type of **TTP** (Tactics, Techniques, and Procedures) that describe the ways in which adversaries attempt to compromise their targets. A well-known example is "spear phishing," although attack patterns can also be defined at a much more specific and technical level, for instance by being explicitly mapped to MITRE ATT&CK techniques through the object's `external_references` property.

In the context of this work, Attack Pattern objects were used in the construction of the feeder to map which MITRE ATT&CK *Impact* or *Initial Access* tactics, and their associated techniques, were employed by the initiator of a given event. This mapping allows the feeder to enrich raw event data with a standardized, machine-readable representation of adversary behaviour, rather than relying on free-text descriptions.

Identity

The Identity SDO can represent actual individuals, organizations, or groups, as well as broader classes of individuals, organizations, systems, or groups. It captures basic identifying information about the entity it describes, allowing other SDOs (such as Threat Actor or Intrusion Set) to reference who created, targeted, or is otherwise associated with a piece of information.

Intrusion Set

An Intrusion Set is a grouped set of adversarial behaviours and resources with common properties, believed to be orchestrated by a single organization. It can encompass multiple Campaigns, or other activities, that are tied together by shared attributes indicating a commonly known or unknown Threat Actor.

The distinction between a Campaign and an Intrusion Set is important: a *Campaign* is a set of attacks carried out over a period of time against a specific set of targets in pursuit of a defined objective, whereas an *Intrusion Set* represents the entire attack

infrastructure and capability package, which may persist over a much longer period and be reused across multiple Campaigns to achieve potentially different objectives.

Location

The Location SDO represents a geographic location, which may be described through any, some, or all of the following: a named region, a civic address, and/or a set of geographic coordinates (latitude and longitude). Locations are primarily used to give geographic context to other SDOs: for instance, to indicate where an Identity or Intrusion Set is based, or to associate a piece of Malware or an Attack Pattern with the geographic regions in which it targets victims.

Threat Actor

Threat Actors are individuals, groups, or organizations believed to be operating with malicious intent. A Threat Actor is conceptually distinct from an Intrusion Set: while it is not itself an Intrusion Set, it may support, direct, or be affiliated with one or more Intrusion Sets over time, and this affiliation may change as the actor's tooling, infrastructure, or objectives evolve.

Relationship

The Relationship object is used to link two SDOs or SCOs together in order to describe how they relate to each other. Unlike SDOs, a Relationship object does not describe an independent entity; rather, it describes a directed, typed connection between a source object and a target object.

2.4.6 STIX Meta Objects

STIX Meta Objects support the primary specification by defining how data markings and language markings are applied to SDOs, SCOs, and SROs.

TLP Marking Object Type

The TLP Marking Object Type defines how a Traffic Light Protocol (TLP) marking is represented within the `definition` property of a Marking Definition object. In this case, the `definition_type` property must be set to `tlp`, and the object defines a `tlp` property, which must take one of the following values:

- `white`
- `green`
- `amber`
- `red`

The TLP scheme allows producers to indicate the intended audience and sharing boundaries for the information contained in a STIX Object, ranging from unrestricted disclosure (`white`) to strictly limited disclosure (`red`).

Object Markings

Object Markings apply data markings to an entire STIX Object and to all of its content. They are specified as embedded relationships through the `object_marking_refs` property, which is an optional list of `id` properties referencing Marking Definition objects.

Granular Markings

Granular Markings allow both data markings and language markings to be applied to individual portions of a STIX Object or Marking Definition, rather than to the object as a whole. Granular markings are specified through the `granular_markings` property, which is a list of `granular-marking` instances.

The Granular Marking type defines how the Marking Definition object referenced by the `marking_ref` property, or alternatively a language specified by the `lang` property, applies to a specific set of content identified by the list of selectors given in the `selectors` property.

Properties (exactly one of `lang` or `marking_ref` must be present):

- `lang`
- `marking_ref`
- `selectors`

This mechanism is particularly useful when only part of an object's content is sensitive or subject to a specific handling restriction, allowing producers to avoid over-classifying an entire object when a coarser, object-level marking would be too restrictive.

2.4.7 STIX Bundle Object

A **Bundle** is a collection of arbitrary STIX Objects grouped together within a single container. It carries no semantic meaning of its own: the objects contained within a Bundle are not considered related to one another merely by virtue of being packaged together. Although a Bundle is not itself a STIX Object, it still makes use of the `type` and `id` Common Properties, in addition to the `objects` property, which contains the list of enclosed objects. Because a Bundle is not a STIX Object, it **must not** be the source or target of any relationship, whether common or explicitly defined.

2.4.8 STIX Vocabularies

STIX vocabularies whose type names end in `-ov` are *open*: they provide a listing of common, industry-accepted terms as guidance for the user, but do not restrict the user to that defined list. Open vocabularies are referenced from STIX Objects using the type `open-vocab`, accompanied by a statement indicating which specific vocabulary applies.

STIX vocabularies whose type names end in `-enum` are *closed*: the only valid values are those explicitly listed in the vocabulary. Closed vocabularies are referenced from STIX Objects using the type `enum`, accompanied by a statement indicating which specific enumeration must be used.

Attack Resource Level Vocabulary

Vocabulary Name: `attack-resource-level-ov`

The Attack Resource Level vocabulary is currently used in the following SDO(s):

- Intrusion Set
- Threat Actor

Attack Resource Level is an open vocabulary that captures the general level of resources that a threat actor, intrusion set, or campaign might have access to. It ranges from `individual`, a single person acting alone, to `government`, denoting access to the resources of a national government.

Vocabulary Values: `individual, club, contest, team, organization, government`

Encryption Algorithm Enumeration

Vocabulary Name: `encryption-algorithm-enum`

The Encryption Algorithm enumeration is currently used in the following SCO(s):

- File

This is a closed enumeration used to describe the encryption algorithm applied to the contents of a File Cyber-observable Object, so that consumers can correctly interpret or reproduce the encryption context of the file's binary contents.

Identity Class Vocabulary

Vocabulary Name: `identity-class-ov`

The Identity Class vocabulary is currently used in the following SDO(s):

- Identity

This vocabulary describes the type of entity that an Identity object represents: whether it denotes an organization, a group, an individual, or a class of entities.

Vocabulary Values: `individual, group, system, organization, class, unknown`

Industry Sector Vocabulary

Vocabulary Name: `industry-sector-ov`

The Industry Sector vocabulary is currently used in the following SDO(s):

- Identity

Industry Sector is an open vocabulary that describes industrial and commercial sectors. It is intended to be holistic: it has been derived from several existing classification schemes and is not restricted solely to sectors typically labelled as "critical infrastructure."

Vocabulary Values: `agriculture`, `aerospace`, `automotive`, `chemical`, `commercial`, `communications`, `construction`, `defense`, `education`, `energy`, `entertainment`, `financial-services`, `government (emergency-services, government-local, government-national, government-public-services, government-regional)`, `healthcare`, `hospitality-leisure`, `infrastructure (dams, nuclear, water)`, `insurance`, `manufacturing`, `mining`, `non-profit`, `pharmaceuticals`, `retail`, `technology`, `telecommunications`, `transportation`, `utilities`

Infrastructure Type Vocabulary

Vocabulary Name: `infrastructure-type-ov`

The Infrastructure Type vocabulary is currently used in the following SDO(s):

- Infrastructure

Infrastructure Type is an open vocabulary that describes the primary purpose of a given piece of adversary or victim infrastructure, such as a command-and-control server or a phishing site. As with other open vocabularies of this kind, an Infrastructure object may be assigned more than one type where its function is multi-purpose.

Vocabulary Values: `amplification`, `anonymization`, `botnet`, `command-and-control`, `exfiltration`, `hosting-malware`, `hosting-target-lists`, `phishing`, `reconnaissance`, `staging`, `unknown`

Report Type Vocabulary

Vocabulary Name: `report-type-ov`

The Report Type vocabulary is currently used in the following SDO(s):

- Report

Report Type is an open vocabulary used to describe the primary purpose or subject of a report. For example, a report that contains malware and indicators for that malware would typically be assigned a report type of `malware`, to reflect that the malware is the primary focus of the report. Report types are not mutually exclusive: a single report can simultaneously be classified as, for instance, both a malware report and a tool report. It is worth noting that the mere presence of an object of a given type within a report does not, by itself, justify assigning the corresponding report type; if an object is included only to provide supporting evidence or context for other objects, it need not be reflected in the report's type.

Vocabulary Values: `attack-pattern`, `campaign`, `identity`, `indicator`, `intrusion-set`, `malware`, `observed-data`, `threat-actor`, `threat-report`, `tool`, `vulnerability`

Threat Actor Type Vocabulary

Vocabulary Name: `threat-actor-type-ov`

The Threat Actor Type vocabulary is currently used in the following SDO(s):

- Threat Actor

Threat Actor Type is an open vocabulary used to describe what kind of threat actor an individual or group represents. For instance, some threat actors are competitors seeking to steal proprietary information, while others are activists acting in support of a social or political cause. Actor types are not mutually exclusive: a threat actor can, for example, be both a disgruntled insider and a spy.

Vocabulary Values: `activist`, `competitor`, `crime-syndicate`, `criminal`, `hacker`, `insider-accidental`, `insider-disgruntled`, `nation-state`, `sensationalist`, `spy`, `terrorist`, `unknown`

Chapter 3

Leonardo's Cyber Threat Intelligence Platform (CTIS / C-THINK)

3.1 Overview and positioning

Among the commercial platforms developed to support structured Cyber Threat Intelligence (CTI) operations, Leonardo S.p.A. offers a solution marketed as **C-THINK** [5], internally also referred to as **CTIS** (Cyber Threat Intelligence System). The platform is a tool intended for analyst teams working in high-confidentiality environments, such as national CERTs/CSIRTs, defense organizations, and critical infrastructure operators. CTIS is Leonardo's next-generation platform, designed to support analyst teams operating in high-confidential environments. Its stated purpose is not merely reactive detection but the anticipation of threats before they materialize, positioning it within the broader shift in the industry from indicator-based detection toward predictive, intelligence-led security By combining advanced automation and orchestration capabilities, it enables the prediction and prevention of the threats most likely to impact an organization, while supporting the adoption of proactive security measures.

This framing is consistent with the strategic rationale Leonardo gives for the product: contemporary threats are described as increasingly **hybrid, blending disinformation, cyberattacks and even elements of military activity**, often sustained over long periods and coordinated across multiple vectors Cyber threats are increasingly hybrid and complex, combining disinformation, cyberattacks, and even military operations-often conducted over extended periods and in a coordinated manner. In its marketing for the platform, Leonardo further emphasizes the growing convergence between state-sponsored actors and cybercriminal groups, who increasingly share tooling to conduct coordinated operations against strategic infrastructure State-sponsored actors increasingly collaborate with cybercrime groups, leveraging their advanced tools to launch simultaneous, coordinated attacks against strategic national infrastructures. This narrative situates CTIS/C-THINK within the "cognitive superiority" paradigm increasingly used in defense-sector cybersecurity marketing, where intelligence tooling is framed as a strategic asset

for national resilience rather than a purely operational IT product.

3.2 Architecture: the CTI lifecycle

CTIS is explicitly built around the classical Cyber Threat Intelligence lifecycle, structured into five phases: Plan & Direct, Collect, Process, Analyse, and Disseminate. CTIS is built on an end-to-end approach that spans the entire Cyber Threat Intelligence lifecycle. The platform collects data from both internal and external sources, processes and analyses them through advanced automation and orchestration, and finally disseminates actionable insights to analysts and security systems. This maps closely to the canonical intelligence cycle described in CTI literature (e.g., the models used by SANS or the U.S. intelligence community), as well as in Section 2.2.

Plan & Direct. The platform supports formal definition of intelligence requirements and requests for information (PIRs/RFIs), task assignment, and performance monitoring, intended to keep collection aligned with strategic priorities The process begins with the planning and coordination of intelligence activities. Analysts can define and monitor intelligence requirements and requests for information, ensuring that collection efforts are always aligned with operational and strategic priorities.

Collect. Data ingestion is centred on a knowledge base built on a highly configurable ontology that extends and customizes the STIX 2.1 standard. External threat intelligence feeds, internal cybersecurity data, and user-supplied inputs are automatically integrated into this knowledge base. Interoperability is achieved through native support for open CTI formats and standards (including MISP, STIX, SIGMA, and YARA) alongside REST APIs for third-party integration and automated entity de-duplication.

Process. Processing relies on a knowledge-graph data model with a configurable, event-based enrichment engine. This enables advanced data enrichment with an innovative event-based approach and a comprehensive library of processing tools. Notably, access control follows an explicit need-to-know / need-to-share model, which is a governance principle borrowed directly from intelligence-community practice, ensuring both effective collaboration and strict confidentiality.

Analyse. The analysis layer offers a graph-based visual interface as well as programmatic (REST API) access, supported by a search engine for correlating data across large datasets. A powerful search engine supports complex queries, helping analysts quickly identify and correlate relevant data across vast datasets. CTIS also provides a dedicated workspace for structured threat analysis, where entities, evidence, and findings can be organized into comprehensive cases. Collaboration is strengthened through an advanced editor that enables analyst to work together, streamlining the creation of accurate and consistent intelligence reports.

Disseminate. Output is dual-channel: human-facing (role-tailored notifications and reports for analysts, managers, and auditors) and machine-to-machine (structured feeds pushed to SIEM, firewalls, and EDR systems). This dual approach human-centric reporting and machine-to-machine integration ensures that intelligence is not only generated but also made immediately actionable across the organization.

3.3 Analytical frameworks and domain specialization

A notable feature relevant to any discussion of methodological maturity is that CTIS does not rely solely on generic threat-modelling standards. It supports the **Diamond Model**, **STIX 2.1**, **MITRE ATT&CK**, and **MISP** for general-purpose analysis, which ensures methodological rigour and interoperability across the cybersecurity community, but it also integrates sector-specific frameworks for two specialized domains: space systems (via SPACE-SHIELD, developed under the European Space Agency, and SPARTA, developed by the Aerospace Corporation) and disinformation analysis (via DISARM). SPACE-SHIELD and SPARTA strengthen resilience against cyber threats targeting space infrastructures, while DISARM supports the analysis and management of risks linked to disinformation campaigns. This is a distinctive design choice that reflects Leonardo's positioning as a defense and aerospace group, extending CTI tooling beyond conventional IT/OT threats into space-domain and information-warfare threat modelling, an area still underdeveloped in most commercial CTI platforms.

3.4 Expansions to the STIX Standard

As mentioned before, the CTIS platform expands upon the STIX 2.1 standard by adding custom entities and relationships not originally defined in the standard. While many such extensions are present in the platform, our analysis will be limited to those used by the feeder under discussion. These custom entities are identified by the prefix `x-`.

3.4.1 Operation

The first and most important entity is `x-operation`, the central entity used to represent EuRepoC incidents and the node from which every relationship in the model originates. Owing to the flexibility of custom STIX entities and the freedom afforded by labels, most of the feeder's original attributes that lacked a direct mapping to existing STIX entities were instead encoded as labels on `x-operation`. This design choice allowed the mapping to preserve the granularity of the EuRepoC dataset without requiring the creation of numerous additional custom entities, at the cost of moving some semantically distinct information into a less strictly typed representation. The specific labels and relationships associated with this entity are described in detail in the chapter dedicated to the feeder description.

3.4.2 News

Another fundamental entity is `x-news`. Although the STIX `news` object type was originally conceived to track individual news items, within this feeder `x-news` is repurposed as a structural counterpart to `x-operation`: it mirrors the operation entity's central role in the model, but is populated with a distinct set of labels and relationships tailored to information that does not fit naturally under an operation. In particular, `x-news` is used to capture information not natively supported by the STIX standard, such as *attributions* (as conceptualized by the EuRepoC consortium), *legal responses*, and *political responses*,

categories considered too narrow in scope, individually, to justify dedicated custom STIX entities. This choice was also motivated by architectural constraints: `CTIS` operates as a federated system, meaning that introducing new custom STIX entities would need to be propagated to and registered within the `CTIS` instance of every connected client in order to preserve cross-client compatibility. Encoding these categories as labels on an existing entity, by contrast, avoided this synchronization overhead, since labels do not require any schema-level changes to be understood across the federation. As with `x-operation`, the specific labels and relationships used for `x-news` are described in the corresponding section of the feeder description chapter.

3.5 Integration with LENS (EDR)

CTIS/C-THINK is designed for bidirectional integration with **LENS**, Leonardo's Endpoint Detection & Response platform covering both IT and OT environments. LENS collects telemetry from agents and probes deployed across converged IT/ICS infrastructures, including legacy environments, ensuring comprehensive protection across both legacy and advanced digital systems. The platform combines automated anomaly detection with a powerful query language, allowing analysts to filter billions of events and focus on relevant evidence with speed and precision. Detection and response rules can be adapted dynamically at runtime, enabling targeted monitoring and remediation both locally and centrally. Integration with CTIS ensures mutual enrichment of their knowledge bases, strengthening situational awareness across the entire security ecosystem. This is a good example of the "detect-to-intelligence feedback loop" architectural pattern, distinct from platforms that treat CTI and EDR as fully separate product lines.

3.6 Deployment models

Reflecting the diversity of its target customers both national agencies and private-sector operators, the platform is offered in three deployment modes: on-premises, SaaS, and fully managed service depending on customer requirements. This flexibility is significant for sovereignty-sensitive customers (e.g., government or defense clients who may require on-premises deployment for data residency) versus commercial clients seeking lower operational overhead via SaaS or managed service.

3.7 Vendor-Listed Capabilities

CTIS's vendor-listed capabilities can be summarized as follows:

- **Native integration with Leonardo's Global Cybersecurity Center**, through synchronization across CTIS instances, providing updated and qualified threat intelligence.
- **Direct connectivity to configurable MISP servers**, enabling the sharing and receipt of structured threat information.

- **Advanced situational awareness** through OSINT and SOCMINT integration, expanding visibility on emerging threats.
- **Customizable threat data modelling**, supporting either standard entities (e.g., STIX) or custom entities defined with specific attributes, relationships, rules, and constraints.
- **Graph-based visual query building**, allowing analysts to define nodes, relationships, and semantic properties through a visual interface.
- **Structured analysis based on standard models** (Diamond Model, STIX, MITRE ATT&CK) **or domain-specific models** (e.g., SPARTA and SPACE-SHIELD for space systems, DISARM for disinformation).
- **Collaborative analyst workflows**, supported by shared reports and dedicated plug-ins for inserting evidence, entities, and correlations.
- **Interoperability with LENS and the broader Leonardo cybersecurity ecosystem**, enabling seamless integration of tools and functions.
- **Flexible deployment models**, SaaS, on-premises, or fully managed service, to meet diverse infrastructure and organizational requirements.

Chapter 4

The Feeder Description

4.1 Basic Information

The feeder was designed to ingest the entire EuRepoC database during its first run and, for subsequent runs, to only add new operations or modify existing ones as updated by the consortium. In particular, to keep track of the different operations, the `Id` field within the EuRepoC dataset was used, ingested as an `x-external-uuid` of the `x-operation` entity in CTIS. This design choice ensures that even when key fields of the entities are changed, the script is able to update all the modifiable fields of the original entity instead of creating new ones leading to possible duplications .

To check for updated operations, EuRepoC conveniently maintains an `updated_at` field for each entry. Relying on this field allows the feeder to detect changes efficiently, by comparing the date to the `last_run` date stored in a separate file and updated after every successful run. This significantly reduces the computational overhead of subsequent runs, as the feeder does not need to reprocess the entire dataset each time, but only the subset of operations that have actually changed since the previous execution.

According to EuRepoC's own website, they advertise a daily newsletter sent out around 17:30 containing the newly added operations; for this reason, 18:00 was chosen as the time to run the script, ensuring that any operations added that day would already be available. The feeder runs once a day via a simple `Crontab` entry that triggers a bash script (`run.sh`), which in turn builds the container and runs the Python application.

4.2 Technologies Involved

4.2.1 Python

The language chosen for the implementation of the feeder was Python. This choice was motivated by several factors.

The primary reason was the availability of the **CTIS API Library**, a Python library that enabled straightforward interaction with the TIP (Threat Intelligence Platform), significantly reducing the development effort required to integrate with the platform.

Additional reasons stem from the characteristics of the language itself. Python offers

a rich ecosystem of libraries for I/O operations and data parsing, such as `pandas` for handling structured data and `lxml`/`BeautifulSoup` for parsing HTML and XML content. This extensive library support removes the need to implement low-level parsing logic from scratch, allowing development to focus on the feeder's core logic rather than on data-format handling.

Python's dynamic typing and concise syntax also enable fast development and rapid iteration, which proved valuable given that the requirements and data sources for the feeder evolved throughout the project.

Finally, Python's easy integration with data science and machine learning tooling was another significant factor, as it allowed the feeder to be seamlessly connected to downstream analysis components without the overhead of serializing and transferring data between different technology stacks.

4.2.2 Docker

The feeder was designed to run within a containerized application using **Docker**. This architectural choice was motivated by several considerations relevant to both development and deployment.

First, containerization ensures reproducibility across different execution environments. By encapsulating the application along with its dependencies, runtime, and configuration into a single Docker image, the feeder can be migrated between machines, whether virtual or physical, without the risk of environment-specific inconsistencies, commonly referred to as the "it works on my machine" problem. This is particularly important in a research or production setting, where the same script may need to be executed on development workstations, testing servers, and production infrastructure, each potentially running different operating systems or software versions.

Second, Docker was chosen to provide process and filesystem isolation, ensuring that the feeder operates within a secure, sandboxed environment separate from the rest of the production system. This isolation reduces the attack surface of the host machine, limits the potential impact of software vulnerabilities or misconfigurations within the feeder itself, and prevents unintended interference with other services running on the same infrastructure. Additionally, containerization simplifies dependency management, as all required libraries and packages are defined explicitly within the Dockerfile, avoiding conflicts with other applications that may rely on different versions of the same dependencies.

In particular, the base image used for building the Docker container was `ctis_api_client`, an image commonly used by the CTI Team to develop projects that rely on the CTIS API Library. The use of a standardized base image further contributes to consistency across projects developed within the team, as it guarantees that all dependent applications share a common set of preinstalled libraries, configurations, and tooling, reducing setup overhead and minimizing the likelihood of version mismatches between projects.

4.2.3 NocoDB

NocoDB is a database management platform that transforms relational databases into a spreadsheet-like interface, allowing users to interact with structured data through familiar operations such as sorting, filtering, grouping, and sharing, without requiring direct knowledge of SQL. Unlike traditional no-code platforms that manage their own proprietary data store, NocoDB connects directly to existing relational databases, including MySQL, PostgreSQL, SQLite, Microsoft SQL Server, and MariaDB, and exposes them through a graphical interface while preserving the underlying schema and data.

NocoDB was used in this project as a way to store **Intrusion Sets (IS)** or **Threat Actors (TA)** that could not be found on CTIS at ingestion time. This could be due to various reasons: sometimes the IS/TA was not yet tracked on the platform, and sometimes it was tracked under a different alias. Even though CTIS supports the use of aliases, not every IS/TA on it had the alias used by every security vendor's framework. This design choice was made to avoid populating the platform with incorrect IS/TA or with duplicates of entities that already existed under different names.

In particular, the workflow around NocoDB consists of the following steps:

1. The EuRepoC incident is processed and the `Initiator` field is used to identify the associated Intrusion Set (IS) or Threat Actor (TA).
2. The initiator's name is used to query CTIS, searching among the IS and TA already stored on the platform.
3. If a matching entity is found, it is related to the operation, and no NocoDB record is created.
4. If no matching entity is found, a row is created in NocoDB for an analyst to manually determine whether it constitutes an IS or a TA, so that it can later be added to CTIS and related to the operation.

4.3 The Feeder Structure

The file structure for the application was based on other similar projects developed by the CTI Team, following an internal convention adopted to ensure consistency, maintainability, and ease of onboarding across the team's codebase. The structure consists of the following:

- `logs` directory for log files, such as `last_run.txt`, which stores the timestamp of the most recent successful execution and is used by `main.py` to determine which records require re-evaluation on subsequent runs.
- `config` directory for the `config.yaml` file, which contains project information necessary for the correct functioning of the application. Centralizing this information in a single configuration file, rather than hardcoding it within the source code, allows the application's behaviour to be adjusted without modifying the codebase itself, and prevents sensitive or environment-specific values from being scattered across

multiple files. For example, the file contains the contents of headers and payloads for requests to the EuRepoC website, and Base/Table IDs for NocoDB.

- `src` directory, which contains the files for the source code, organized into distinct modules according to their responsibility:
  - `main.py`, containing the main function that handles high-level checks, such as last-run and `updated_at` comparisons, and calls the methods from `functions.py` for each line in the dataset. This file acts as the entry point of the application and orchestrates the overall execution flow.
  - `functions.py`, containing methods to download and pre-process the EuRepoC dataset, and to extract the individual CTIS entities. Isolating this logic from `main.py` allows the core data-processing functions to be tested and reused independently of the application's control flow.
  - `logger.py`, containing logging methods used throughout the application to record execution details, warnings, and errors, supporting easier debugging and auditing of past runs.
  - `notification.py`, containing methods to send success or failure notifications to a Mattermost channel, allowing the team to be promptly informed of the feeder's execution status without needing to manually inspect the logs after every run.
  - `wrappers.py`, containing method decorators to handle exceptions and log execution times.
  - `cti_api_library`, a submodule containing the CTIS API library, which abstracts the interaction with the CTIS platform and exposes a set of higher-level methods for creating, retrieving, and updating entities on the platform.
  - `nocodb_api_library`, a submodule containing the NocoDB API library, used to programmatically access the EuRepoC table containing the IS/TA entities, enabling the feeder to create and query records without direct interaction with NocoDB's REST API.
- `docker-compose.yml`, containing the information necessary to build and run the Docker container, including the base image, volume mounts, and environment variables required by the application.
- `requirements.txt`, listing the Python packages required by the application, which are installed within the container during the image build process, ensuring that the application has access to all necessary dependencies regardless of the host machine's local Python environment.
- `run.sh`, the bash script invoked by the `Crontab` entry to schedule the feeder's execution. The script relies on a mutex file, created when the container starts and removed once execution finishes, to check whether an instance of the feeder is already running. If the mutex file is present, `run.sh` does not start a new instance; otherwise, it creates one using the `docker compose up` command. This mechanism

prevents overlapping executions, which could otherwise occur if a previous run takes longer than expected or fails to terminate correctly.

4.4 Feeder Execution Flow

4.4.1 Dataset Download

Although access to EuRepoC's API was not granted, the dataset remains freely available for download through their website. By issuing HTTP requests with Python's `requests` library, and by inspecting the requests generated by the website through the browser's network tab, I was able to identify the correct headers and payload structure required to trigger the dataset download programmatically, after some trial and error.

The file is downloaded in **Excel** format and saved locally as `eurepoc-database.xlsx`. It is subsequently loaded into a `pandas` `DataFrame` to facilitate further pre-processing and analysis.

4.4.2 Dataset Pre-processing

The methodology used by EuRepoC to encode information within each cell is described in detail in their codebook, which explicitly specifies the characters used as delimiters. These delimiters are the hyphen (`'-'`) and the semicolon (`';'`), and they are the ones relied upon throughout the pre-processing stage, as well as in the definition of the individual entities that will subsequently be created within CTIS.

Although the original dataset on EuRepoC's servers is stored in `csv` format, only an `xlsx` export is offered for download. This introduces a limitation, since the Excel format imposes a maximum cell content length of **32,767** characters. Given the way the EuRepoC consortium encodes multiple attributions and multiple aliases for the same intrusion set within a single cell, this limit is occasionally exceeded, resulting in truncated and therefore lost information.

To address this issue, I first identify the fields in which the character limit has been exceeded. For these fields, I verify that the number of discrete values recoverable from the (truncated) cell is consistent with the number of values present in other, related fields of the same row, which are not subject to truncation. This cross-referencing allows me to detect where truncation has occurred even when the cell content itself no longer reflects the full original data.

A second, related issue arises from what appears to be an artifact of the `csv`-to-`xlsx` conversion performed by EuRepoC prior to publishing the download: certain cells contain a literal hyphen (`'-'`) as part of their actual value, even though this same character is conventionally used elsewhere in the dataset as a delimiter separating multiple values within a cell. This ambiguity is particularly problematic in the `receiver_name` field, where entries are expected to be hyphen-separated lists of names, but where a literal hyphen occurring inside a single name cannot be distinguished from a delimiter using string parsing alone.

To detect this inconsistency, I compare the number of values obtained by splitting the `receiver_name` cell on the hyphen delimiter against the number of values present in

the corresponding `receiver_category` cell, which draws from a fixed, predefined set of categories and therefore never contains a literal hyphen. Any mismatch between these two counts indicates that at least one hyphen in `receiver_name` is part of an actual name rather than a separator.

Because the correct receiver name cannot be reliably reconstructed from the `xlsx` file alone once this ambiguity arises, I resolve each flagged case individually by querying EuRepoC's website directly. Specifically, I issue two sequential HTTP requests: the first retrieves metadata needed to locate the specific incident page, and the second, parametrized using information extracted from the first, retrieves the full HTML content of the incident's dedicated page, from which the correct, unambiguous set of receiver names is then extracted.

4.4.3 Entities Extraction

Once pre-processing is complete, the DataFrame is ready to be processed. For each row, the `updated_at` field is compared against the `last_run_date` value extracted from the log file, in order to determine whether that row requires processing or can be skipped. This comparison ensures that only incidents that are new or have been modified since the previous execution are processed, avoiding redundant work and reducing the feeder's overall runtime.

If a row does require processing, the feeder begins extracting values from the relevant cells in order to construct the actual entities that will later be inserted into the TIP. As mentioned previously, due to the structure of the dataset, multiple logically distinct elements from the same field belonging to the same incident is encoded within a single cell, separated by delimiter characters that vary depending on the field in question.

While EuRepoC explicitly advises against attempting to decouple these fields manually, recommending instead the use of their already-normalized datasets (**Incidents**, **Initiators**, and **Attributions**), these normalized datasets are updated only infrequently, approximately once a year. This update frequency makes them unsuitable as a source for a feeder that is required to ingest fresh information on a daily basis.

For this reason, I defined a set of dedicated methods within `functions.py`, each responsible for decoupling the information encoded in a specific field according to its particular delimiter structure and semantics. These methods follow the naming convention `define_<`*`field_name`*`>`, and their implementation will be discussed in greater detail later in this chapter.

4.4.4 Global Mappings and Sentinel Values

Throughout the feeder, several pieces of information extracted from EuRepoC need to be translated into the vocabulary used by CTIS before they can be represented as STIX entities. Rather than being resolved ad-hoc within each extraction method, these translations are centralized in a small set of global dictionaries, defined once at the top of the module and reused wherever the relevant EuRepoC field appears. This design avoids duplicating mapping logic across methods and ensures that a given EuRepoC value is

always translated to the same CTIS value, regardless of which entity or relationship it contributes to.

Examples of such dictionaries include `metrics_mapping`, `disclosure_mapping`, and `attribution_mapping`, each translating a specific EuRepoC field into its corresponding CTIS representation. `sector_mapping`, described later in this chapter, follows the same pattern for the sector information associated with Receivers. All of these dictionaries share the same lookup-with-fallback logic: if an EuRepoC value is present as a key, the corresponding CTIS value is used; otherwise, the original value is preserved, so that no information is silently lost when a mapping has not yet been defined.

Alongside these mappings, the module also defines `missing`: a list of sentinel values representing data that EuRepoC records as unavailable, non-informative, or otherwise not meaningful (e.g. empty strings, or placeholder tokens such as `"nan"` used when a field is not populated in the source dataset). Because this condition can arise in any field extracted from a row, `missing` is defined once globally and checked consistently across extraction methods, rather than being redefined or hard-coded separately in each one. This ensures that the criteria for discarding a value as "not meaningful" remain uniform throughout the feeder.

4.4.5 CTIS Entities Definition

As briefly mentioned before, the entire `Incident` object described by EuRepoC is mapped onto CTIS through a set of distinct entities, all of which are connected to the central `x-operation` entity. A visual representation of the relationships and entities involved can be seen in Figure 4.1.

Operation

The first CTIS entity created is the `x-operation` entity, the central entity that identifies the incident and from which every relationship originates.

The entity itself uses only a small set of basic fields (`name`, `description`, `first_seen date`, and `last_seen date`), all of which are explicitly present in the EuRepoC dataset. Most of the “non-standard” fields are instead attached to the entity in the form of labels and parameters, generated through two dedicated methods from the `functions.py` module: `define_operation_labels` and `define_operation_parameters`.

```
1     operation = Operation(
2                     row.name,
3                     row.description,
4                     first_seen=datetime.strptime(row.start_date, '%Y-%m-%d') if
      isinstance(row.start_date, str) else None,
5                     last_seen=datetime.strptime(row.end_date, '%Y-%m-%d') if
      isinstance(row.end_date, str) and row.end_date != "Not available" else None,
6                     labels=func.define_operation_labels(row),
7                     parameters=func.define_operation_parameters(row, url),
8                     xsources=["eurepoc_feeder"]
9                 )
10
```

Listing 4.1. Operation entity definition

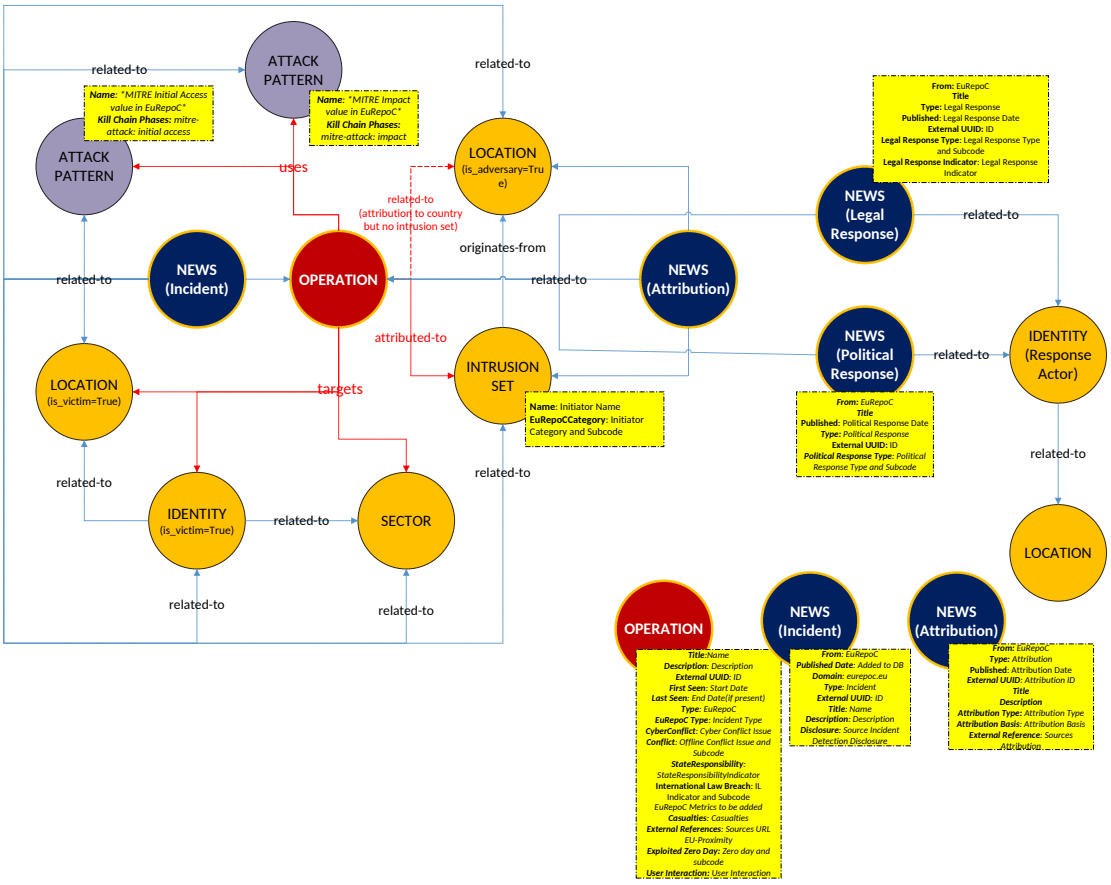


Figure 4.1. EuRepoC CTIS Entities Schema

The first method, `define_operation_labels`, processes and attaches the following labels to the entity by mapping the corresponding EuRepoC fields:

- `Type:EuRepoC`, a default label added to every entity regardless of its content
- `CyberConflictIssue`, mapping the **Cyber Conflict Issue** field (if defined and available)
- `Conflict`, mapping the **Offline Conflict Issue** field (if defined and available)
- `IL Breach`, mapping the **International Law Breach Indicator** field (if defined and available)
- `State Responsibility`, mapping the **State Responsibility Indicator** field (if defined and available)
- `Casualties`, mapping the **Casualties** field (if defined and available)
- `ExploitedZeroDays`, mapping the **Zero Days** and **Zero Days Subcode** fields (if defined and available)
- `RequiresUserInteraction`, mapping the **User Interaction** field whenever its value is `"Required"`
- `LegalResponseIndicator`, mapping the **Legal Response Indicator** field (if defined and available)
- All fields related to the **Cyber Intensity Metrics**, provided that a score is present and different from zero:
  - `EuRepoCCyberIntensity(Weighted)`, mapping the **Weighted** and **Unweighted Cyber Intensity** fields
  - `LevelOfDisruption`, mapping the **Disruption** field
  - `LevelOfDataTheft`, mapping the **Data Theft** field
  - `LevelOfHijacking`, mapping the **Hijacking** field
  - `LevelOfSpatialEffects`, mapping the **Physical Effects Spatial** field
  - `LevelOfTemporalEffects`, mapping the **Physical Effects Temporal** field
  - `TargetPoliticalImportance`, mapping the **Physical Effects Spatial** field, also used to compute the weight of the intensity score
- All fields related to the **Impact Metrics**, provided that a score is present and greater than zero:
  - `EuRepoCImpactIndicator`, mapping the **Impact Indicator** field
  - `FunctionalImpact`, mapping the **Functional Impact** field
  - `IntelligenceImpact`, mapping the **Intelligence Impact** field
  - `NumberOfAffectedEntities`, mapping the **Impact Affected Entities** field

- `NumberOfAffectedThirdCountries`, mapping the **Impact Third Countries** field
- `EconomicImpact`, mapping the **Economic Impact Exact Value** and **Currency** fields

The second method, `define_operation_parameters`, is instead used to populate the entity's `x-external_uuid` field (using the EuRepoC incident ID) and its `x-external_references` field, which stores the **Sources** provided by EuRepoC together with the URL of the incident's page on the EuRepoC website.

News (Incident)

The next entity created is an `x-news` entity that represents a sort of mirroring of the `x-operation` entity. This choice was made for reasons related to dissemination: the way reports are created on CTIS relies on collecting `x-news` entities programmatically from the platform, which is done through the use of **Advanced Queries** that load `x-news` entities. To make the EuRepoC incidents "compatible" with these queries, this newly created entity is then related to the **Operation** entity and other main entities.

```
    news_incident = News(
                row.description, # preview
                row.name, # title
                row.description, # description
                "", # author username
                row.added_to_DB + " 00:00:00Z", # published
                domain="eurepoc.eu",
                labels=func.define_newsIncident_labels(row),
                parameters={"x-external_uuid": [str(row.ID)],
                            "x-external_references": [url + "?cyber_incident=" +
    str(row.ID)]},
                xsources=["eurepoc_feeder"]
            )

```

Listing 4.2. News (Incident) entity definition

As visible in Listing 4.2, some EuRepoC fields such as **Name**, **Description**, **Added to DB** (as the **Published Date** of the **News**), and **ID** are used very straightforwardly. In this case, `x-external_uuid` and `x-external_references` do not rely on a separate method, since the EuRepoC sources are already stated in the `x-operation` entity. A method for attaching labels is still used, however: `define_newsIncident_labels`, which creates two labels:

- `Type:Incident`, a standard label attached to every entity of this kind to distinguish the type of `x-news` entity
- `Disclosure`, mapping the **Source Incident Detection Disclosure** field, since this label was already commonly used for other `x-news` entities in this way

After creation, the `x-news` entity is then related to the `x-operation` entity via a `related_to` relationship.

Initiators

This section of the feeder deals with what EuRepoC identifies as the **Initiators**, the **Intrusion Sets/Threat Actors** responsible for the Incident. The mapping to CTIS uses the `intrusion-set` and `threat_actor` entities.

The `define_initiators` method (visible in 4.3) extracts the EuRepoC `names`, `countries`, and `categories` of each Initiator in order to map them to `Intrusion Sets` or `Threat Actors`, their Labels, and their `locations` on CTIS.

```
1     # Extract the Initiator(s) fields from a single row
2     def define_initiators(single_row):
3         initiators = set()
4
5         # No field is always present, so we take the maximum length of the lists
      to define the number of initiators, in case of fields with nan the list will
      be filled with empty strings
6         names = [item.strip() for item in single_row.initiator_name.split(';')] if
       isinstance(single_row.initiator_name, str) else [""] # Get the names, if nan
      get empty string
7         countries = [item.strip() for item in single_row.initiator_country.split('
      ;')] if isinstance(single_row.initiator_country, str) else [""] # Countries
8         categories = [item.strip() for item in single_row.initiator_category.split
      (';')] if isinstance(single_row.initiator_category, str) else [""] #
      Categories
9         category_subcodes = [item.strip() for item in single_row.
      initiator_category_subcode.split(';')] if isinstance(single_row.
      initiator_category_subcode, str) else [""] # Category Subcodes
10        length = max(len(names), len(countries), len(categories), len(
      category_subcodes))
11        lists = [names, countries, categories, category_subcodes]
12
13        for l in lists:
14            if len(l) < length:
15                l.extend([""] * (length - len(l))) # Adjust the length of the
      lists where no values are present
16
17        keys = ['name', 'country', 'category', 'subcode']
18        for i in range(length):
19            # Create a dictionary using keys and values for each initiator
20            combination_dict = {keys[index]: lists[index][i] if lists[index][i]
      else "" for index in range(len(keys))}
21            initiators.add(frozenset(combination_dict.items()))  # Frozenset to
      make it hashable for the set
22
23        return initiators
```

Listing 4.3. `define_initiator` method

The first item searched on CTIS is the `location`: the country name is normalized using the **pycountry** library and the normalized name is then used to query CTIS. This entity is related to the `x-operation` via an `originates-from` relationship, and to the `x-news` via a `related-to` relationship. Normalizing the country name before querying is necessary because EuRepoC and CTIS do not necessarily share a consistent naming convention for countries (e.g. official names, ISO codes, or colloquial forms), and **pycountry** provides a reliable way of reconciling these differences and reducing the risk of failed or duplicate lookups.

After the location, the Initiator name is used to query CTIS. The query is executed first on the Intrusion Sets' names, then on their aliases, then on the Threat Actors' names, and finally on their aliases. If nothing is found, the entity is not automatically

created. This design choice was made for two main reasons: first, to avoid creating duplicate entities in cases where names are spelled slightly differently, or where an alias is not yet present on CTIS; and second, and more importantly, to avoid mistakenly creating an `intrusion-set` entity for what is in fact a **Threat Actor**, or vice-versa, since EuRepoC does not operate this specific distinction in its own data model. Automating this classification step would therefore risk introducing systematic misclassifications into CTIS, which is why the task is deliberately left to human judgement (see below).

In case an Intrusion Set or Threat Actor *is* found, the categories extracted from the dataset are appended to the entity in the form of labels by a dedicated method, `define_initiator_labels`, which defines and attaches the `EuRepoCCategory:`*`category`* and `EuRepoCSubCategory:`*`subcode`* labels. These labels preserve EuRepoC's own categorization of the Initiator alongside the CTIS entity, allowing analysts to cross-reference the two taxonomies without losing information in the mapping process. The located entity is then related to the `x-operation` entity via an `attributed-to` relationship, and to the `x-news` via a `related-to` relationship

In case EuRepoC does not provide any Initiator name but only the country, the country is still used to track the origin of the operation via the `originates-from` relationship.

When the Initiator's name is present but no equivalent Intrusion Set/Threat Actor is found on CTIS, the feeder creates a **NocoDB** record (example visible in 4.2) containing all the information related to the Initiator. This record includes a checkbox that allows an analyst to indicate whether the Initiator should be considered an Intrusion Set or a Threat Actor, and to use the information contained in the record to manually create the corresponding entity on CTIS. This human-in-the-loop step ensures that the ambiguity inherent in EuRepoC's data which, as noted above, does not distinguish between the two entity types, is resolved by an analyst rather than by an automated heuristic, preserving the quality and consistency of the CTIS taxonomy. The record uses the combination of the Operation and Initiator names as its identifier, which allows for multiple "Unknown" initiators to be tracked separately even when they belong to the same Operation.

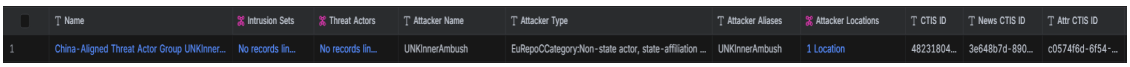


Figure 4.2. Example of NocoDB row

Attributions

Attributions probably represent the most complex information to be extracted from EuRepoC, due to the number of fields which compose them. As for all main entities, a dedicated method is used: `define_attributions`. It extracts the following information:

- **Attribution ID**: the Attribution ID given by EuRepoC, used as an `x-external_uuid` in the `x-news` (attribution) entity.
- **Attribution Date**: the date of the Attribution, mapped as the `published_date` of the `x-news`.

- **Attribution Type**: the category of the Attribution as defined by EuRepoC, mapped as a label of the `x-news` entity using `define_newsAttribution_labels`.
- **Attribution Basis**: the basis of the Attribution, also mapped as a label of the `x-news` entity using `define_newsAttribution_labels`.
- **Attributing Actor**: the actor performing the attribution, mapped to an `identity` entity.
- **Attributing Actor IT Company**: the IT company (if present) of the attributing actor. Currently processed but not mapped to a CTIS entity.
- **Attributing Country**: the country of the Attributing Actor.
- **Attributed Initiator**: the name of the Attributed Initiator, mapped to an `intrusion-set` or `threat-actor` entity in the same way as the Initiator object.
- **Attributed Initiator Country**: the country of the Attributed Initiator, mapped to a `location` entity.
- **Attributed Initiator Category**: the categories of the Attributed Initiator, mapped as labels of the `intrusion-set` or `threat-actor` entity in the same way as the Initiator object.

Several of these fields (the Attribution Type, Basis, and the Attributed Initiator's category) mirror handling logic already introduced for the Initiator object. This is not incidental: Attributions can be understood as a specialization of the same underlying problem, namely reconciling an external actor claim (whether of the incident's origin or of who attributed it) against CTIS's existing taxonomy of Intrusion Sets and Threat Actors.

The extracted information is used to create an `x-news` entity (4.4) which acts as a stand-in for an attribution entity, since this specific entity type is absent from both CTIS and the STIX 2.1 standard. This is a deliberate modelling choice: rather than extending CTIS's data model with a bespoke object, which would require custom support throughout the platform and break interoperability with the STIX 2.1 standard, the feeder repurposes an existing, semantically adjacent entity (`x-news`) and disambiguates it through labeling. The `define_newsAttribution_labels` method is used to attach the **Attribution Type**, the **Attribution Basis**, and the `Type:Attribution` labels. This last one, in particular, is used to differentiate the `x-news` entity from other `x-news` entities created elsewhere in the feeder (for instance, from the News object discussed in Section 4.4.5), ensuring that attributions remain distinguishable and queryable as a coherent subset of the CTIS `x-news` entities.

```
    news_attribution = News(
        description, # preview
        f"Attribution {id} of Operation: {row.name}", # title
        description, # description
        "", # author username
        datetime.strptime(date, "%Y-%m-%d %H:%M:%S") if date not in missing else "
    ", # published which corresponds to the attribution date or in case it is
    absent to ""
```

```
        domain="eurepoc.eu",
        labels=func.define_newsAttribution_labels(attr_dict),
        parameters={"x-external_uuid": [str(id)],
        "x-external_references": ([item.strip() for item in sources.split("; ") if item.strip()] + [url + "?cyber_incident=" + str(row.ID)])},
        xsources=["eurepoc_feeder"]
    )
```

Listing 4.4. `news_attribution` definition

After the entity's creation, it is immediately linked to the `x-operation` entity via the `related-to` relationship, thereby preserving the connection between a given attribution and the incident it refers to. The **Attributed Initiator Country** is used to find the `location` entity that must, in turn, be related to the `x-news`.

The processing of the **Attributed Initiator** object follows the same model as the **Initiator** object described in the previous section: a corresponding `intrusion-set` or `threat-actor` entity is searched for on CTIS. If the entity is found, the categories defined by EuRepoC are attached to it as labels, and the entity is linked to the `x-news` via the `related-to` relationship. If it is not found, the same NocoDB record-creation procedure described for the Initiator object is followed, allowing an analyst to subsequently determine whether the unmatched Attributed Initiator should be created on CTIS as an **Intrusion Set** or a **Threat Actor**.

Legal Responses

Legal Responses (together with **Political Responses**) are entities not usually tracked by CTI sources, which makes them fairly unique in the context of a CTI feeder. Because of this uniqueness, the STIX standard does not include a dedicated entity to represent them appropriately, so, as with **Attributions**, the `x-news` entity is used instead. **Legal Responses** are simpler entities compared to **Attributions**, as they only contain the identity of the actor performing the response, its country of origin, the date of the response, and its type.

The `define_legalResponses` function extracts the name and country of the actor performing the response, along with the response date and type. The `x-news` entity is then defined as shown in 4.5, with the `description` field composed as follows:

```
description = f"Legal Response to Incident {row.ID} of {actor if actor not in missing else "} ({country if country not in missing else "}) {'on ' + date if date not in missing else "}"
```

```
    news_legalResponse = News(
        description, # preview
        description, # title
        description, # description
        "", #author username
        date_datetime if date not in missing else "", # published date which corresponds to the legal date or, if absent, to ""
        domain="eurepoc.eu",
        labels=func.define_newsLegalResponse_labels(legalResp_dict),
        parameters={"x-external_uuid": [str(row.ID)], "x-external_references": [url + "?cyber_incident=" + (row.ID)]},
        xsources=["eurepoc_feeder"]
        )
```

Listing 4.5. `news_legalResponse` definition

Notably, the `description` field is reused for the `preview`, `title`, and `description` attributes of the entity, since Legal Responses do not carry enough distinct information to justify separate, more detailed text for each field. This mirrors the approach taken for Attributions and keeps the entity generation logic consistent across `x-news` types.

The `define_newsLegalResponse_labels` function is instead used to attach four labels:

- the `Type:LegalResponse` label, used to distinguish this `x-news` entity from the other types (Incident and Attribution);
- the `LegalResponseType` label, which maps the type of legal retaliation;
- the `LegalResponseSubcode` label, which maps a subcategory of the `LegalResponseType`;
- the `LegalResponseIndicator` label, which is already mapped as an `x-operation` label but is added here for redundancy and because of its relevance to the Legal Response object.

After creation, the `x-news` entity is related to the corresponding `x-operation`, and the extracted actor's name is used to look for, or create, an `identity` entity, which is then related to the `x-news`. As with other entities, the actor's country is used to look for a `location` entity on **CTIS** and relate it to the `identity` entity. One limitation of the **STIX** standard implementation on **CTIS** is that, when dealing with `x-news` entities, only `related-to` relationships are allowed, this restricts the semantic granularity with which the pipeline can express, for example, the specific nature of the link between a legal response and the actor who issued it.

Political Responses

As mentioned in 4.4.5, **Political Responses** are also fairly unique in the context of CTI feeders, and their structure closely mirrors that of **Legal Responses**. In fact, the information extraction logic is almost identical to that of **Legal Responses**, with the `x-news` entity differing only in the `name`, `preview`, and `description` fields, as well as in the labels attached to it. The `description` is defined as follows:

```
description = f"Political Response to Incident {row.ID} of {actor if actor not in missing else "} ({country if country not in missing else "}) {'on ' + date if date not in missing else "}"
```

while 4.6 shows the `define_newsPoliticalResponse_labels` method.

```
    # Define labels for Political Response News entities
    def define_newsPoliticalResponse_labels(polResp_dict):
        labels = []

        labels.append("Type:PoliticalResponse")

        # PoliticalResponseType: political_response_type if defined and available
        if polResp_dict.get('type','') not in missing:
            labels.append(f"PoliticalResponseType:{polResp_dict.get('type','').replace(': ','-')}")

        # PoliticalResponseSubcode: political_response_subcode if defined and available
```

```
12          if polResp_dict.get('subcode','') not in missing:
13              labels.append(f"PoliticalResponseSubcode:{polResp_dict.get('subcode
    ','')}")
14
15          return labels
```

Listing 4.6. `define_newsPoliticalResponse_labels` definition

In particular, the labels it attaches are:

- `Type:PoliticalResponse`, the standard label used to distinguish this `x-news` entity from the other `x-news` types;
- the `PoliticalResponseType` label, which maps the type of political retaliation;
- the `PoliticalResponseSubcode` label, which is used to map a subcategory of the `PoliticalResponseType`.

Unlike Legal Responses, Political Responses do not carry an equivalent of the `LegalResponseIndicator` label, reflecting the fact that political retaliation is not tracked elsewhere in the pipeline as an `x-operation` attribute. The handling of the actor performing the response is otherwise the same as for Legal Responses, as are the relationships created with the `x-operation`, `identity`, and `location` entities.

Attack Patterns

Attack Patterns represent the tactics used by the Initiators to cause the security incident that impacted the Receivers. In particular, EuRepoC maps the **MITRE Initial Access** and **MITRE Impact** tactics employed during the incident, following the MITRE ATT&CK taxonomy. Being part of the STIX 2.1 standard, they are easily mapped onto CTIS; in particular, they only require a slight mapping of their values to match the naming conventions used on the platform. The details of the information extraction can be found in 4.7.

```
1   # Map EuRepoC values to MITRE ATT&CK techniques
2   def define_attackPatterns(single_row):
3       initial_access = set()
4       impact = set()
5
6       # MITRE Initial Access
7       if isinstance(single_row.MITRE_initial_access, str):
8           initial_access.update(item.strip() for item in single_row.
    MITRE_initial_access.split(';') if item.strip())
9
10      # MITRE Impact
11      if isinstance(single_row.MITRE_impact, str):
12          impact.update(item.strip() for item in single_row.MITRE_impact.split('
    ;') if item.strip())
13
14      initial_access_mapped = {initial_access_mapping[x] for x in initial_access
     if x in initial_access_mapping} # Mapping    EuRepoC attack patterns (initial
     access) to CTIS attack patterns
15      impact_mapped = {impact_mapping[x] for x in impact if x in impact_mapping}
     # Mapping EuRepoC attack patterns (impact) to    CTIS attack patterns
16
17      return initial_access_mapped, impact_mapped
```

Listing 4.7. `define_attackPatterns` method

Since EuRepoC stores multiple MITRE values for a single incident as a semicolon-separated string, the function first splits and deduplicates the raw values into two sets, one for Initial Access and one for Impact, before mapping each value onto its CTIS-compatible counterpart via the `initial_access_mapping` and `impact_mapping` dictionaries. Values that have no corresponding entry in these mapping dictionaries are silently discarded, so that only techniques recognized by CTIS are propagated further down the pipeline.

After being extracted, the Tactic's values are used to search for the corresponding `attack-pattern` entity on CTIS and to link it to both the `x-operation` entity and the `x-news` (incident) entity. The relationship used is of type `uses` for the **x-operation**, reflecting the fact that STIX explicitly supports this relationship type between `attack-pattern` and `campaign`/`intrusion-set`-like objects to express that a given operation employed that tactic, and of type `related-to` for the **x-news**, consistent with the general limitation (already noted for Legal and Political Responses) that `x-news` entities on CTIS only support generic `related-to` relationships.

Receivers

The final object extracted by the pipeline from the EuRepoC dataset represents the victims of the cyber incident, referred to as **Receivers**. The extraction is performed by the `define_receivers` method, shown in Listing 4.8.

```
    # Extract the Receiver(s) fields from a single row
    def define_receivers(single_row):

        # ' - ' used to separate multiple receiver names
        names = list()
        if isinstance(single_row.receiver_name, str):
            names.extend([item.strip() for item in single_row.receiver_name.split(
" - ")])

        # ';' used to separate multiple receiver countries
        countries = list()
        if isinstance(single_row.receiver_country, str):
            countries.extend([item.strip() for item in single_row.receiver_country
.split(";")])

        # ' - ' used to separate region(s) of each country (there can be multiple
regions for each country, in that case they are separated by a ;)
        regions = list()
        if isinstance(single_row.receiver_region, str):
            regions.extend([item.strip() for item in single_row.receiver_region.
split(" - ")])

        categories = list()
        if isinstance(single_row.receiver_category, str):
            categories.extend([item.strip() for item in single_row.
receiver_category.split(" - ")])

        subcodes = list()
        if isinstance(single_row.receiver_category_subcode, str):
            subcodes.extend([item.strip() for item in single_row.
receiver_category_subcode.split(" - ")])

        # Some fields might be nan and therefore create empty lists in the code
above; to align the lengths we find the maximum length and fill the empty
lists with empty strings
        lists = [names, countries, regions, categories, subcodes]
        length = max((len(lst) for lst in lists), default=0)
```

```
30
31          for l in lists:
32              if len(l) < length:
33                  l.extend([""] * (length - len(l))) # Adjust the length of the
    lists where no values are present
34
35          # Truncate lists longer than the min_length to the length of the shortest
    list and fill empty lists with empty strings
36          return tuple([lst if lst else [""] * length for lst in lists])
```

Listing 4.8. `define_receivers` method

The extracted information is used to construct three main STIX entities:

- `identity`, created from the **Receiver Name** and subsequently linked to the `x-operation` entity via a `targets` relationship and to the `x-news` entity via a `related-to` relationship;
- `location`, resolved on CTIS using the **Receiver Country** and **Receiver Region** values and subsequently linked to the `x-operation` entity via a `targets` relationship, to the `x-news` entity via a `related-to` relationship, and, where an `identity` entity was created, to that entity via a `located-at` relationship;
- `sector`, resolved on CTIS using the **Receiver Category** and **Receiver Category Subcode** values. For this field, a dedicated method, `mapping_sectors` (Listing 4.9), is required, because of how sectors are represented in CTIS and in the STIX standard more generally. Rather than being attributes or labels attached to the `identity` entity, sectors in STIX are modeled as standalone `sector` entities with their own identifiers and relationships. Accordingly, the method first reconstructs the full set of EuRepoC sector/subsector categories for the row and then maps each one to the closest corresponding STIX `sector` entity. The resulting entities are linked to the `x-operation` entity via a `targets` relationship, to the `x-news` entity via a `related-to` relationship, and, where an `identity` entity was created, to that entity via a `related-to` relationship.

```
1       # Mapping EuRepoC receiver categories to CTIS sectors
2       def mapping_sectors(sectors, subsectors):
3           sec = set()
4           split_sectors = sectors.split(';')
5           split_subsectors = subsectors.split(';')
6           if len(split_sectors) != len(split_subsectors):
7               split_subsectors.extend([""] * (len(split_sectors) - len(
    split_subsectors))) # Adjust the length of the subsector list in case of
    sectors without subsectors (empty values)
8           for category, subcategory in zip(split_sectors, split_subsectors):
9               category = category.strip()
10              subcategory = subcategory.strip()
11              if category in missing:
12                  continue
13              if subcategory not in missing: # Case with subsector present
14                  sector = category + ": " + subcategory
15              else: # Case without subsector
16                  sector = category
17              sec.add(sector)
18
19          mapped_sectors = set()
```

```
20          mapped_sectors = {sector_mapping[x] if x in sector_mapping else x for x in
        sec}  # if sector is found in mapping use mapped value, else use original
        value
21
22          return mapped_sectors
```

Listing 4.9. `mapping_sectors` method

The method builds composite sector labels of the form `"Category: Subcategory"` whenever a subcategory is present, and falls back to the bare category otherwise. The variable `missing` is a list of sentinel values used throughout the pipeline to denote unavailable or non-informative data (e.g. empty strings or placeholder tokens produced when a field is not populated in the source dataset); categories whose value matches one of these sentinels are discarded before being added to the sector set, preventing spurious or empty sector entries from being created. The resulting set of composite labels is then passed through `sector_mapping`, a lookup table that translates EuRepoC's sector taxonomy into the corresponding CTIS `sector` entity names. Categories without a predefined mapping are handled according to a deliberate distinction: some are preserved as-is and attached as labels to the `x-operation` entity (e.g. **Social Groups**), reflecting categories deemed meaningful even without a direct CTIS sector equivalent, while others are intentionally dropped when no suitable one-to-one correspondence with CTIS exists.

Chapter 5

Feeder evaluation

5.1 Evaluation Methodology

Having described the design and implementation of the feeder in the previous sections, this chapter turns to its evaluation. Rather than assessing the feeder purely on functional grounds, i.e. whether it runs without errors and produces syntactically valid CTIS entities, the evaluation presented here aims to characterize the *substantive* output of the feeder: what kinds of information it actually extracts from the EuRepoC dataset, how that information is distributed, and what this reveals about both the underlying dataset and the feeder's mapping logic.

This distinction matters because a feeder can be functionally correct while still producing an output that is uninformative or skewed in ways that are not immediately visible from the code alone. For instance, a mapping method might successfully create an `intrusion-set` entity for every Initiator it encounters, yet the vast majority of those entities might still correspond to a single dominant actor, or the dataset might overwhelmingly favor incidents attributed to a small number of countries. Such patterns are not implementation errors, but they are important for an analyst to understand, since they shape how representative the resulting CTIS knowledge base is of the broader cyber threat landscape EuRepoC purports to describe.

The evaluation therefore focuses on three categories of information processed by the feeder, each corresponding to a core entity discussed earlier in this thesis: **Operation Labels**, **Intrusion Set Labels**, and **Sectors Targeted**. These were selected because they represent, respectively, (i) how incidents are classified in terms of the type of activity involved, (ii) which category of threat actors are most frequently responsible for incidents in the dataset, and (iii) which sectors are most commonly targeted. Taken together, these three dimensions give a reasonably comprehensive picture of the dataset's composition without requiring an exhaustive analysis of every field the feeder processes.

For each of these three categories, the feeder's output was aggregated into a dictionary mapping each distinct category value to the number of times it occurs across the processed EuRepoC incidents[1]. These frequency dictionaries were then used to produce

[1]Note that these counts reflect *label occurrences*, not distinct incidents: a single incident may be tagged

the distribution charts discussed in Section 5.2, which show, for each category, the most frequently occurring values, their relative share of the total, and where the number of distinct values exceeds what can be legibly displayed an aggregated "Other" category summarizing the long tail of infrequent values. Another important thing to note is that the labels used are not only from the EuRepoC feeder, but they may also be labels added by other sources of information to the Operations or the Intrusion Sets.

This approach was chosen over, for example, manually inspecting a sample of incidents, because it allows the evaluation to draw on the *entire* processed dataset rather than a subset, and because frequency distributions are well suited to surfacing both the dominant patterns in the data (e.g. the most common Intrusion Sets or Threat Actors) and its long tail (e.g. rarely-occurring sectors or highly fragmented attribution), both of which are relevant to assessing the feeder's real-world usefulness to a CTI analyst working with CTIS.

5.2 Numerical results

The labels were further divided in macro-areas to show more precise distributions based on their the type.

5.2.1 Targeted Sectors

One of the most important metrics to consider are the **Targeted Sectors**, they show which are the operations mostly covered by the EuRepoC consortium.

Figure 5.1 shows the distribution of the 20 distinct **Sectors Targeted** values extracted by the feeder. Rather than reflecting a general picture of global cyberattack targeting, this distribution is best understood as a direct consequence of EuRepoC's inclusion criteria, and should be interpreted accordingly.

EuRepoC's coding methodology restricts its dataset to incidents with a demonstrable *political* dimension: incidents must have affected a political or state actor, be attributable to a state-affiliated initiator, or have become politically salient regardless of the nature of the target [2]. This criterion mechanically privileges government institutions over private commercial entities, since an incident against a ministry or state agency almost by definition satisfies the political-dimension requirement, whereas an incident against, say, a retailer or a food and beverage company must clear a substantially higher bar, typically by being attributed to a state actor or by generating significant public and political attention, before it is included at all. The overwhelming prevalence of **National Government** as a targeted sector in the feeder's output is therefore less a reflection of attacker preference and more a structural property of what EuRepoC chooses to record in the first place.

This same methodological choice also explains the comparatively low counts observed for several commercial and industrial sectors toward the bottom of the distribution, such

with multiple sectors or multiple operation labels, and will therefore contribute to more than one count. This distinction is relevant when interpreting the totals reported alongside each distribution.

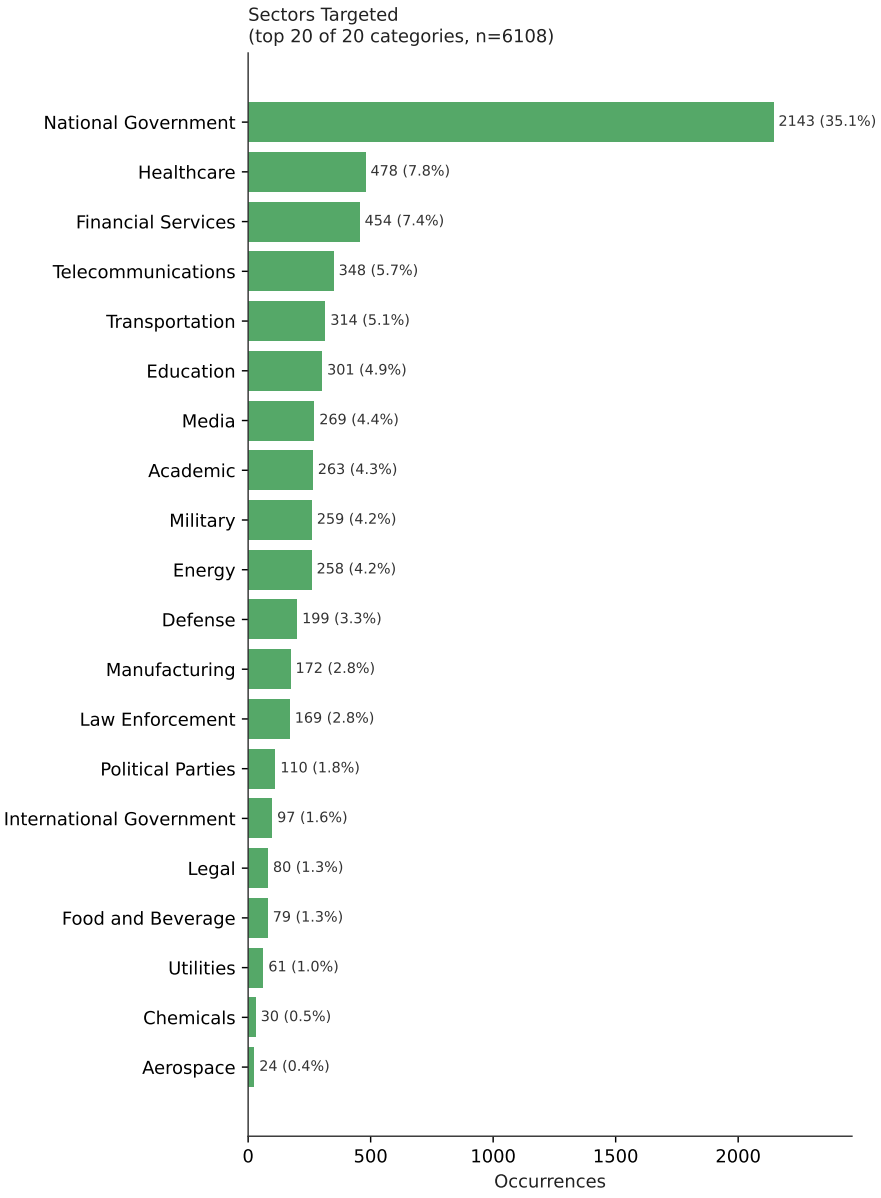


Figure 5.1. The Sectors Targeted

as **Food and Beverage**, **Chemicals**, and **Aerospace**. EuRepoC explicitly excludes the large majority of criminally motivated attacks against commercial entities (ransomware operations mostly) unless they carry a political dimension or a state nexus [2]. Since these sectors are frequent targets of ordinary financially motivated cybercrime but are less often the target of politically motivated operations, their under representation in the feeder's output is consistent with the dataset's scope rather than indicative of any shortcoming in the feeder's extraction logic.

A further factor concerns the **Critical Infrastructure** category of incidents, which EuRepoC only began systematically including from February 2023 onward, irrespective of the attributed initiator [2]. Several of the lower-frequency sectors observed in Figure 5.1, including **Energy**, **Utilities**, and, again, **Chemicals**, correspond closely to this category. Their relatively modest counts may therefore partly reflect the shorter time window over which EuRepoC has systematically tracked them, rather than these sectors being inherently less targeted; a similar chart produced from data collected several years from now might show these categories considerably more prominently represented.

Conversely, the comparatively strong showing of **Healthcare** and **Financial Services** is plausibly explained by their dual status: both sectors are widely designated as critical infrastructure while also being frequent targets of politically motivated operations, for instance, ransomware attacks against hospitals or state-linked espionage against financial institutions, giving them two independent routes into EuRepoC's dataset where a purely commercial sector might have only one, or none.

Taken together, these observations suggest that the sector distribution extracted by the feeder is a faithful reflection of EuRepoC's own coding scope and evolving inclusion criteria, and should be read as a description of *the dataset the feeder was built to process*, rather than as an independent claim about the global cyber threat landscape.

5.2.2 Conflicts

Figure 5.2 shows the distribution of **Conflict Labels** extracted by the feeder, reflecting 44 distinct conflicts across 1,546 label occurrences (top 25 shown). As with the Sectors Targeted distribution, this chart is best interpreted in light of EuRepoC's coding methodology rather than as a direct measure of "where cyberattacks happen most."

EuRepoC's offline-conflict-issue variable is derived from the Conflict Issues classification used by the Heidelberg Institute for International Conflict Research's *Conflict Barometer*, with an additional "cyber-specific" issue introduced to capture incidents that do not map onto any existing **offline conflict** [2]. In other words, the Conflict Label attached to an incident does not describe a property of the cyber incident itself, but rather identifies the pre-existing, offline political or interstate conflict to which EuRepoC attributes it, the cyber operation is treated as one further manifestation of a conflict that is already being tracked independently of the cyber domain. The distribution shown in Figure 5.2 can therefore be read as an indicator of which offline conflicts have also acquired a discernible cyber dimension, rather than as a standalone ranking of conflict intensity in its own right.

Read this way, the overwhelming dominance of **Russia – Ukraine** is unsurprising: it corresponds to the single largest, most sustained interstate conflict of the period covered by the dataset, one in which cyber operations have been documented as an explicit and recurring component of the broader military campaign, from both state and state-aligned non-state actors. The similarly high counts for **EU** and **USA et al. – Russia** plausibly reflect the same underlying conflict viewed through a wider lens: rather than being confined to a strict bilateral dyad, hostilities linked to the Russian invasion of Ukraine have generated cyber activity against the EU and its member states collectively, as well as against the broader Western coalition supporting Ukraine, which EuRepoC

appears to capture as related but analytically distinct conflict labels.

The remaining upper tier of the distribution is dominated by long-running great-power and regional rivalries with an established cyber dimension well documented in the wider cyber-conflict literature like **China – USA**, **Iran – Israel**, **China (Taiwan)**, **Iran – USA**, and **North Korea – South Korea** foremost among them. These are conflicts that predate the observation window of most cyber incident datasets and have historically been associated with sustained state-sponsored espionage, sabotage, or disruptive operations, so their appearance near the top of the distribution is consistent with prior research on state-sponsored cyber activity rather than a novel finding of this feeder's output.

By contrast, the long tail of the distribution, conflicts such as **Cuba (social protests)**, **Bahrain (opposition)**, or **Venezuela (opposition)**, each appearing only a handful of times, likely reflects sporadic, often hacktivist-driven activity tied to discrete political events (e.g. a wave of protests or a contested election) rather than a sustained cyber-conflict dimension. Because EuRepoC's inclusion criteria require a political dimension but do not require a conflict to be primarily fought through cyber means, such conflicts can appear in the dataset on the basis of a small number of opportunistic incidents, without implying any deeper or continuing cyber component to the underlying dispute.

5.2.3 Cyber Conflicts

Whereas the Conflict Labels discussed above identify the specific **offline conflict** an incident is attributed to (e.g. Russia – Ukraine, China – USA), the **Cyber Conflict Issue** labels shown in Figure 5.3 classify a different, complementary dimension of each incident: the underlying material or immaterial good that the offline conflict is fought over, following the conflict-issue typology defined by the HIIK Conflict Barometer [3] (territory, secession, decolonization, autonomy, system/ideology, national and subnational predominance, international power, and resources).

The two categories that dominate the distribution, **System/ideology** and **International power**, together account for the clear majority of all occurrences, which is consistent with the composition of the offline conflicts identified in the previous section: rivalries such as Russia – Ukraine, China – USA, or Iran – Israel are, at their core, disputes over international standing, spheres of influence, and competing political-ideological orders, rather than disputes over a specific piece of territory or a discrete material resource. Cyber operations, in particular espionage, sabotage, and influence operations, are a natural instrument for actors pursuing these kinds of systemic, non-territorial goals: they can be conducted remotely, deniably, and without the logistical footprint required by conventional military action, making them a comparatively low-cost tool for advancing a state's international standing or ideological agenda.

Territory and **Resources** occupy a smaller but still meaningful share of the distribution, plausibly reflecting the cyber dimension of a narrower set of conflicts with a more concrete territorial or material dispute at their core. For instance, incidents linked to the India – Pakistan or South China Sea disputes identified among the Conflict Labels. The comparatively low share of the residual `Cyber-specific` category is also worth noting: it suggests that the large majority of incidents captured by the feeder can, in fact, be traced back to a pre-existing offline conflict, and that incidents with a purely cyber-native origin

(lacking any identifiable offline political dispute) remain a minority in the dataset.

By contrast, issues more typically associated with intrastate or localized conflicts like **Autonomy**, **Secession**, **Subnational predominance**, and **Decolonization** appear only marginally in the distribution. A plausible explanation is that such conflicts are more often waged through conventional or asymmetric physical means by actors (separatist movements, sub-state militias, or similar) who typically lack the technical infrastructure and resources to conduct sustained cyber operations, and are less likely to attract the involvement of the well-resourced state and state-affiliated actors that dominate EuRepoC's Initiator data. Their near-absence from the Cyber Conflict Issue distribution is therefore consistent with the broader pattern observed throughout this evaluation: cyber operations captured by the feeder skew heavily toward interstate, systemic rivalries rather than localized or intrastate disputes.

5.2.4 EuRepoC Types

Figure 5.4 shows the distribution of **EuRepoC Type** labels extracted by the feeder, across 12,382 label occurrences. Unlike a conventional categorical variable, EuRepoC's incident-type coding does not assign each incident to a single, mutually exclusive category; rather, a single incident is typically tagged with several Type labels simultaneously, since these labels capture two conceptually distinct dimensions of an incident rather than competing descriptions of it [2]. **Data theft**, **Disruption**, **Data theft & Doxing**, and **Ransomware** describe the observed *effect* of an incident (what was done to the target) whereas **Hijacking with misuse** and **Hijacking without misuse** describe a more technical, architectural property: whether, and to what extent, the attacker achieved and exploited privileged access to the target's systems. Because hijacking is, in most cases, a technical prerequisite for causing an effect such as data theft or disruption, an incident coded as **Data theft** will frequently also be coded as **Hijacking with misuse**, rather than the two labels representing alternative, competing descriptions of the same event.

This two-dimensional structure explains why **Hijacking with misuse** is by a considerable margin the most frequently occurring label in the dataset. Rather than describing a distinct class of incidents in its own right, it functions as a near-universal companion label attached to any incident in which an attacker's system access was subsequently leveraged to cause harm, which, in practice, describes the majority of impactful incidents captured by EuRepoC, regardless of whether the specific harm involved data theft, disruption, or financial manipulation. Its high frequency is therefore better understood as a reflection of how consistently privileged access underlies successful cyber operations, rather than as evidence that "hijacking" by itself is the predominant *motive* behind the incidents in the dataset.

Among the effect-oriented labels, **Data theft** and **Disruption** clearly dominate, together accounting for close to half of all label occurrences. This is consistent with the two broad categories of Initiator motivation already suggested by the earlier distributions in this evaluation: state and state-affiliated actors are frequently associated with espionage and data exfiltration operations, coded here as Data theft, while less resourced actors (like hacktivist collectives) more often rely on comparatively low-cost disruptive

techniques such as distributed denial-of-service attacks or defacements, coded here as Disruption. **Data theft & Doxing**, a smaller but still notable category, likely captures a related but distinct pattern in which stolen data is not merely exfiltrated but deliberately made public, a tactic particularly associated with hacktivist and influence-driven operations seeking to embarrass or delegitimize a target rather than to exploit stolen information covertly.

By contrast, **Hijacking without misuse** and **Ransomware** occupy the smallest shares of the distribution, for two rather different reasons. Hijacking without misuse, by definition, describes incidents in which an attacker gained a foothold in a target's systems without any subsequent observable action [2]. Because such incidents leave no visible effect and are therefore substantially harder to detect and publicly report than incidents involving an active disruption or data breach, their comparatively low count is plausibly an artifact of EuRepoC's reliance on open-source and publicly reported incidents, rather than an indication that unexploited access is rare in absolute terms. Ransomware, meanwhile, is coded by EuRepoC not as a mutually exclusive category but always in conjunction with at least one other Type label, since a ransomware attack necessarily also involves disruption, data theft, or hijacking with misuse [2].

5.2.5 Technical Impact Labels

Figure 5.5 presents a markedly more heterogeneous picture than the previous distributions. Rather than representing values of a single categorical variable, the 22 labels shown here are drawn from several distinct sub-indicators that together feed into EuRepoC's **Cyber Intensity Indicator**, the composite measure EuRepoC uses to assess the severity of an incident along technical, functional, and socio-political dimensions, partly informed by the MITRE ATT&CK framework [2] [8]. Variables such as `LevelOfDataTheft`, `LevelOfDisruption`, and `LevelOfHijacking` record the severity of a specific effect *conditional on that effect having been coded for the incident in the first place*, an incident only receives a `LevelOfDisruption` value if it was already tagged with the **Disruption** Type label discussed in the previous section. As a result, the raw counts shown in Figure 5.5 are not directly comparable across variable families in the way the counts in a genuinely categorical distribution would be: a label's frequency reflects not only how severe incidents of that kind tend to be, but also how many incidents were eligible to receive a value for that variable at all.

The single most frequent label, **ExploitedZeroDays: No**, together with the comparatively negligible count for **ExploitedZeroDays: Yes**, confirms a pattern well established in the wider cyber-conflict literature and in EuRepoC's own reporting: zero-day vulnerabilities are costly to develop or acquire and are correspondingly rare in practice, with only a small fraction of incidents in any given year making use of one [2].

A related pattern emerges from **LevelOfHijacking**, where **System misuse** substantially outweighs **No misuse**. This is consistent with the dominance of **Hijacking with misuse** observed among the EuRepoC Type labels discussed previously, and plausibly reflects the same underlying dynamic: incidents in which an attacker gains access but does not visibly exploit it are inherently harder to detect and report on than incidents with an observable effect, so their relative scarcity in the dataset likely reflects a degree

of observation bias inherent to any dataset built primarily from public reporting, rather than a true rarity of unexploited intrusions.

The remaining labels, covering **FunctionalImpact**, **IntelligenceImpact**, **LevelOfTemporalEffects**, and **LevelOfSpatialEffects**, each occupy comparatively small shares of the total, which is consistent with their conditional coding structure described above: these variables provide finer-grained detail only for the subset of incidents already coded with a corresponding Disruption or Data theft effect, and are consequently applicable to, and coded for, a smaller pool of incidents than the broader Type labels themselves. Within this group, the near-total absence of **Widespread effects** and **Long lasting effects (>24h)** in the `LevelOfSpatialEffects` and `LevelOfTemporalEffects` variables is particularly notable, and is consistent with the well-established observation in the cyber-conflict literature that cyberattacks producing sustained, physically observable effects across a wide geographic area remain exceptionally rare events, even among a dataset specifically curated to capture politically significant cyber incidents.

5.2.6 Legal and Political Labels

Figure 5.6 aggregates a further set of variables from EuRepoC's international law and political-response dimension, distinct from both the technical severity indicators discussed previously and the conflict-attribution labels discussed earlier still. As with the technical labels, these are not values of a single categorical variable but belong to several conceptually related sub-indicators: the **International Law (IL) Breach Indicator**, the **State Responsibility Indicator**, the **Legal Response Indicator**, the **Target/Effect Multiplier** (labelled here as Target Political Importance), and the **Targeted Social Groups** category of the Receiver variable [2]. The **EU Proximity** label is instead a non-EuRepoC label which was added to the **Operation** entity after the upload on CTIS. Still, it serves as a way to have a basic understanding of the geographical distribution of the incidents.

The composition of the **IL Breach** labels is itself informative. **Sovereignty** is by a clear margin the most frequently invoked norm, which is consistent with its status in international law as the broadest and least demanding basis on which a cyber operation can be considered a breach: an unauthorized intrusion into another state's systems can, on a widely (if not universally) held customary-law argument, already constitute a violation of that state's territorial sovereignty, independently of whatever more specific harm the operation goes on to cause. More specific and demanding legal bases like **human rights**, **due diligence**, and **cyber espionage** appear less frequently, since establishing them requires evidence of a more particular kind of harm or omission, while narrowly scoped regimes such as **international telecommunication law**, **air law**, or **intellectual property law** are invoked only for the comparatively small subset of incidents that engage those specific technical domains.

A similar evidentiary logic appears to shape the distribution of the **State Responsibility Indicator**. **None/Negligent** is the lowest of the three degrees of state responsibility EuRepoC codes and is markedly more common than **Indirect** or **Direct** responsibility, the latter of which requires evidence that officials of state entities, agencies, or units were themselves responsible for the operation. This mirrors a pattern already suggested before: attribution to a state, and especially attribution meeting a

legally rigorous evidentiary threshold, is the exception rather than the rule among the incidents EuRepoC records, even within a dataset that specifically privileges politically significant incidents.

The **Legal Response Indicator** follows a closely related structure, since it is explicitly defined in terms of the same two underlying conditions, state attribution and breach of international law. **Countermeasures under international law justified** requires both conditions to be satisfied simultaneously and is consequently the least frequent of the three response categories, whereas **Unfriendly acts / retorsions justified** requires only that *one* of the two conditions hold, making it, by construction, the more permissive and therefore more frequently applicable category. The comparative rarity of **No response justified**, which requires that *neither* condition be met, indicates that the large majority of incidents in the dataset that present a Legal Response entity clear at least one of the two thresholds, even where they fall short of both.

The **Target Political Importance** labels reflect a similarly graduated design: **Very high** political importance is reserved for a narrower set of targets and is applied far more sparingly than the broader **High** category, consistent with it being the stricter of the two thresholds in EuRepoC's own intensity-weighting scheme.

Finally, the low overall share occupied by the **Targeted Social Groups** labels indicates that incidents specifically directed at a social group, as opposed to an institutional, corporate, or governmental target of the kind discussed in the Sectors Targeted distribution, remain a minority within the dataset. Among these, **advocacy / activist** organizations are the most frequently targeted subgroup, ahead of political opposition figures, religious groups, or other social groups.

The **EU Proximity** label is instead a non-EuRepoC label which was added to the **Operation** entity after the upload on CTIS. Still, it serves as a way to have a basic understanding of the geographical distribution of the incidents.

5.2.7 Intrusion Set Labels

Figure 5.7 shows the distribution of **Intrusion Set** labels attached by the feeder, spanning 64 distinct categories across 17,938 occurrences. Unlike every other distribution discussed in this evaluation, the labels shown here are of two fundamentally different origins. The `EuRepoCCategory:` and `EuRepoCSubCategory:` labels are added by the feeder itself, following the same logic described in Section 4.4.5: when an Initiator name is successfully matched to an existing Intrusion Set or Threat Actor entity on CTIS, the categories EuRepoC assigns to that Initiator are appended to the entity as labels. The remaining labels, malware and tooling references such as **RAT**, **Backdoor**, **Spearphishing**, **Downloader**, or **dropper**; motivation tags such as **FinancialGain**, **IntelligenceGathering**, or **State-Sponsored**; and generic activity descriptors such as **DDoS**, **Ransomware**, or **Criminal**, are not produced by the feeder at all. They *already existed* on the matched CTIS entities prior to the feeder's execution, most plausibly originating from earlier curation efforts such as MISP galaxy imports or prior analyst tagging. This distribution therefore reflects not only how EuRepoC itself categorizes the Intrusion Sets it identifies, but also the pre-existing richness of the CTIS entities the feeder happens to match against.

This composite nature is, in fact, informative in its own right. The strong presence of the feeder-added `EuRepoCCategory:Non-state group` and `EuRepoCCategory:Non-state actor, state-affiliation suggested` labels near the top of the distribution echoes a pattern already suggested by the Initiator data discussed earlier in this thesis: a substantial share of the Intrusion Sets the feeder successfully matches on CTIS are not unambiguously state actors, but non-state or state-affiliated groups occupying the same interpretive gray zone.

Among the pre-existing, non-EuRepoC labels, a similarly coherent pattern emerges. Tags associated with sophisticated, persistent intrusion activity like **RAT**, **Backdoor**, **Spearphishing**, **CredentialHarvesting**, **Downloader**, and **dropper**, together with the motivation tags **IntelligenceGathering** and **State-Sponsored**, occupy a considerable share of the distribution. This is consistent with the observation that state and state-affiliated Intrusion Sets tend to be among the most extensively documented threat actors in the open-source CTI community, and are therefore more likely to already possess a rich, pre-existing label set on a platform such as CTIS by the time the feeder attempts to match against them. By contrast, tags more typically associated with financially motivated or hacktivist activity like **Ransomware**, **DDoS**, **Type:ddos**, **Type:defacement**, **Criminal**, and **FinancialGain**, appear with lower, though still notable, frequency, broadly mirroring the relative prevalence of these actor types already observed in the **Initiator** and **Cyber Conflict Issue** distributions discussed earlier.

The unusually large **Other** category is itself a consequence of the composite origin of this chart. Because the non-EuRepoC labels were not assigned according to any single controlled vocabulary, but accumulated independently over time across different curation efforts, they naturally exhibit far greater fragmentation than EuRepoC's own tightly defined category scheme. This long tail is therefore better understood as a property of CTIS's pre-existing threat-actor curation than as a property of EuRepoC's data.

This distinction carries a broader implication for the interpretation of Figure 5.7: because the feeder's own labels coexist with, rather than replace, whatever labelling an entity already carried, the aggregated counts shown here conflate two different signals, *how EuRepoC categorizes a given Intrusion Set*, and *how extensively that Intrusion Set happened to already be documented on CTIS before the feeder ran.* While this behaviour is arguably desirable from a knowledge-preservation standpoint, since it avoids overwriting or duplicating prior analyst work, it does mean that this particular distribution should be read as a joint description of EuRepoC's categorization and of CTIS's pre-existing threat-actor coverage, rather than as a clean reflection of EuRepoC's data alone.

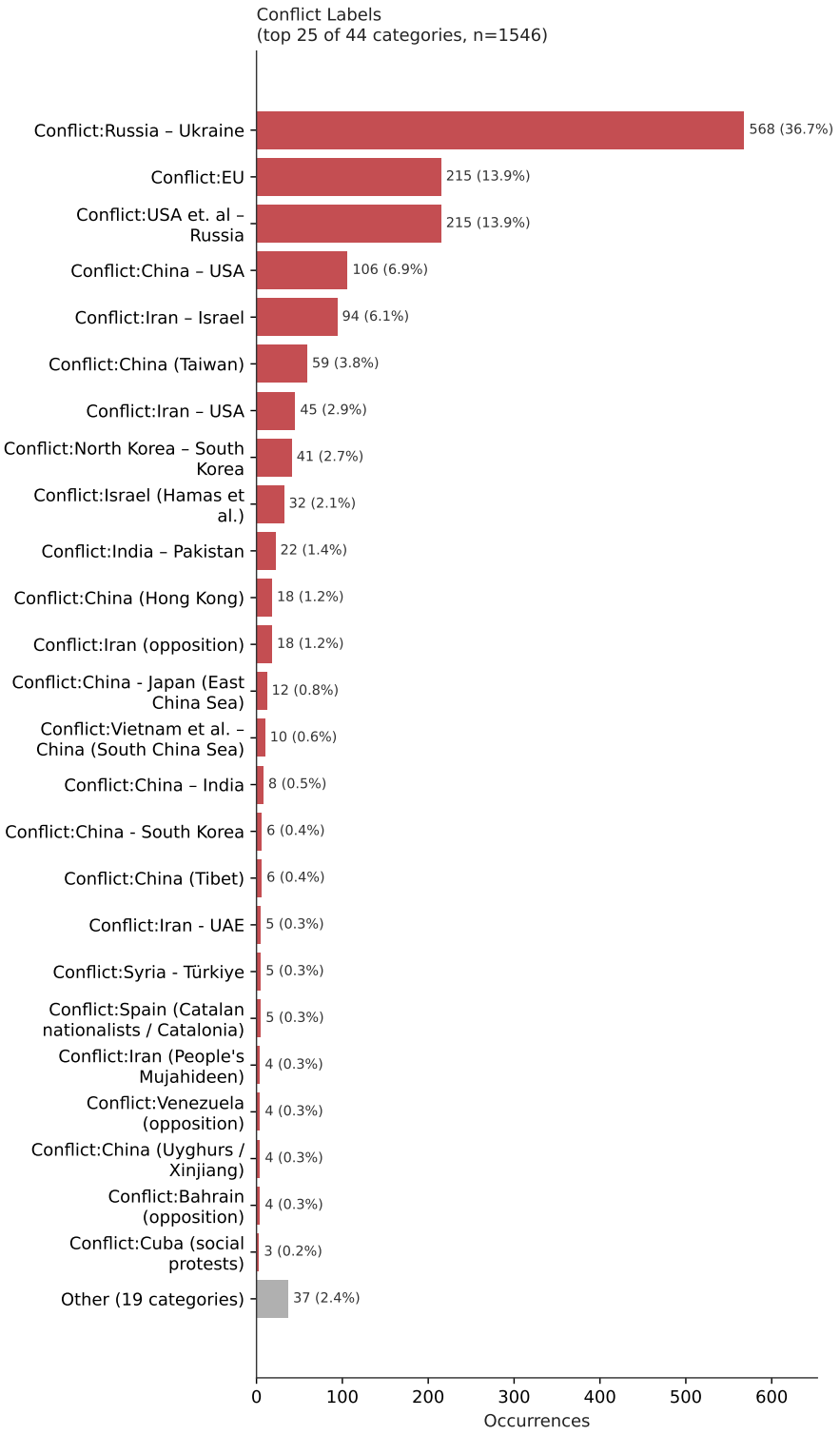


Figure 5.2. Conficts Involved

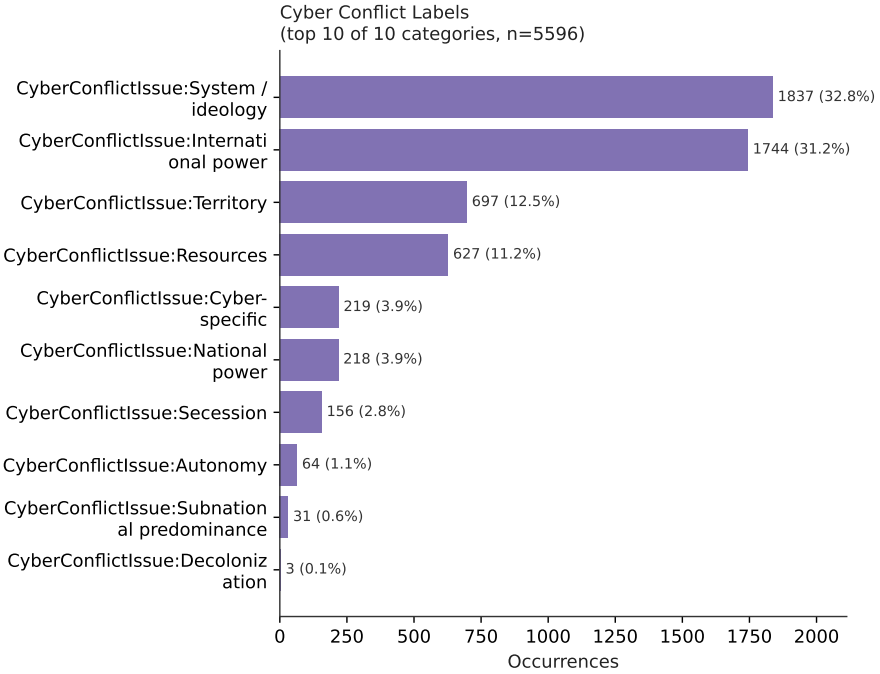


Figure 5.3. Nature of the conflicts involved

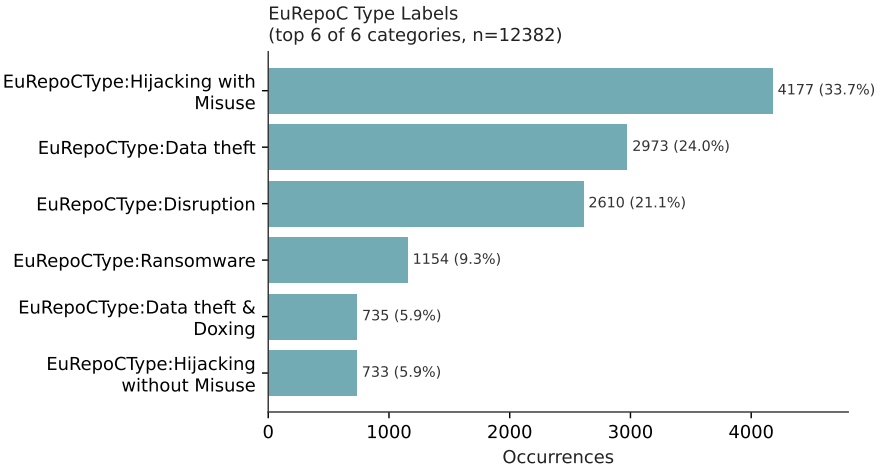


Figure 5.4. Distribution of EuRepoC Types

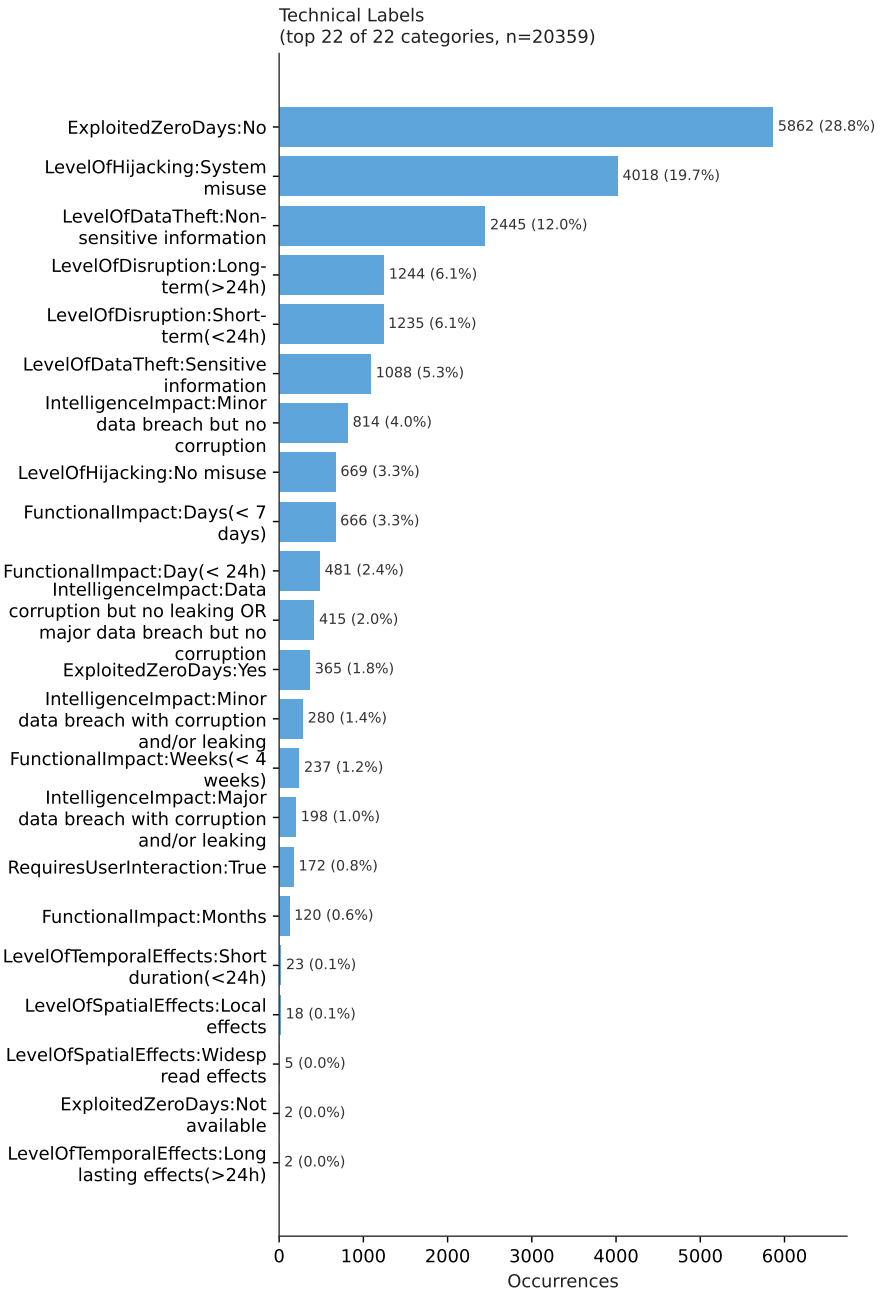


Figure 5.5. Technical Impact Labels

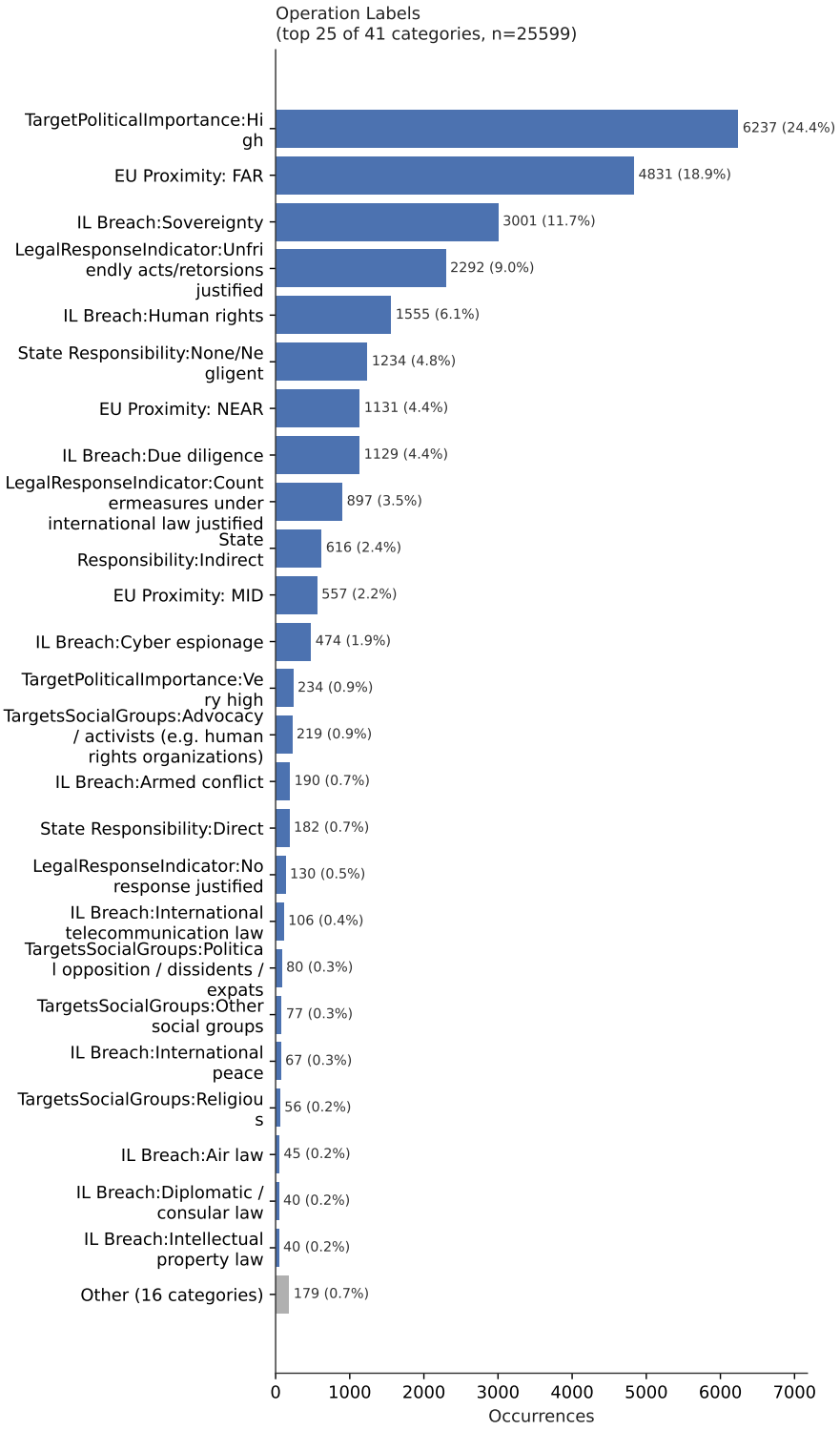


Figure 5.6. Legal and Political Labels

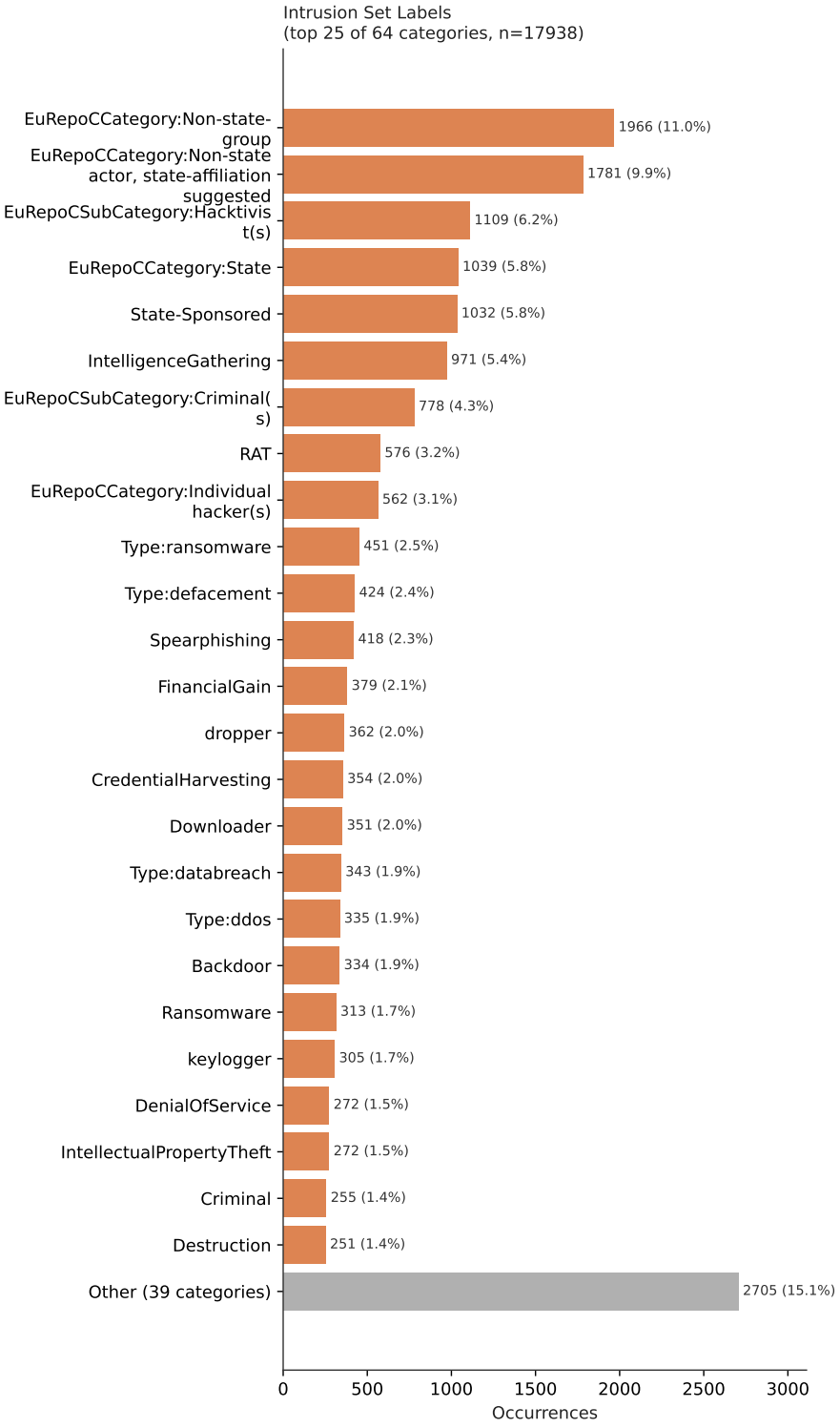


Figure 5.7. Intrusion Set Labels

Chapter 6

Conclusion

This thesis set out to describe in detail the **Collection** phase of the **CTI Cycle** from the point of view of a **CTI Analyst** tasked with identifying new **OSINT** sources of cyber incident data. To this end, the **European Repository of Cyber Incidents** (EuRepoC) was selected as the source for the feeder, and, in order to produce a knowledge base that is both reliable and standards-compliant, the feeder was designed around the theoretical foundations of the **CTI Cycle**, the **Diamond Model**, and, most importantly, the **STIX** standard, so as to load the dataset's information into **Leonardo**'s existing **TIP**, CTIS.

The feeder achieved an effective translation and mapping of EuRepoC-coded information into CTIS's modified STIX 2.1 standard. This mapping enabled the ingestion of external, heterogeneous information into an existing TIP, supporting both the creation of new entities and the enrichment of existing ones. Beyond its technical performance and robustness, the feeder also proved valuable from an analytical standpoint: it allowed for the qualitative evaluation of the incidents tracked by EuRepoC, together with the associated **Intrusion Sets** and the **Sectors** they targeted, offering CTI analysts a richer and more contextualized picture of the threat landscape than either source could provide in isolation. In this sense, the work confirms that automated, standards-based ingestion pipelines can meaningfully extend the analytical reach of an existing TIP without requiring analysts to manually reconcile disparate data formats.

Although the feeder successfully met its primary objective, some limitations remain. In particular, while the logic underlying the EuRepoC-to-STIX mapping is general and adaptable to other sources, the code responsible for data download, pre-processing, and de-coupling remains tightly coupled to EuRepoC's specific formatting choices. This dependency stems from the current reliance on the dataset's public-facing table view rather than a dedicated data-access layer, which makes the feeder somewhat brittle: any change to the website's formatting risks breaking the pre-processing pipeline. This limitation could be addressed by obtaining access to the EuRepoC consortium's API, which would provide a more structured version of the dataset, with entities already de-coupled, removing the need to continuously update the feeder in response to formatting changes. A second, related consequence of the lack of programmatic API access is the loss of information caused by cell truncation in the exported spreadsheet, as described in 4.4.2; this is a data-quality issue rather than a design flaw, but it nonetheless constrains the

completeness of the resulting knowledge base.

Several directions for future work follow naturally from these limitations. The most immediate would be securing access to EuRepoC's API in order to resolve the issues above. Other future projects could focus on further enriching the CTIS knowledge base by cross-referencing EuRepoC data with information from additional OSINT sources, thereby increasing both the coverage and the confidence of the resulting intelligence. Alternatively, the mapping logic established in this work could serve as a template: it could be reused to build new feeders for other incident-tracking sources, or generalized into a broader mapping framework capable of translating arbitrary incident data into the STIX standard used by CTIS, reducing the effort required to onboard future OSINT feeds.

Finally, one particularly promising avenue would be to address the most limiting aspect of the current feeder: its reliance on a human analyst to distinguish between an **Intrusion Set** entity and a **Threat Actor** entity. While this distinction is relatively straightforward for a human analyst able to draw on external context and research, it is considerably harder to automate without additional contextual information from the source itself. Future work could explore whether supplementary data sources, or targeted use of natural language processing techniques, could narrow this gap and move the feeder closer to full automation, though it is likely that some degree of analyst oversight will remain necessary for the foreseeable future, given the interpretive judgment this classification often requires.

Ultimately, this work demonstrates that OSINT sources such as EuRepoC can be systematically and reliably integrated into operational threat intelligence platforms, laying the groundwork for a more automated, standards-driven approach to cyber incident collection. In addressing how OSINT-derived incident data can be mapped onto an existing STIX-based TIP, this thesis contributes not only a working implementation, but also a reusable methodology, one that future researchers and practitioners can extend to other sources and to other stages of the CTI Cycle. As the volume and diversity of publicly available threat data continue to grow, approaches of this kind will be essential to ensuring that CTI analysts can keep pace, transforming the Collection phase from a manual bottleneck into a scalable, semi-automated process.

Bibliography

[1] Sergio Caltagirone, Andrew Pendergast, and Christopher Betz. The diamond model of intrusion analysis. Technical report, Center for Cyber Intelligence Analysis and Threat Research, 2013.

[2] European Repository of Cyber Incidents. EuRepoC codebook. Codebook European Repository of Cyber Incidents (EuRepoC) 1.2, 2023. EuRepoC.

[3] Heidelberg Institute for International Conflict Research. Conflict barometer 2023. Technical report, Heidelberg Institute for International Conflict Research (HIIK), Heidelberg, Germany, 2023.

[4] Eric M. Hutchins, Michael J. Cloppert, and Rohan M. Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. Technical report, Lockheed Martin Corporation, 2011.

[5] Leonardo S.p.A. C-THINK: Leonardo cyber threat intelligence. Product datasheet, MM09135, 2025.

[6] Katie Nickels. The cycle of cyber threat intelligence. SANS Digital Forensics and Incident Response, YouTube video, 2019.

[7] OASIS Open. STIX version 2.1. Oasis standard, OASIS Open, 2021.

[8] Adam Pennington, Amy Robertson, and Jackie Lasky. ATT&CK MITRE training. MITRE, 2020.

[9] Scott J. Roberts. CTI SquadGoals — setting requirements. Blog post, 2016. Originally published 30 March 2016 at sroberts.github.io.

[10] Pasquale Stirparo. Defining threat intelligence requirements. SANS Internet Storm Center (ISC) Diary, 2016. Published 25 September 2016.