



**POLITECNICO**  
**MILANO 1863**

SCUOLA DI INGEGNERIA INDUSTRIALE  
E DELL'INFORMAZIONE

# Argus: A Platform for Multi-source Cyber Threat Intelligence Collection and Analysis

TESI DI LAUREA MAGISTRALE IN  
COMPUTER SCIENCE AND ENGINEERING - INGEGNERIA IN-  
FORMATICA

Authors:

**Attilio Polito 10978292**

**Biagio Marra 11013670**

Advisor: Prof. Michele Carminati

Co-advisors: Lorenzo Bossi, Francesco Panebianco, Tommaso Paladini

Academic Year: 2025-2026





# Abstract

In the modern cybersecurity landscape, Cyber Threat Intelligence (CTI) has become essential for proactive defense strategies. However, its effectiveness is limited by massive data volumes and scattered information sources [23, 29]. Security analysts are increasingly challenged by the lack of a unified, timely, and granular view of the threat landscape, necessitating the manual correlation of heterogeneous data from Surface Web reports, ephemeral social media channels, and restricted communities within the Deep and Dark Web. This thesis addresses these challenges through the design and implementation of **Argus**, a distributed, microservices-based platform engineered to automate the entire lifecycle of intelligence data collection and processing. The proposed solution leverages a scalable ETL (Extract, Transform, Load) pipeline that orchestrates advanced web spiders to harvest data from disparate sources, overcoming technical barriers such as anti-bot protections and anonymous network routing (Tor). An AI pipeline enriches the raw data, transforming it into actionable intelligence, including Named Entity Recognition (NER) and automated classification (to filter noise from relevant, high-quality data), before being normalized into a structured Data Warehouse optimized for analytical queries. Furthermore, this work utilizes the deployed platform to conduct a comprehensive comparative analysis of the monitored sources. By defining and applying rigorous metrics, specifically timeliness, information density, and exclusivity to real-world case studies, the research highlights the distinct contribution of non-conventional sources to the early warning process. The empirical results confirm that official advisories are the most reliable baseline for validated disclosure and remediation, while underground sources provide a measurable Time-to-Insight advantage: Telegram is increasingly becoming a key reference point for early detection and forums can anticipate disclosure by weeks to months, while still providing strategic signals.

**Keywords:** Cyber Threat Intelligence, OSINT, Dark Web Monitoring, Microservices Architecture, ETL Pipeline, Automated Scraping, Named Entity Recognition.



## Abstract in lingua italiana

Nel panorama odierno della sicurezza informatica, la Cyber Threat Intelligence (CTI) è diventata fondamentale per abilitare strategie di difesa proattive. Tuttavia, la sua efficacia è limitata da volumi di dati massivi e da fonti informative frammentate [23, 29]. Gli analisti di sicurezza sono sempre più penalizzati dall'assenza di una visione unificata, tempestiva e granulare del threat landscape, che rende necessaria la correlazione manuale di dati eterogenei provenienti da report del Surface Web, canali effimeri dei social media e comunità a accesso ristretto nel Deep e Dark Web. Questa tesi affronta tali sfide attraverso la progettazione e l'implementazione di **Argus**, una piattaforma distribuita basata su microservizi, ingegnerizzata per automatizzare l'intero ciclo di vita della raccolta e dell'elaborazione dei dati di intelligence. La soluzione proposta sfrutta una pipeline ETL (Extract, Transform, Load) scalabile che orchestra web spider avanzati per acquisire dati da sorgenti eterogenee, superando barriere tecniche quali protezioni anti-bot e il routing su reti anonime (Tor). Una pipeline di AI arricchisce i dati grezzi, trasformandoli in intelligence azionabile, questa include un modulo di Named Entity Recognition (NER) e uno di classificazione (per filtrare il rumore dai dati CTI rilevanti), prima della normalizzazione in un Data Warehouse strutturato e ottimizzato per query analitiche. Questo lavoro utilizza inoltre la piattaforma implementata per condurre un'analisi comparativa completa delle sorgenti monitorate. Valutando metriche come tempestività, densità informativa ed esclusività, a casi di studio reali, la ricerca mette in evidenza il contributo distintivo delle sorgenti non convenzionali nel processo di early warning. I risultati empirici confermano che gli advisory ufficiali rappresentano la baseline più affidabile per disclosure e remediation validate, mentre le sorgenti underground forniscono un vantaggio misurabile in termini di Time-to-Insight: Telegram sta diventando un punto di riferimento chiave per l'identificazione precoce e i forum possono anticipare la disclosure da settimane a mesi, preservando comunque segnali strategici.

**Parole chiave:** Cyber Threat Intelligence, OSINT, Dark Web Monitoring, Architettura a Microservizi, Pipeline ETL, Scraping Automatizzato, Named Entity Recognition.



# Contents

|                                                                    |             |
|--------------------------------------------------------------------|-------------|
| <b>Abstract</b>                                                    | <b>i</b>    |
| <b>Abstract in lingua italiana</b>                                 | <b>iii</b>  |
| <b>Contents</b>                                                    | <b>v</b>    |
| <b>List of Figures</b>                                             | <b>ix</b>   |
| <b>List of Tables</b>                                              | <b>xiii</b> |
| <br>                                                               |             |
| <b>1 Introduction</b>                                              | <b>1</b>    |
| 1.1 Context and Motivation . . . . .                               | 1           |
| 1.2 Problem Statement . . . . .                                    | 2           |
| 1.2.1 The Fragmentation of Sources . . . . .                       | 2           |
| 1.2.2 Information Overload and "Unstructured Chaos" . . . . .      | 3           |
| 1.3 Proposed Solution and Contributions . . . . .                  | 3           |
| <br>                                                               |             |
| <b>2 Background and Motivation</b>                                 | <b>5</b>    |
| 2.1 Threat Intelligence Platforms and Standards . . . . .          | 5           |
| 2.1.1 Standardization Protocols: STIX and TAXII . . . . .          | 5           |
| 2.1.2 Collaborative Ecosystems: MISP . . . . .                     | 6           |
| 2.1.3 Knowledge Management: OpenCTI . . . . .                      | 6           |
| 2.2 Advanced Crawling Methodologies . . . . .                      | 7           |
| 2.2.1 Focused Crawling and the Deep Web "3D" Framework . . . . .   | 7           |
| 2.2.2 The Challenge of Dynamic Web: SPA and CSR . . . . .          | 7           |
| 2.2.3 Adversarial Challenges: Anti-Bot and CAPTCHAs . . . . .      | 8           |
| 2.2.4 Intelligence Timeliness . . . . .                            | 8           |
| 2.3 Natural Language Processing for CTI . . . . .                  | 8           |
| 2.3.1 Evolution: From Regex to Transformers . . . . .              | 9           |
| 2.3.2 Architectural Decision: Encoder vs. Decoder Models . . . . . | 9           |

|          |                                                                      |           |
|----------|----------------------------------------------------------------------|-----------|
| 2.4      | Research Questions . . . . .                                         | 9         |
| 2.5      | Related Works . . . . .                                              | 10        |
| 2.5.1    | Automated Collection from Adversarial Environments . . . . .         | 10        |
| 2.5.2    | NLP for Cybersecurity: From Extraction to Semantics . . . . .        | 11        |
| 2.5.3    | Comparative Analysis of Intelligence Sources . . . . .               | 13        |
| 2.5.4    | Comparative Analysis of First-Hand Underground CTI Sources . . . . . | 13        |
| 2.6      | Ethical Considerations in CTI Collection . . . . .                   | 14        |
| <b>3</b> | <b>Proposed Approach and Methodology</b>                             | <b>15</b> |
| 3.1      | Sources Selection . . . . .                                          | 15        |
| 3.1.1    | Structured Technical Feeds . . . . .                                 | 15        |
| 3.1.2    | Reports and Vendor Analysis . . . . .                                | 16        |
| 3.1.3    | Underground Forums and Dark Web . . . . .                            | 17        |
| 3.1.4    | Messaging Platforms . . . . .                                        | 18        |
| 3.1.5    | Summary of Source Selection . . . . .                                | 19        |
| 3.2      | System Architecture Overview . . . . .                               | 21        |
| 3.2.1    | Two-Phase Decoupling . . . . .                                       | 21        |
| 3.2.2    | Logical Clustering . . . . .                                         | 22        |
| 3.2.3    | The Asynchronous Orchestration Pattern . . . . .                     | 23        |
| 3.2.4    | Content-Based Routing Strategies . . . . .                           | 24        |
| 3.3      | The Relevance Gate Logic . . . . .                                   | 26        |
| 3.3.1    | Relevance Criteria . . . . .                                         | 26        |
| <b>4</b> | <b>Implementation and Engineering</b>                                | <b>29</b> |
| 4.1      | The Collection Layer – Extract . . . . .                             | 29        |
| 4.1.1    | Scrapy Cluster Implementation . . . . .                              | 29        |
| 4.1.2    | Dark Web Collection Infrastructure . . . . .                         | 31        |
| 4.1.3    | Structured Feed Integration . . . . .                                | 32        |
| 4.1.4    | Instant Messaging Platforms . . . . .                                | 32        |
| 4.1.5    | Data Lake Implementation . . . . .                                   | 33        |
| 4.2      | Intelligence Engine (Transform) . . . . .                            | 34        |
| 4.2.1    | Custom Sliding Window Algorithm . . . . .                            | 37        |
| 4.2.2    | Generative Summarization . . . . .                                   | 40        |
| 4.3      | Persistence and Observability (Load) . . . . .                       | 42        |
| 4.3.1    | Hybrid Data Warehouse . . . . .                                      | 42        |
| 4.3.2    | Control Plane Architecture . . . . .                                 | 43        |
| 4.4      | Security and Human-in-the-Loop . . . . .                             | 44        |

|          |                                                                     |            |
|----------|---------------------------------------------------------------------|------------|
| <b>5</b> | <b>Experimental Results and Analysis</b>                            | <b>47</b>  |
| 5.1      | Dataset Composition . . . . .                                       | 48         |
| 5.2      | AI Component Validation . . . . .                                   | 48         |
| 5.2.1    | Named Entity Recognition (NER) Performance . . . . .                | 48         |
| 5.2.2    | Threat Classification Model Selection . . . . .                     | 51         |
| 5.3      | Comparative Source Analysis . . . . .                               | 53         |
| 5.3.1    | Volume and Density Analysis . . . . .                               | 55         |
| 5.3.2    | Source Exclusivity and Uniqueness . . . . .                         | 65         |
| 5.3.3    | Cross-Source Overlap and Redundancy . . . . .                       | 71         |
| 5.3.4    | Timeliness and Celerity Analysis . . . . .                          | 80         |
| 5.3.5    | Strategic Coverage and Blind Spots . . . . .                        | 89         |
| 5.3.6    | Attribution Capabilities . . . . .                                  | 94         |
| 5.3.7    | Case Study: The MOVEit Transfer Campaign (CVE-2023-34362) . . . . . | 99         |
| 5.3.8    | Discussion and Answers to Research Questions . . . . .              | 102        |
| <b>6</b> | <b>Conclusions</b>                                                  | <b>107</b> |
| <b>7</b> | <b>Limitation and Future Work</b>                                   | <b>111</b> |
| 7.1      | Limitations . . . . .                                               | 111        |
| 7.2      | Future Work . . . . .                                               | 112        |
|          | <b>Bibliography</b>                                                 | <b>115</b> |
| <b>A</b> | <b>Configuration of the Scheduler</b>                               | <b>125</b> |
| A.1      | Declarative Job Definitions (config.yaml) . . . . .                 | 125        |
| A.2      | Dynamic Loading Logic (scheduler.py) . . . . .                      | 132        |
| A.3      | Job Execution Handlers (jobs.py) . . . . .                          | 136        |
| <b>B</b> | <b>Data Warehouse Backbone</b>                                      | <b>141</b> |
| B.1      | Sql_init - Data Warehouse initialization script . . . . .           | 141        |





## List of Figures

|      |                                                                                                                                                                                                                                                                           |    |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3.1  | System Architecture Overview . . . . .                                                                                                                                                                                                                                    | 23 |
| 5.1  | Training and Validation Loss curves over 15 epochs. . . . .                                                                                                                                                                                                               | 49 |
| 5.2  | Precision-Recall curves for Experiment C (Test Set). The Stacking Ensemble (Red line) compensates for the instability of the single Transformer (Orange) on small data, achieving the highest area under the curve (0.774) and outperforming the Baseline (Blue). . . . . | 53 |
| 5.3  | Total Entity Volume (2025). The Intelligence feeds confirms the high-volume, low-context nature of OSINT streams. . . . .                                                                                                                                                 | 56 |
| 5.4  | Average Entities per Item (2025). The convergence between Telegram and Bulletins, both acting as high-density dissemination vectors. . . . .                                                                                                                              | 57 |
| 5.5  | Information Density in 2025. Intelligence feeds provide pure signal, while Forums and Reports are diluted by narrative text. . . . .                                                                                                                                      | 58 |
| 5.6  | Total Entity Volume (2024). The significant volume of DLS (3,567), reflecting a period of stability for Tor-based leak sites before the 2025 decline. . . . .                                                                                                             | 59 |
| 5.7  | Average Entities per Item (2024). The DLS metric remains fixed at 3.0, confirming the deterministic nature of the extraction pipeline regardless of total volume. . . . .                                                                                                 | 60 |
| 5.8  | Information Density in 2024. The low density of Reports (4.29), indicating a prevalence of narrative-heavy strategic documents. . . . .                                                                                                                                   | 61 |
| 5.9  | Total Entity Volume (2023). Bulletins dominate due to high persistence, while DLS volumes (60) are negligible due to the decay of .onion infrastructure. . . . .                                                                                                          | 63 |
| 5.10 | Average Entities per Item (2023). The consistency of the metrics (DLS at 3.00) validates the stability of the extraction pipelines across the years. . . . .                                                                                                              | 64 |
| 5.11 | Information Density in 2023. Reports show minimal density (6.09), confirming their role as narrative and strategic documents rather than technical feeds. . . . .                                                                                                         | 65 |

|                                                                                                                                                                                                                                            |    |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 5.12 Exclusive Entities (2025). Telegram (29,825) outperforms Bulletins, proving its role as a generator of unique intelligence. Reports show minimal exclusivity (126), confirming their role as contextualizers of known threats.        | 67 |
| 5.13 Exclusive Entities (2024). The traditional hierarchy is visible, with Bulletins (25,978) leading Telegram. DLS shows a significant contribution of 1,200 unique domains not found elsewhere. . . . .                                  | 69 |
| 5.14 Exclusive Entities (2023). Bulletins (26,981) dominate the historical landscape, proving their archival stability. The low value for Intelligence (7,428) and DLS (26) illustrates the rapid decay of ephemeral technical data. . . . | 71 |
| 5.15 Source Overlap Table (2025). The Bulletin-Telegram 16.87% Jaccard Similarity, we can see massive synchronization between official advisories and underground channels. . . . .                                                        | 72 |
| 5.16 Overlap Heatmap (2025). The central axis is formed by Telegram and Bulletins, with Discord acting as a satellite community. . . . .                                                                                                   | 73 |
| 5.17 Strategic Overlap (APT-TTP Links). 1,038 semantic links between Bulletins and Telegram proves that they provide consistent attribution logic, not just raw indicators. . . . .                                                        | 74 |
| 5.18 Source Overlap Table (2024). The Bulletin-Telegram axis has 15.50% similarity, proving that the synchronization of raw indicators was established before the 2025 volume boom. . . . .                                                | 75 |
| 5.19 Overlap Heatmap (2024). High connectivity between Telegram, Forums, and Bulletins, contrasting with the isolation of DLS. . . . .                                                                                                     | 76 |
| 5.20 Strategic Overlap (APT-TTP Links) in 2024. The shared links (338), lower compared to 2025 (1,038) highlights that Telegram had not yet fully matured as a source of attribution intelligence. . . . .                                 | 77 |
| 5.21 Source Overlap Table (2023). Bulletins and Telegram coupling is already evident (14.16% Jaccard). . . . .                                                                                                                             | 78 |
| 5.22 Overlap Heatmap (2023). The visualization confirms the density of the Bulletin-Telegram connection, while highlighting the total isolation of DLS (due to Survivorship Bias). . . . .                                                 | 79 |
| 5.23 Strategic Overlap (APT-TTP Links) in 2023. Telegram had not yet matured into a source capable of mirroring complex attribution logic (low volume of shared semantic links, 347). . . . .                                              | 80 |
| 5.24 Source Anticipation Matrix (2023–2025). The analysis shows how often one source family anticipates another across the longitudinal dataset (ties are excluded from the evaluation). . . . .                                           | 81 |

|      |                                                                                                                                                                                                                       |     |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 5.25 | "First to Report" Ranking (2025). Bulletins lead the standard disclosure cycle, but Telegram's high rank (6,342) confirms its capability to preempt official channels. . . . .                                        | 83  |
| 5.26 | First Seen Matrix (2025). Rows highlighted in green show the earliest detection. . . . .                                                                                                                              | 84  |
| 5.27 | "First to Report" Ranking (2024). The dominance of Bulletins (8,286) confirms that the "Standard Disclosure" model was the norm, with underground sources mostly reacting to official news. . . . .                   | 85  |
| 5.28 | First Seen Matrix (2024). Official sources were usually the first to detect threats. CVE-2024-4577 (Forum lead) represents a rare strategic outlier. .                                                                | 86  |
| 5.29 | "First to Report" Ranking (2023). The landscape is dominated by Bulletins (7,189), confirming a reactive CTI model where official vendors initiate the disclosure cycle. . . . .                                      | 87  |
| 5.30 | First Seen Matrix (2023). Specific exploit development community active prior to official disclosure in Discord. . . . .                                                                                              | 89  |
| 5.31 | Threat Coverage by Source (2025). Intelligence (Malware-focused) and DLS (APT-focused), contrasted with the broad, full-spectrum coverage of Telegram and Bulletins. . . . .                                          | 91  |
| 5.32 | Threat Coverage by Source (2024). Bulletins leadership covering Actors, Malware, and Vulnerabilities. Discord shows a specific specialization in TOOLS, surpassing other underground sources in this domain. . . . .  | 92  |
| 5.33 | Threat Coverage by Source (2023). Bulletins display the most balanced coverage. Discord stands out for its high density of TOOLS, while Intelligence feeds remain restricted to Actors and Malware. . . . .           | 94  |
| 5.34 | Linked APTs per Source (2025). Bulletins (1,630) and Telegram (1,386) coupling, they are the two primary engines for attribution. Intelligence feeds show minimal capability in this domain. . . . .                  | 96  |
| 5.35 | Linked APTs per Source (2024). Bulletins (1,255) lead significantly, but Telegram (699) has already substituted traditional Forums as the primary underground source for actor identification. . . . .                | 97  |
| 5.36 | Linked APTs per Source (2023). Dominance of Bulletins (1,112), illustrating a traditional model where attribution is an institutional function. Communities like Discord (15) show zero strategic visibility. . . . . | 99  |
| 5.37 | Dread post sharing a PoC for MOVEit Transfer. . . . .                                                                                                                                                                 | 100 |



# List of Tables

- 3.1 Detailed taxonomy of monitored intelligence sources, strategic value, and validating literature. . . . . 20
- 5.1 Global performance metrics of the Neural NER model. . . . . 50
- 5.2 Detailed classification report by entity type. . . . . 50
- 5.3 Comprehensive Model Selection Matrix. The table highlights that the model trained on the small, high-quality dataset (Exp C) consistently outperforms models trained on larger, noisier datasets (Exp A, Exp B) across all key metrics. . . . . 52
- 5.4 Timeline Analysis of the MOVEit Campaign. Note the 37-day gap between the Forum PoC and the CISA Alert. . . . . 101



# 1 | Introduction

In the contemporary digital landscape, the information security paradigm has gone into a big shift. Traditional security programs, centered on preventive perimeter controls and the detection of known artifacts through rules and signatures, are increasingly insufficient to guarantee organizational resilience. As documented by the European Union Agency for Cybersecurity (ENISA) in its *Threat Landscape* report [19], the current ecosystem is characterized by the increasing specialization of cybercrime groups and the attack vectors rapid evolution. In this environment, *proactive* defense refers to intelligence-led anticipation: identifying emerging threats and pre-positioning detections and mitigations before incidents materialize.

This necessity made **Cyber Threat Intelligence (CTI)** a core discipline in modern Security Operations Centers (SOCs). CTI is not merely the consumption of data; it is the evidence-based knowledge, including context, mechanisms, indicators, implications, and actionable advice, about an existing or emerging menace to assets [45].

However, despite its clear theoretical advantages, operationalizing Cyber Threat Intelligence (CTI) remains, in practice, a difficult task. Analysts must defend an expanding perimeter while navigating a complex ecosystem of heterogeneous sources and massive data volumes. As highlighted in this work, the efficacy of modern CTI is often constrained by two systemic factors: the extreme fragmentation of sources and the difficulty for analysts to process and prioritize an excessive volume of incoming information. [70].

## 1.1. Context and Motivation

The motivation behind this research arises from a tangible gap observed in the daily practice of threat analysis: the persistent *fragmentation* of cyber threat intelligence. To construct a comprehensive threat model, analysts frequently need to correlate heterogeneous signals across multiple, technically incompatible domains (e.g., vendor advisories, CERT bulletins, commercial reports, social-media channels, and underground sources). While enterprise platforms can partially address this problem, in many settings the in-

tegration effort, access constraints, and irregular coverage across sources still result in substantial manual triage and correlation work.

In practice, analysts must consult official advisories from CERTs and vendor reports (e.g., Mandiant, CrowdStrike) to obtain reliable but often delayed information. At the same time, they may need to monitor fast-moving channels such as X or Telegram, where early technical artefacts (e.g., proof-of-concepts or exploitation informations) can surface before consolidated, official recognition [3, 60]. This phenomenon is also reflected in our empirical results (Chapter 5), which quantify a measurable time-to-insight advantage for specific underground and semi-public channels.

In addition, relevant signals frequently reside within the Deep and Dark Web (DDW), in underground forums and Data Leak Sites (DLS). The access to these environments requires specialized tools, careful OpSec, and significant time investment to handle anti-bot protections and CAPTCHAs.

Although several commercial "all-in-one" solutions attempt to aggregate parts of this ecosystem, their licensing, integration, and operational overhead can represent a significant entry barrier for smaller organizations and independent researchers. Consequently, this thesis addresses the need for a unified and automated pipeline that reduces the manual burden of multi-source collection, normalization, and correlation. Rather than replacing existing enterprise tools, the proposed approach aims to bridge gaps in coverage and interoperability across fragmented sources, enabling a more systematic and scalable characterization of the threat landscape by empirically comparing the monitored source families through dedicated metrics (timeliness, exclusivity, and information density) that are directly informative of their practical CTI value (Chapter 5).

## 1.2. Problem Statement

Despite the abundance of data, analysts suffer from the absence of a unified view of the threat landscape. This thesis identifies and addresses two distinct classes of problems: the friction caused by fragmentation and the poor signal-to-noise ratio of unstructured data.

### 1.2.1. The Fragmentation of Sources

The primary obstacle is the lack of interoperability between intelligence sources. Analysts are often forced to switch contexts repeatedly between a Tor Browser session to check a ransomware leak site, a social media feed to track active exploitation, and a SIEM



dashboard to search for Indicators of Compromise (IoCs). This manual correlation is slow, error-prone, and inherently unscalable. Furthermore, there is a lack of rigorous empirical research that quantifies the strategic difference between these sources. Without a unified platform to capture data concurrently, it is difficult to scientifically assess whether a type of source genuinely offers a "Lead Time" advantage over another, or to identify the "Blind Spots" where specific sources fail to detect certain threat categories.

### 1.2.2. Information Overload and "Unstructured Chaos"

The second challenge lies in the nature of the data itself. A raw extraction from a Dark Web forum often yields a dataset characterized by an extremely low signal-to-noise ratio. A critical post regarding a new ransomware variant is frequently obscured under massive volumes of noise, including spam, drug trafficking, and credit card fraud.

Moreover, this intelligence is rarely structured in standard formats like STIX. It is hidden within natural language discussions, colloquial slang, and unstructured text. Without automated mechanisms to filter noise via **Relevance Filtering** (filtering what is relevant for the environment, from what is not relevant) and to extract entities via **Named Entity Recognition (NER)**, the raw data collected from these sources remains practically unusable for automated defense strategies.

## 1.3. Proposed Solution and Contributions

To address these challenges, this thesis presents **Argus**, a distributed platform based on a microservices architecture, specifically engineered to automate the entire lifecycle of intelligence data collection and processing. Argus is designed not only as an engineering tool but as a research instrument to validate the efficacy of continuous monitoring, and to compare the intelligence value of different sources over time.

The contributions of this work are categorized into three key areas:

1. **Automated Distributed ETL Pipeline:** We propose a resilient architecture that implements a "Two-Phase Decoupling" strategy, separating data acquisition (Extract) from processing (Transform). This allows a scalable ingestion of heterogeneous data, from structured feeds, to unstructured Dark Web forums; overcoming technical barriers such as Tor latency, CAPTCHA challenges, and anti-bot measures.
2. **AI-Driven Intelligence Engine:** To solve the signal-to-noise ratio problem, the system integrates a Deep Learning pipeline. This includes a **Relevance Gate**,

a binary classification model to filter out non-CTI content at the source, and a Transformer-based **NER** model. The latter is fine-tuned to extract 21 specific cybersecurity entities (e.g., **Malware**, **Threat\_Actor**, **CVE**), transforming unstructured text into a structured one.

3. **Comparative Source Analysis:** Quantitative evaluation of source quality through various metrics. We analyze the **Timeliness**, measuring the "Lead Time" of specific source types in anticipating threats compared to the rest; **Exclusivity**, identifying unique intelligence coverage; and **Information Density**, evaluating sources by the substance of high-value indicators per document.

To ensure that the comparison is statistically meaningful, the platform produced a longitudinal dataset spanning from January 2023 to January 2026 and collecting approximately 500,000 artifacts. This scale enables a quantitative characterization of how different source families behave under a unified schema.

Comparative evaluation (Chapter 5) highlights measurable, longitudinal differences across source families. Across 2023–2025, Bulletins remain the dominant driver of the standard disclosure cycle (first-to-report: 7,189 → 8,286 → 9,663), while Telegram’s early-warning role increases significantly (3,011 → 2,735 → 6,342). Telegram’s unique coverage expands (13,302/11,851 → 29,825 exclusive entities), while Bulletins remain relatively stable at ~26k per year, and the strategic coupling between Bulletins and Telegram becomes stronger (shared APT–TTP links: 347/338 → 1,038). In the end, underground forums consistently act as an anticipation layer for specific high-impact cases, surfacing exploitation discussions and PoCs ahead of official advisories, with lead times that range from weeks to several months in representative outliers. These observations motivate the core claim of this work: while official sources provide validated remediation, semi-public and underground streams can complement them with earlier signals and additional coverage.

## 2 | Background and Motivation

This chapter provides a comprehensive and critical review of the current technological landscape regarding Cyber Threat Intelligence (CTI) platforms, automated data collection methodologies, and Natural Language Processing (NLP) techniques applied to the security domain. The objective of this analysis is not merely to list existing technologies, but to highlight the specific methodological gaps and engineering challenges that characterize the current state of the art. These limitations are the primary motivation behind the design and implementation of the Argus platform, which aims to propose a new architectural approach to the problems of collection and analysis in heterogeneous environments.

### 2.1. Threat Intelligence Platforms and Standards

In recent years, the exponential growth in the volume of threat data has necessitated a shift from unstructured, ad-hoc communication to formalized standards and dedicated management platforms. The industry has moved towards the adoption of Threat Intelligence Platforms (TIPs) designed to aggregate, correlate, and analyze data from multiple feeds.

#### 2.1.1. Standardization Protocols: STIX and TAXII

To facilitate interoperability between disparate security tools and organizations, the OASIS consortium developed the **Structured Threat Information Expression (STIX)** language and the **Trusted Automated Exchange of Intelligence Information (TAXII)** protocol [50].

STIX 2.1 represents a significant evolution in how threat information is modeled. It provides a standardized, graph-based language to describe not just atomic indicators (such as IP addresses or file hashes), but also complex relationships involving Threat Actors, Campaigns, Attack Patterns, and Victim Targets. However, while these standards are fundamental for the exchange of mature intelligence, literature suggests that their rigid schema can introduce significant overhead during the early stages of the intelligence

cycle. When dealing with raw, noisy, and unstructured data, such as informal discussions on a Dark Web forum, forcing such data immediately into a strict STIX object can be premature and computationally expensive, often resulting in a loss of the original context.

### 2.1.2. Collaborative Ecosystems: MISP

**MISP (Malware Information Sharing Platform)** has established itself as the de-facto industry standard for storing and sharing structured threat intelligence [66, 71]. Its success lies in its ability to foster a collaborative defense ecosystem, allowing organizations to share Indicators of Compromise (IoCs) rapidly. MISP excels at correlating events across different organizations, enabling analysts to see if an attack seen in their network has already been observed elsewhere.

However, existing studies and practical experience highlight that MISP is primarily optimized for the *post-processing* and *dissemination* phases of the intelligence cycle. From the perspective of *automated collection*, it presents specific architectural limitations. Firstly, it relies on a Passive Ingestion paradigm [66]: it is designed to consume feeds of already formatted data (e.g., CSV, JSON lists, or STIX files) rather than actively harvesting data. It lacks native, built-in capabilities to perform active, "aggressive" scraping of unstructured web sources, navigate dynamic websites, or bypass CAPTCHAs. Secondly, its Data Validation Paradigm is optimized for managing intelligence that has already been validated. Feeding raw scraping results directly into MISP without a dedicated pre-processing pipeline would result in database saturation and a severe degradation of the signal-to-noise ratio.

Consequently, Argus is not a replacement for MISP, but a complementary instrument, capable of collecting fresh data, filtering the noise, and feeding MISP only with high-value, structured intelligence.

### 2.1.3. Knowledge Management: OpenCTI

OpenCTI represents a modern approach to TIPs, designed to unify knowledge management in CTI [22]. Unlike MISP's event-centric model, OpenCTI focuses on a knowledge-graph approach, strictly adhering to the STIX 2.1 data model [22, 50].

While OpenCTI offers a modular "connector" architecture for data ingestion, it presents challenges for agile, custom deployments focused on scraping. The Infrastructural Overhead is significant, as it relies on a heavy enterprise-grade microservices architecture (ElasticSearch, Redis, RabbitMQ, TypeDB) which can be prohibitive for lightweight research

deployments. Furthermore, developing custom logic for scraping dynamic websites or navigating complex authentication flows via connectors is often more complex compared to using dedicated, specialized scraping frameworks like Scrapy.

## 2.2. Advanced Crawling Methodologies

The collection of Open Source Intelligence (OSINT) and, more specifically, data from the Deep and Dark Web, presents unique engineering challenges. Let's dive into a general overview of the ways, identified in the State of the Art, to overcome them.

### 2.2.1. Focused Crawling and the Deep Web "3D" Framework

Unlike general crawlers that aim to index the entire web, focused crawlers prioritize pages relevant to a specific topic. As discussed by Khder (2021), web scraping has evolved from simple HTML parsing to complex automation required for Business Intelligence and Security [36].

The challenge is amplified in the Dark Web, the main problem is the lack of a central index. Adewopo et al. (2020) propose a "3D Framework" approach to Deep Web scraping, involving three distinct phases: Discovery, Description, and Discrimination [1]. The first phase is the process of finding new, unknown hidden services ('.onion' sites). This is the most critical bottleneck, often requiring directory spidering or link analysis. Then we have the automated characterization of the site's content (e.g., distinguishing a marketplace from a forum). In the end, there is the filtering phase, identifying whether the content poses a relevant threat.

Argus philosophy is similar to this framework, leveraging specific feeds or spiders to feed the scraping queue dynamically.

### 2.2.2. The Challenge of Dynamic Web: SPA and CSR

Traditional HTTP crawlers are increasingly ineffective against modern web applications. The widespread adoption of **Single Page Applications (SPAs)** and Client-Side Rendering (CSR) frameworks often lead to an empty HTML response. The actual content is loaded asynchronously via JavaScript.

For years, Selenium has been the standard tool for browser automation. However, its synchronous, blocking architecture limits scalability. Recent best practices adopt Playwright as a better alternative for high-concurrency scraping. Unlike Selenium, Playwright

supports asynchronous execution and communicates directly with the browser engine protocols (DevTools Protocol), allowing for seamless integration with asynchronous crawling frameworks like Scrapy.

### 2.2.3. Adversarial Challenges: Anti-Bot and CAPTCHAs

Underground communities actively deploy countermeasures to block automated scrapers. A crawler must be "stealthy," mimicking human behavior to avoid bans. TLS Fingerprinting (e.g., JA3 signatures) is a primary identification mechanism used by defenses like Cloudflare. Standard Python libraries transmit parameters in a fixed order that differs from legitimate browsers, leading to blocking [27]. Advanced scrapers must utilize techniques such as JA3 impersonation to randomize cipher suites and mimic legitimate handshakes.

The application layer presents in addition the barrier of CAPTCHAs. In the context of the Dark Web, forums use increasingly complex puzzles. While OCR solutions exist, modern logic-based CAPTCHAs often exceed current automated solving capabilities. This necessitates a "Human-in-the-loop" design pattern: rather than aiming for impossible complete automation, a resilient architecture must allow the pipeline to pause for manual intervention during authentication, automating only the subsequent data retrieval.

### 2.2.4. Intelligence Timeliness

A recent research by Paladini et al. (2024) [51] has analyzed the temporal relationship between underground forums and official reports, demonstrating that, while the "Lead Time" (the time advantage of the Dark Web) is narrowing, underground forums still anticipate official disclosures for a significant percentage of threats [51]. This empirically validates the strategic necessity of monitoring these high-latency sources despite the engineering cost.

## 2.3. Natural Language Processing for CTI

The transformation of the unstructured text collected from the web into structured, actionable intelligence requires advanced Information Extraction (IE) techniques.

### 2.3.1. Evolution: From Regex to Transformers

Early approaches relied on Regular Expressions (Regex). While efficient for structured IoCs (e.g., IPs, Hashes), they fail to capture context. The state of the art is represented by Transformer-based models (e.g., BERT, RoBERTa). Papers demonstrate that Transformers significantly outperform previous architectures (CNN/RNN) in CTI tasks. Their "Self-Attention" mechanism enables a deep understanding of context, essential for distinguishing between a "patch" for a vulnerability and a "patch" of code used in an exploit. Specific models like DarkBERT [34], pre-trained on Dark Web data, have shown superior performance in handling the specific slang of underground communities.

### 2.3.2. Architectural Decision: Encoder vs. Decoder Models

A critical distinction must be made between Encoder-only models (e.g., BERT) and Generative Large Language Models (LLMs, e.g., GPT-4, Llama).

Encoder Models (BERT) are the standard for Named Entity Recognition (NER). They are deterministic, computationally efficient, and less prone to "hallucinations". In cybersecurity, extracting a non-existent CVE ID is a critical failure; thus, the probabilistic nature of LLMs makes them risky for pure extraction tasks [33].

Models (LLMs) instead, are unrivaled for Abstractive Summarization and reasoning. They can synthesize long, technical discussions into executive summaries.

## 2.4. Research Questions

Based on the implemented analysis framework and the longitudinal dataset collected (2023-2025), this thesis addresses three specific Research Questions (RQs). These questions are designed to move beyond simple descriptive statistics and rigorously validate the strategic capability of the Argus platform to anticipate, contextualize, and attribute cyber threats.

- **Temporal Dynamics and "Time-to-Insight" (RQ1):** Can underground and unstructured sources anticipate official disclosure channels? Specifically, is it possible to quantify the gap between the first appearance of a threat artifact in informal communities (e.g., Forums, Discord) and its formal acknowledgment in Vendor Bulletins? In addition, can the system reconstruct the full Lifecycle of a campaign (e.g., Zero-Day brokerage) before it becomes a public incident?
- **Source Topology and Functional Specialization (RQ2):** How is the intelli-

gence value distributed across the ecosystem? By analyzing Exclusivity, Coverage, and Overlap, is the threat landscape hierarchical (with official sources dominating) or polycentric (with specialized sources covering distinct "Blind Spots")? Specifically, do emerging platforms like Telegram act merely as "Echo Chambers" for existing news, or have they evolved into primary sources of unique intelligence?

- **Strategic Attribution and Semantic Linking (RQ3):** Can the automated aggregation of heterogeneous data, bridge the gap between raw indicators and high-level attribution? Is it possible to empirically establish valid semantic links between Threat Actors (APTs) and their Attack Techniques (TTPs) at a scale comparable to human-curated reports, effectively transforming "Big Data" noise into actionable strategic intelligence?
- **Longitudinal Analysis (RQ4):** How does the threat intelligence landscape evolve over a multi-year window? By comparing data snapshots from 2023 to 2025, is it possible to identify structural shifts in source behavior, such as the transition of communities like Telegram from passive "Echo Chambers" to primary intelligence generators? Furthermore, how do factors like Infrastructure Decay (e.g., in Tor-based DLS) and Data Permanence (in Official Vendor Bulletins) impact the historical recoverability and long-term validity of threat artifacts?

## 2.5. Related Works

The domain of Cyber Threat Intelligence (CTI) has been extensively studied, with research primarily branching into three distinct directions: the standardization of data representation, the application of NLP for entity extraction, and the isolated analysis of specific underground communities. However, a comprehensive approach that integrates these components into a unified comparative framework remains a significant gap in the current literature. This section analyzes existing works and positions the *Argus* platform as a solution to the identified limitations.

### 2.5.1. Automated Collection from Adversarial Environments

Research on automated data collection has focused heavily on specific challenges such as Tor network latency and anti-bot evasions. Adewopo et al. [1] proposed a framework for Deep Web scraping, but their approach lacks the resilience required for modern Client-Side Rendered (CSR) applications. More recent works by ScrapingAnt [64] and Husák et al. [27] emphasize the criticality of TLS fingerprinting (e.g., JA3) in evading detection.



*Argus* advances this state of the art by integrating a "Human-in-the-Loop" (HITL) architecture [18] to bypass logical CAPTCHAs, a capability often missing in purely automated academic crawlers.

### 2.5.2. NLP for Cybersecurity: From Extraction to Semantics

This section reviews prior work on the main NLP building blocks adopted in CTI pipelines. Specifically, we summarize the state of the art on relevance classification for filtering non-CTI content, text pre-processing strategies prior to extraction, NER for cybersecurity entities, and CTI-oriented summarization techniques designed to preserve key indicators and named entities. The following subsections discuss these components separately.

#### Relevance Classification

A recurring requirement in CTI mining is an initial *relevance* (or *content selection*) stage that filters raw text streams into CTI-relevant versus non-relevant content, reducing noise before downstream extraction. In OSINT settings, early pipelines often relied on keyword-based filtering; however, this approach produces substantial false positives, motivating supervised relevance classifiers (e.g., cyber-related vs. irrelevant) as a dedicated pre-filtering step. Panagiotou et al. explicitly frame this task as content selection for CTI extraction from blogs, and show that stronger filtering improves the quality of the subsequent NER stage [52].

Outside of standard supervised setups, Le et al. explore novelty detection to identify CTI-relevant tweets, reducing dependence on negative samples [39]. More recent pipelines adopt Transformer-based filters. Arikkat et al. use a BERT-based classifier to separate threat-intelligence messages from generic Telegram content prior to large-scale IoC extraction [3].

#### Text Pre-processing before NER

In the state of the art, CTI NER pipelines typically introduce a pre-processing stage with two main goals: removing acquisition noise (e.g., HTML templates, boilerplate, banners, non-content elements) and making the text sufficiently standardized for robust downstream extraction. For instance, iACE [41] explicitly includes a preprocessing phase to reduce noise and information loss prior to IoC extraction. At the NLP level, common pipelines start from tokenization and may include POS-tagging and lemmatization to reduce lexical variability and support multi-token entity recognition [51]. In high-noise streams such as instant messaging (e.g., Telegram), additional normalizations are often

applied (e.g., lowercasing and removal of non-essential symbols) [3].

## CTI Report Summarization

CTI reports are very narrative, but they often deal with sparse, high-value facts (e.g., threat actor names, CVEs, malware families, IoCs) that analysts must retain during compression. Recent work explicitly targeted CTI-specific summarization: CTISum frames the task as generating concise highlights from heterogeneous web intelligence and introduces an "attack process summarization" angle (summaries that preserve procedural steps and technical artifacts rather than only producing a fluent overview) [54].

A key direction is to make summarization "terminology- and entity-aware", so that generation is guided toward preserving domain-critical items. RETO proposes a terminology optimization pipeline that annotates and generates CTI terminology and then uses it to steer LLM prompting, with the explicit goal of improving terminology coverage and reducing hallucinations in CTI summaries [16]. Complementary work also explores coupling summarization with information extraction signals (e.g., NER) so that detected entities act as anchors for what must appear in the final report summary [47].

## Named Entity Recognition for Cyber Threat Intelligence

Named Entity Recognition (NER) is a core NLP component for transforming unstructured CTI text (reports, posts, bulletins) into machine-actionable entities, spanning both low-level observables (e.g., IPs, URLs, file hashes) and higher-level semantic concepts (e.g., malware, threat actors, vulnerabilities). Early systems largely focused on extracting Indicators of Compromise via rule-based and pattern-driven methods, showing that many technical indicators appear in predictable textual contexts and formats [41].

Recent work adopts Transformer NER models, demonstrating strong improvements over traditional sequence taggers when applied to threat-intelligence reports. For example, Evangelatos et al. evaluated multiple Transformer backbones (e.g., BERT/RoBERTa/XLNet/ELECTRA) on DNRTI and reported that these architectures substantially improve the extraction of cybersecurity entities compared to baselines [20].

A key enabling factor has been the availability of annotated TI-NER datasets. DNRTI and APTNER are among the most widely used resources, providing multi-class label schemas tailored to CTI reports and enabling fine-grained extraction of entities such as threat actors, vulnerabilities, tools, and IoCs [20, 72, 73]. However, progress is often limited by dataset fragmentation, where corpora adopt inconsistent tag names and entity definitions. Recent work addresses this issue through STIX 2.1-aligned harmonization

pipelines that map labels onto interoperable STIX objects [48].

### 2.5.3. Comparative Analysis of Intelligence Sources

The most significant gap in the literature lies in the comparative evaluation of source value. Most studies focus on a single source type in isolation:

- **Dark Web Markets:** Nunes et al. [49] and recent related works [51] analyze the underground economy but rarely compare it with official disclosure channels.
- **Social Media Channels:** The rise of Telegram and Discord as CTI sources has been documented by Dincy R. Arikkat et al. [3] and Roy et al. [60]. However, these studies often lack a direct comparison with traditional Vendor Bulletins.
- **Metrics Definition:** Different metrics have been proposed for CTI feed quality (e.g., by Ramsdale et al. [57]), but they often focus on technical correctness rather than strategic value.

*Argus* bridges this gap by enforcing a normalized schema across all sources. Unlike previous works that analyze Telegram or Dark Web forums independently, our longitudinal analysis (2023-2025) allows for a direct comparison using various metrics such as *Time-to-Insight* [56], *Information Density* [9], and *Source Exclusivity* [40]. This enables us to empirically verify the "democratization of cybercrime" on platforms like Discord [28] and quantify the lead time of underground sources relative to official advisories.

### 2.5.4. Comparative Analysis of First-Hand Underground CTI Sources

Schröer et al.[63] studies first-hand cybercrime sources (underground forums, chat channels, and darknet websites) to understand how much actionable intelligence can be extracted and how the content and "CTI value" vary across platform types. Their work shares the core assumption that underground channels are highly noisy and that turning them into CTI requires an automated pipeline that performs (i) text normalization, (ii) relevance filtering to separate useful information from chatter and spam, and (iii) semantic analysis to summarize and compare what emerges from different sources.

A first substantial difference concerns the data and how it is obtained. In their case, the analysis is conducted on data that have already been collected and made available to the research community CrimeBB [53] for underground forums and chat channels, complemented by a separate darknet-website corpus obtained through an external monitoring

provider. In our case, the data is produced through continuous collection via Argus, that periodically acquires information from live sources and makes it available in an updated and repeatable manner for source-level analyses.

A further difference concerns the transformation pipeline and the type of output it produces. In the paper, the pipeline is used to clean and filter large text data in order to estimate what fraction of the content is truly CTI-relevant and how this proportion differs across forums, chats, and darknet websites, and provide a comparative characterization of platforms by distinguishing whether the extracted intelligence is more “technical” or more “strategic,” and by summarizing the dominant themes to highlight differences in focus and diversity across sources. In our work, instead, the pipeline is designed to build structured data from the stream collected by Argus: after cleaning and filtering, the central goal is to extract entities (NER) and normalized signals, store them with traceability, and use them as the basis for source analyses (e.g., overlap, exclusivity, density).

## 2.6. Ethical Considerations in CTI Collection

Finally, the automated collection of data, particularly from underground sources, raises significant ethical and legal questions. While "implied consent" is generally assumed for public web scraping (Surface Web), accessing criminal forums often sits in a legal and ethical gray area [6]. This thesis adheres to a rigorous ethical framework that prioritizes the "greater good" of societal protection. The collection methodology is designed to be **passive** (non-intrusive), strictly observing rate limits to respect the stability of the target infrastructure, and explicitly avoiding any active interaction with illicit markets (e.g., purchasing data or tools), limiting activities to read-only access for defensive research purposes. Furthermore, to minimize legal exposure and prevent the acquisition of potentially illegal or disturbing content, the platform is intentionally restricted to collecting text-only data, explicitly excluding images and any form of attachments or downloadable files.

# 3 | Proposed Approach and Methodology

This chapter details the conceptual framework and the architectural design underlying the **Argus** platform, addressing the limitations identified in the State of the Art, specifically in the lack of active collection capabilities in existing TIPs and the high signal-to-noise ratio of underground sources. This work proposes a methodology based on a distributed, asynchronous ETL (Extract, Transform, Load) paradigm. The following sections introduce the methodology adopted to select and validate the monitored sources, and present the high-level concepts, the data acquisition strategy, and the logic behind the transformation of raw data into actionable intelligence.

## 3.1. Sources Selection

The core value proposition of Argus CTI lies in its heterogeneous coverage. To ensure a comprehensive view of the threat landscape, the acquisition strategy must cover the entire spectrum of the internet. The considered sources have been chosen to cover the complete "Pyramid of Pain" [8], ensuring that the system captures both atomic indicators (IPs, Hashes) and higher-level adversarial behaviors (TTPs).

In this section, we define a precise taxonomy of four source categories.

### 3.1.1. Structured Technical Feeds

This category encompasses API-based sources provided by CERTs, security vendors, or open-source community projects. These sources provide high-confidence Indicators of Compromise (IoCs). As highlighted by Liao et al. [41], while technical feeds typically have high precision, they often provide limited context. In Argus, they are primarily used as machine-readable evidence to validate and correlate signals extracted from unstructured sources.

- **URLhaus**[69]: A database specialized in URLs used for malware distribution. It

supports correlation between unstructured discussions and active malicious infrastructure.

- **ThreatFox**[35]: A community-driven feed that links higher-level context (e.g., malware families and campaigns) to the corresponding IoCs, enabling the contextualization of atomic indicators.
- **MalwareBazaar**[26]: A repository of malware samples used to retrieve hashes and artifacts for detection workflows and reproducible analysis pipelines.
- **FeodoTracker**[21, 31]: An intelligence feed for Command and control botnets (e.g., Emotet/QakBot-related infrastructure), that is used to capture high-frequency C2 endpoints and support infrastructure-level correlation.
- **AlienVault OTX**[2]: A large-scale, crowdsourced CTI sharing platform that aggregates community “pulses” and IoCs, useful for capturing multi-stage campaign indicators.
- **CISA KEV**[14, 65]: Known Exploited Vulnerabilities catalog, used to activate remediation based on the evidence of exploitation available in the wild.
- **OpenPhish**[7]: A near-real-time phishing intelligence feed supporting fast identification of phishing URLs and targeted brands.

### 3.1.2. Reports and Vendor Analysis

Unstructured textual data from authoritative industry sources, including technical vendor blogs, report repositories, and official institutional documents. These sources provide the high-level context (e.g., attribution, TTPs) that is typically missing from atomic IoC feeds. Works on mining open CTI and narrative reports motivates the extraction of structured knowledge (e.g., actor–tool relationships) from these documents [41, 62, 70].

The ingestion of narrative intelligence is divided into three distinct streams based on the source format and delivery method:

- **Technical Blogs and Vendor Intelligence:** The system monitors top-tier industry leaders to capture technical analysis of malware and cyber campaigns, including **Mandiant**, **Cisco Talos**, **AhnLab (ASEC)**, **Malwarebytes**, **Symantec**, **ESET**, and **Trend Micro**. Technical research papers and vulnerability disclosures are also retrieved from **Exploit-DB** [4, 32, 43, 62].
- **Institutional and Governmental Reports:** Official security alerts and strategic reports are gathered from national and international agencies, including **CISA**,

CERT-EU, ENISA, CERT-PL, and the National Cyber Security Centres (NCSC) of the United Kingdom, Switzerland, and New Zealand [19, 57, 62].

- **Specialized News and Intelligence Repositories:** To ensure timeliness and historical depth, the system monitors specialized cybersecurity outlets (e.g., **The Hacker News (THN)** and **Security Affairs**) and curated repositories. Furthermore, the system integrates **APTnotes** and the **APT Cybercriminals Campaign Collection** to support historical attribution and longitudinal analysis [3, 55, 57, 62].

### 3.1.3. Underground Forums and Dark Web

This category covers *underground communities* (anonymity networks and restricted-access communities on the Surface Web). It includes Forums, Marketplaces, and Ransomware Data Leak Sites (DLS).

Monitoring these sources makes it possible to detect exploit discussions, initial-access sales, and planning signals before they materialize into observable attacks. While early works, such as those by Nunes et al. [49] and Paladini et al. [51], validated this methodology on now-defunct markets, Argus applies the same principles to the modern underground ecosystem.

To rigorously define the collection perimeter for underground communities, the discovery and selection of candidate forums was grounded on the *CrimeBB* dataset [53], a widely recognized academic baseline for large-scale forum monitoring. Given the high volatility of the cybercrime ecosystem, the initial target list was subjected to a strict filtering protocol prior to ingestion:

- **Linguistic Filtering:** Non-English communities were excluded to improve the performance and reliability of the adopted AI models.
- **Status:** Permanently inactive communities or seized by law enforcement (e.g., discontinued .onion addresses) were pruned from the scope.
- **Access Barriers:** Forums requiring paid membership were excluded to stick to the ethical framework of passive, non-intrusive collection.

With this filtering process, Argus successfully acquired content from the majority of the qualified targets. The communities correctly ingested data from **Dread (Tor)** and multiple Surface-Web underground forums such as **OGUsers**, **BlackHatWorld**, **Pirate Bay Forum**, **UnknownCheats**, and **Garage4Hackers**.

Since the fact that several of these platforms are not purely focused on hacking or cyber-crime, the acquisition scope was further restricted at *section* level. For each community, only sub-forums whose section title clearly indicated cybersecurity-relevant content (e.g., hacking-related discussions, programming, exploitation or tooling, operational security, credential and access abuse) were selected for the ingestion, while all remaining sections were excluded to reduce noise.

It is important to cite the fact that one candidate community could not be crawled reliably during the experimental window: Hack Forums. Unlike other targets, Hack Forums enforces particularly strict anti-automation controls at navigation and authentication layers, including aggressive browser fingerprinting and recurring CAPTCHA challenges. In addition, repeated automated access attempts lead to progressive enforcement actions: after temporary bans at the IP or session level, the platform may escalate to account-level bans. While it would be theoretically possible to mitigate this by continuously creating fresh accounts, this approach would require a substantial effort (manual overhead for registrations, verification steps, and continuous account rotation), we decided to exclude it from the collection, in order to make the platform more robust in terms of automatic-no maintenance working purpose.

To partially mitigate this limitation, the collection perimeter was extended beyond the standard CrimeBB definition by including **Wilders Security**, a prominent and highly technical Surface Web forum. Although it is a legitimate community, it provides high-signal security discussions and has been leveraged in research that mines security forums for actionable indicators (e.g., malicious infrastructure) [24], and was therefore included as a complementary source.

In the end, Argus monitors **Ransomware Data Leak Sites (DLS)**, extortion blogs used by ransomware groups (such as LockBit or Medusa) to announce victims and publish stolen data. To manage the high volatility of these sites, the platform utilizes *Ransomware.live* as its primary intelligence source, a choice also aligned with recent academic studies leveraging publicly available ransomware victim disclosure [44], tracking a database of approximately 300 unique .onion addresses. In practice, Argus ingests only the links that *Ransomware.live* flags as active at collection time, typically around 120 entries.

#### 3.1.4. Messaging Platforms

This category covers platforms like Telegram and Discord. These channels are characterized by extreme velocity. Studies like *Vulnerability disclosure in the age of social media*



[61] confirm that social signals often anticipate official CVE publication by days.

For *Telegram* Argus monitors 9 channels, including *Blue Team Alerts*, *BleepingComputer*, *Cyber Threat Intelligence*, *Pentesting News*, *Security Harvester*, *The Hacker News*, *DARKFEED*, *Dark Web Intelligence*, and *Hackmanac Cyber News*. The channels were selected by prioritizing those with the largest subscriber base and the highest visibility in Telegram search results for keywords such as "*cyber threat intelligence*" and "*hacking*", with particular emphasis on the most followed channels returned for *dark web*-related queries.

*Discord* Monitoring is extended to 29 channels distributed across key CTI-oriented servers focused on malware, exploits, reverse engineering, and intelligence, including *BlackHills*, *RootKit*, *OwlSec*, *ReverseEng*, and *HackDifferent*. The servers were selected by prioritizing the top results (in terms of popularity and member count) obtained by searching for keywords such as "*hacking*", "*exploit*", and "*malware*". Within the selected servers, the 29 channels were chosen based on the channel name, retaining only those whose titles clearly indicated relevant content (e.g., *exploit*, *malware*, *reverse engineering*, *threat intel*) and excluding off-topic areas.

Overall, Telegram sources mostly provides alerts and announcements that often link to external pages, whereas Discord sources are primarily user-to-user conversations, occasionally complemented by shared links and reposted news.

### 3.1.5. Summary of Source Selection

The following table summarizes the strategic value and the validating literature for each source category selected for Argus CTI.

| Category                        | Specific Monitored Sources                                                                                                                                                                                                                                                                                                                                    | Primary Value (CIA)                                                                                                                      | Validation                                                  |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------|
| Structured Technical Feeds      | URLhaus; ThreatFox; MalwareBazaar; FeodoTracker; AlienVault OTX; CISA KEV; OpenPhish.                                                                                                                                                                                                                                                                         | High-confidence, machine-readable IoCs (URLs, hashes, C2 endpoints, phishing targets, exploited-vulnerability catalog).                  | Liao et al. [41]                                            |
| Reports and Vendor Analysis     | <i>Technical blogs</i> (Mandiant; Cisco Talos; AhnLab/ASEC; Malwarebytes; Symantec; ESET; Trend Micro; Exploit-DB).<br><i>Institutional reports</i> (CISA; CERT-EU; ENISA; CERT-PL; NCSC UK/CH/NZ).<br><i>Specialized news</i> (The Hacker News; Security Affairs; APTnotes; APT Cybercriminals Campaign Collection).                                         | Context layer: attribution and TTP-level insights, campaign reconstruction, and actor-tool relationships missing from atomic feeds.      | Liao et al. [41]                                            |
| Underground Forums and Dark Web | <i>Dark-web forum</i> : Dread.<br><i>Surface forums</i> : OGUsers; BlackHatWorld; Pirate Bay Forum; UnknownCheats; Garage4Hackers; Wilders Security.<br><i>Ransomware DLS</i> : via Ransomware.live                                                                                                                                                           | Early exploit or access discussions, operational planning signals, and DLS victimology not available in mainstream channels.             | Nunes et al. [49]; Paladini et al. [51]; CrimeBB [53]; [24] |
| Messaging Platforms             | <i>Telegram</i> : Blue Team Alerts; BleepingComputer; Cyber Threat Intelligence; Pentesting News; Security Harvester; The Hacker News; DARKFEED; Dark Web Intelligence; Hackmanac Cyber News.<br><i>Discord</i> : BlackHills; RootKit; OwlSec; ReverseEng; HackDifferent (selected by channel title: exploit, malware, reverse engineering and threat intel). | Celerity: near-real-time alerts and link-based announcements (Telegram) plus high-density technical conversations among users (Discord). | Sabottke et al. [61]                                        |

Table 3.1: Detailed taxonomy of monitored intelligence sources, strategic value, and validating literature.

## 3.2. System Architecture Overview

This section provides a high-level view of the Argus architecture. The following subsections describe the decoupled acquisition and processing model, the main microservice clusters, and the orchestration and routing logic.

### 3.2.1. Two-Phase Decoupling

A critical architectural challenge in modern Cyber Threat Intelligence is the synchronization mismatch between data acquisition and data processing. Collection tasks are typically **I/O-bound** and prone to high latency; for instance, waiting for a Tor circuit to stabilize or a dynamic page to render requires seconds of idle time. Conversely, analysis tasks, such as Neural Network inference for Named Entity Recognition (NER), are **CPU-bound** and computationally intensive. In traditional monolithic crawler architectures, these two phases are often tightly coupled: the collector fetches a document and immediately attempts to process it before moving to the next target. This results in a *stop-and-wait* inefficiency that severely limits throughput and creates a single point of failure. To address this, the proposed approach adopts a strictly decoupled strategy, separating the workflow into two independent planes connected by a persistent buffer, following message-oriented integration principles [25].

The system logic is engineered around a **Producer-Consumer** pattern [25] involving two distinct stages:

**Collection Layer:** This layer works as a set of distributed, independent agents. Their only responsibility is to harvest data from the target sources and load it immediately into a Staging Area (Data Lake). These agents are agnostic to the downstream analysis; their objective is maximizing coverage and timeliness. This design ensures resilience: even if the analysis pipeline undergoes maintenance or experiences a computational bottleneck, the collection agents continue to buffer raw data without interruption, preventing data loss from ephemeral sources.

**Intelligence Layer:** This layer acts as the processing engine. It operates asynchronously, retrieving batches of raw items from the Staging Area and coordinating a pipeline of AI microservices. To avoid overload under bursty workloads, the system supports **backpressure management**, as formalized by Reactive Streams [59]: the processing rate can be dynamically adjusted (e.g., by scaling the number of consumer instances) independently of the ingestion rate.

### 3.2.2. Logical Clustering

To satisfy the requirements of throughput, fault tolerance, and modularity, the platform adopts an **event-driven microservices architecture**. As discussed by Dragoni et al. [17], decomposing the system into specialized services improves isolation and evolvability compared to monolithic designs, while the Staging Area provides a persistent buffer between collection and analysis.

This design provides independent scalability and resilience: the ingestion layer can be scaled to maximize coverage and timeliness, while the processing layer can be scaled (or throttled) according to available computational resources. When the analysis throughput is temporarily lower than the ingestion rate, the excess workload is safely accumulated in the persistent buffer, preserving data from transient sources and enabling the system to be caught up later. On the other side, when additional compute resources is available, the processing layer can drain the backlog, without requiring changes to the collectors.

The microservices are organized into three functional clusters, each optimized for a different workload profile.

The **I/O Bound Collection Layer** works at network edge and focuses only on acquiring content under high-latency conditions (e.g., dynamic rendering and anonymity networks), persisting raw items immediately to the Staging Area to avoid data loss.

The **CPU Bound Intelligence Layer** includes the Orchestrator and the AI microservices (Classifier, Summarizer, NER, Pre-Processing). It can be scaled horizontally to increase enrichment throughput. To protect stability under heavy workloads, each service enforces internal concurrency limits (e.g., semaphores), queueing excess requests rather than failing under load.

In the end, the **Control and Persistence Layer** ensures durability and observability. It hosts the persistent Staging Area (Data Lake) that acts as a buffer, the PostgreSQL Data Warehouse for analytics, and the control dashboard for monitoring and analysis.

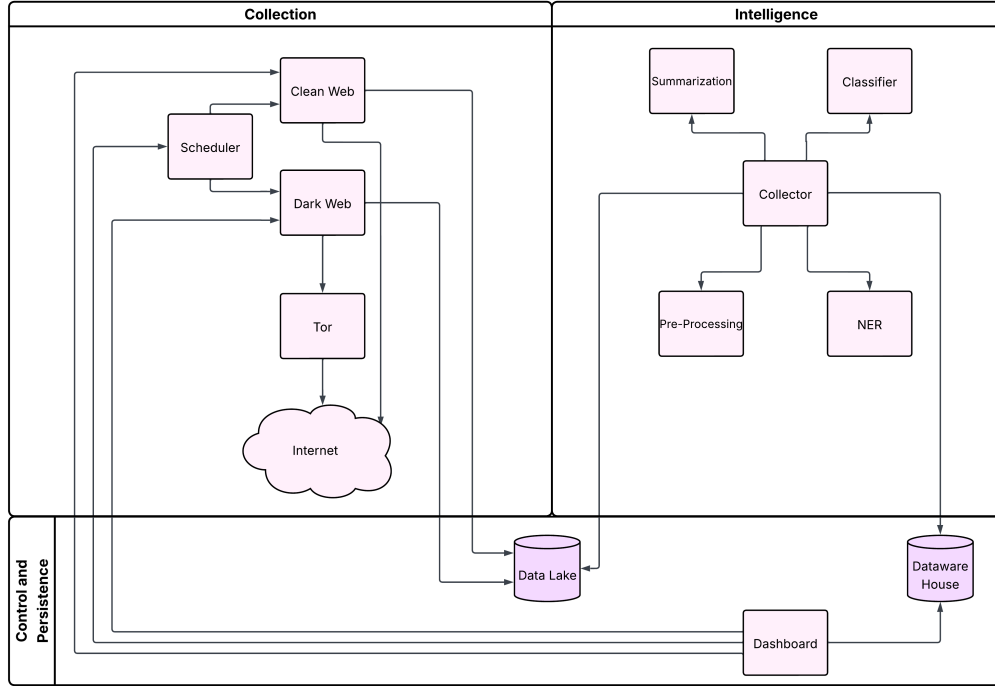


Figure 3.1: System Architecture Overview

### 3.2.3. The Asynchronous Orchestration Pattern

The central nervous system of the platform is the Asynchronous Event Loop running within the Orchestrator. Instead of processing items sequentially (Stop-and-Wait), the system adopts a high-concurrency model governed by three key design patterns.

Multiple Orchestrator instances must operate in parallel without race conditions (distributed Locking) to enable horizontal scaling (i.e., two workers processing the same document). According to Burrows’ distributed lock service model [10], the system implements an atomic check-and-set mechanism on the Data Lake to ensure that each item is claimed by at most one worker. When a worker claims a batch of items, it atomically updates their metadata, tagging them with a unique `WorkerID` and a timestamp. This guarantees strict isolation: a document is either processed by exactly one worker or remains available in the pool.

Resilience is enforced through a specific error-handling strategy, a priority dead letter queue. Documents that fail processing (e.g., due to malformed HTML or parser crashes) are not discarded but moved to a “Quarantine” collection (Dead Letter Queue). This mirrors the Dead Letter Channel pattern described by Hohpe and Woolf [25], ensuring failures are isolated from the main processing flow. The architecture implements a Priority-Based

Recovery: the Orchestrator attempts to re-process these failed items only during “Idle Cycles” (when the main ingestion queue is empty). This ensures that transient errors never block the ingestion of live, critical intelligence.

To optimize throughput for specific high-volume sources, the architecture supports logical horizontal partitioning of the workload across workers (e.g., dedicating instances to a subset of the stream such as DLS). As presented by Kleppmann [37], partitioned processing is a common strategy to handle skewed workloads in distributed systems. Specific worker instances can be configured to process only a subset of the data stream (e.g., `FILTER_TYPE="DLS"`). This allows operators to scale the infrastructure, allocating more CPU resources to a specific source type, without over-provisioning the entire system.

### 3.2.4. Content-Based Routing Strategies

Upon claiming a document, the Orchestrator applies a dynamic routing logic. Depending on the source taxonomy defined in Section 3.1, the data flows through one of six distinct processing pipelines.

### Authoritative Source Ingestion

Content from technical blogs and security news is inherently high-signal and curated by human analysts. This stream consequently bypasses the Relevance Gate in order to avoid unnecessary computational overhead and potential False Negatives on complex technical reports. The pipeline proceeds directly to an aggressive Pre-processing phase to remove web-related noise (HTML artifacts, navigation menus, and social sharing boilerplate) and normalize technical indicators. The cleaned, high-density text is then processed by the NER engine for precise extraction of threat actors, malware families, and IOCs, ensuring that all intelligence from trusted sources is preserved and correctly indexed.

### Deep Analysis

PDF Reports contain high-value TTPs but are often embedded in lengthy, unstructured text. To address this, the pipeline applies a full enrichment chain. First, PDF reports bypass the relevance gate: as curated, analyst-authored documents, they are assumed relevant by design and are immediately forwarded to the deep-analysis pipeline. Here, Abstractive Summarization utilizes a Large Language Model (LLM) to compress the narrative into a concise brief. Subsequently, the full text undergoes the Named Entity Recognition (NER) phase to extract technical indicators. This dual approach ensures that the analyst receives both the "Big Picture" and the atomic IOCs.

## Aggregated Context

Regarding forum discussions, a single forum post (e.g., "Price?") is often semantically meaningless in isolation. To address this, the system implements Thread Reconstruction logic, aggregating the main post with all associated comments into a single context window. This payload is passed to the Relevance Gate (Classifier) for filtering; if identified as relevant, it enters a Pre-processing stage that strips redundant quotes, user signatures, and UI boilerplate while de-obfuscating IOCs (e.g., converting `hxxp[.]` to `http`) and filtering low-information replies such as "thanks". Only the refined text is then sent to the NER engine for entity extraction. This pipeline, applied to both Clean and Dark Web sources, significantly reduces False Positives compared to analyzing posts atomically.

## Forensic Archival

For DLS (Data Leak Sites), the priority is not semantic interpretation but absolute factual precision regarding the analyzed actor, and its DLS domain. The pipeline bypasses the probabilistic AI models. Instead, it applies a Deterministic Rule-Based Extraction to parse the actor's name and its website directly from the post structure. The raw page is also archived permanently for forensic auditing.

## Fast-Track

Data from "Intelligence" feeds is already structured and highly reliable. The pipeline applies a **Fast-Track** pattern: these items bypass the expensive NLP and Classification stages entirely. They are normalized into the standard schema and persisted directly to the Data Warehouse. This optimization prevents the waste of GPU and CPU resources on data that is already machine-readable, maximizing the system's overall throughput.

## Community Platforms Routing

Telegram and Discord Channels data is handled through a dynamic, content-aware routing logic. Since these channels often serve a dual purpose, the pipeline splits based on the nature of the message:

If a post is identified as a shared link to an external technical site or report, the system fetches the remote content and redirects it to the authoritative source ingestion pipeline. If, instead, the content consists of native messages or discussions, it follows the aggregated context pipeline.

This hybrid approach allows the system to adapt the depth of analysis to the specific

communication style of the source.

### 3.3. The Relevance Gate Logic

A key design element of Argus is the logic introduced to mitigate the “Unstructured Chaos” that characterizes many underground sources. Raw scraping from forums typically yields a dataset dominated by noise (e.g., spam, unrelated commerce, and non-technical chatter). To improve the signal-to-noise ratio before downstream enrichment, Argus implements a **Relevance Gate**: a supervised classification stage that filters content prior to the extraction pipeline. This approach is consistent with prior work, including Paladini et al. [51], where a relevance filtering stage is adopted; in Argus, the classifier is trained on a manually labelled dataset.

#### 3.3.1. Relevance Criteria

Unlike prevalent academic methodologies that rely on static, pre-packaged datasets often outdated by several years, the training corpus for this module was actively harvested directly from the *Dread* underground forum using the custom crawling infrastructure described in Section 4. This approach ensured that the model was exposed to current slang, emerging evasion techniques, and active threat narratives.

To construct a high-quality ground truth, we selected a representative corpus of thread samples and subjected them to a rigorous manual annotation process. The labeling protocol was formalized based on strict criteria derived from the MITRE ATT&CK framework [67] and industrial CTI standards. The objective was to create a binary distinction between **Actionable Intelligence (Class 1)** and **Noise (Class 0)**.

#### Taxonomy of Relevance Class

A thread is classified as Relevant if and only if it provides technical insight into the adversary’s capability, infrastructure, or victimology. In particular, the annotation protocol requires the presence of at least one of the following dimensions:

- **Technical Indicators (IoCs)**: The text contains observable artifacts such as C2 IP addresses, file hashes, malicious domains, or specific URL patterns associated with malware distribution.
- **Adversarial Behaviors (TTPs)**: The text describes specific Tactics, Techniques, and Procedures. This includes tutorials on bypassing Endpoint Detection and Response



(EDR) solutions, persistence mechanisms, or lateral movement techniques.

- **Vulnerability Intel:** The text discusses about zero-day exploits, Proof-of-Concepts (PoC) for specific CVEs, or details regarding unpatched vulnerabilities in widely used software.
- **Criminal Economy & Trends:** The text reveals structural shifts in the cybercrime ecosystem, such as the emergence of new "Bulletproof" hosting providers, Phishing-as-a-Service (PhaaS) platforms, or changes in Ransomware-as-a-Service (RaaS) affiliate models.
- **Verifiable Leaks:** The text provides evidence of successful breaches, such as databases dumps, access logs, or exfiltrated documents, provided that they allow victim identification.

## Taxonomy of Noise Class

Content is classified as Non-Relevant if it lacks of true information value, even if the general topic is security-related. The exclusion criteria includes:

- **Generic Opinions:** Ethical debates, moral discussions, or general IT support questions (e.g., "Which VPN is best for privacy?") that do not expose adversarial TTPs.
- **Low-Sophistication Requests:** "Script-kiddie" expressing malicious intent without technical capability (e.g., "How do I hack a Facebook account?"), lacking specific details.
- **Commercial Spam:** Listings for illegal goods (e.g., drugs, credit cards, weapons) that do not reveal underlying cyber-infrastructure or new criminal patterns.

**Boundary Condition: Intent vs Capability.** A critical shade of our protocol involves the distinction between intent and capability. A thread expressing malicious intent is generally considered noise unless it inadvertently discloses novel tooling or infrastructure details useful for defensive fingerprinting. Conversely, illicit commerce is treated as noise unless it exposes a new TTP (e.g., a novel method for cashing out specific cryptocurrencies avoiding AML controls).



## 4 | Implementation and Engineering

While the previous chapter defined the conceptual methodology, this chapter details the concrete engineering realization of the **Argus CTI** platform. The system is implemented as a distributed ecosystem of 16 containerized microservices, orchestrated via Docker Compose. The codebase is primarily written in **Python 3.10**, chosen for its rich ecosystem of cybersecurity libraries and AI frameworks. The following sections describe the technical implementation of the pipeline stages: Extract, Transform, and Load.

### 4.1. The Collection Layer – Extract

The Collection Layer is engineered to handle high concurrency required for mass scanning, while respecting the constraints of adversarial environments. The implementation distinguishes structurally between active scraping of unstructured web content and autonomous polling of structured intelligence feeds.

#### 4.1.1. Scrapy Cluster Implementation

The unstructured collection engine is powered by **Scrapy** [38], an asynchronous framework built on the Twisted reactor, selected for its capability to manage thousands of concurrent requests efficiently. To ensure fault isolation and strict network hygiene, the crawlers are segregated into two distinct Docker containers: **LightCore** for Surface Web targets and **DarkCore** for the Dark Web, which routes traffic exclusively through the Tor proxy. The orchestration is primarily managed via the **Scrapyd API**, allowing the external Scheduler service to trigger specific spiders via HTTP POST requests. To complement this execution layer, the architecture integrates **ScrapydWeb**, a specialized management interface. Acting as a centralized dashboard, ScrapydWeb connects to both **scrapyd** instances to aggregate job statuses and logs. While the Scheduler handles the automated dispatching of periodic tasks, ScrapydWeb provides the necessary observability for the

system, enabling real-time monitoring of the job queues, visual log inspection, and the ability to manually trigger or terminate jobs across the distributed cluster.

## Stateful Middleware for Incremental Crawling

In order to maximize efficiency and prevent redundant network requests, the architecture implements a custom Stateful Middleware layer that enforces a strict "crawl-once" policy directly within the spider's execution loop. Unlike standard deduplication that works at database level, this mechanism intercepts the crawling process early, keeping a persistent local history of processed entities using an embedded SQLite database configured in WAL (Write-Ahead Logging) mode.

The implementation adapts its strategy based on target's nature:

For targets with stable addressing (e.g., news blogs, static sites), the middleware (MarkOnItemDeltaFetch) intercepts outgoing **Requests** before they are sent to the downloader. It computes a canonical hash of the target URL stripping volatile parameters like session IDs or tracking codes and queries the local registry. If the URL is found, the request is immediately dropped, saving bandwidth and processing time. The registry is updated only when an item is successfully scraped, ensuring that failed attempts stay available for future retries.

In dynamic environments (e.g., Tor), hidden services or social platforms (Telegram, Discord), where content is often accessed via infinite scrolling or unstable URLs, instead, the middleware (MarkOnItemDF) operates on the extracted **Items** rather than requests. It inspects the output stream of the spider, extracting the source's native unique identifier (e.g., `post_id`, `message_id`, or `tweet_id`). If this ID exists in the local history, the item is immediately discarded. This allows spiders to catch new content without re-processing older data.

This dual approach transforms spiders into incremental agents, ensuring that resources are focused exclusively on acquiring new intelligence, rather than reprocess the old one.

## Item Processing Pipelines

Once an item is extracted by a spider, it passes through a specific pipeline that changes depending on the data's origin; which performs data normalization, source-specific cleaning, and final persistence. This modular approach allows the system to apply different rules based on the source network's characteristics.

The **Surface Web Pipeline** focuses primarily on content hygiene. A representative

component, the `AsecCleanPipeline`, leverages regular expressions and the `w3lib` library to remove residual HTML tags, strip recurring promotional banners, and normalize white-spaces. This pre-processing step is appropriate for the surface web, where the extracted text is typically less ambiguous and the main source of noise is structural. As a result, the data lake is populated with cleaner, more standardized content.

The **Dark Web Pipeline**, instead, deliberately follows a raw preservation philosophy. The primary objective is to minimize information loss: content from hidden services is often irregular, where seemingly “noisy” artifacts (unusual spacing, misspellings, obfuscation) may carry intelligence value. For this reason, the collection layer avoids to use an aggressive cleaning or normalization and stores the extracted text. This does not mean that dark web content is never cleaned: the responsibility for normalization is intentionally postponed to the *AI-driven processing pipeline*, where task-specific and model-aware cleaning steps can be applied in a controlled way.

In conclusion, both clusters share the `RawItemMongoPipeline`, which serves as the gateway to the staging area. This component performs two critical tasks: *Metadata Enrichment*, injecting technical details such as the source network (CleanWeb vs. DarkWeb) and UTC timestamps; and *Idempotent Persistence*. The latter one generates a global `dedupe_key` (SHA-256) and utilizes a MongoDB Upsert with the `$setOnInsert` operator, ensuring a "Write-Once" policy where only the first captured version of an item is preserved.

### 4.1.2. Dark Web Collection Infrastructure

Access to ‘.onion’ hidden services is managed through a transparent proxy architecture. Rather than configuring proxies at the application level, which is prone to leakage, specifically, unintentional DNS exposure or direct connections bypassing the Tor circuit, the routing is enforced at the container networking level. The `DarkCore` container is configured with global environment variables (`http_proxy`, `https_proxy`) pointing to a Privoxy sidecar service on port 8118, which subsequently forwards traffic to the Tor daemon.

## Dynamic Discovery and Target Acquisition

Given the ephemeral nature of Dark Web services, static URL lists are rapidly rendered obsolete. The system implements a Dynamic Discovery logic, exemplified by the `Data Leak Sites` spider and the `Dread` forum spider. These components function as meta-crawlers that initiate their execution cycle by fetching authoritative external sources to resolve the current active `.onion` addresses before any intelligence collection begins.

Ransomware Group Enumeration process begins by querying "*Ransomware.live*" feed (`groups.json`). It parses this structured, dynamic dataset (updated by source in near-real-time) to extract the latest `.onion` mirrors of active groups (e.g., LockBit, Clop). For each valid entry, it dynamically generates a `scrapy.Request`, injecting the group's metadata directly into the crawling task. This ensures that the target scope self-updates at every execution, automatically including new groups without manual reconfiguration.

In the same way, Forum Address Resolution (Dread Spider) overcomes the instability of the main forum URL by implementing an "Auto-Discovery" mode: before attempting to access the forum, it scrapes a trusted directory service (e.g., *onion.live*) to obtain the currently verified `.onion` address. Once the active URL is resolved, the spider proceeds to initialize the crawling session, ensuring connectivity even if the forum has rotated its primary domain since the last run.

### 4.1.3. Structured Feed Integration

The feed-service is a flexible ingestion engine designed for periodic ingestion of structured intelligence sources, such as APIs, RSS feeds, and document repositories. It is managed by a central orchestrator. For technical Indicators of Compromise (IoCs) sourced from providers like ThreatFox and AlienVault OTX, the system implements a deterministic deduplication strategy: to each entry is assigned a unique ID derived from a SHA-256 hash of its source and native identifier, ensuring stable storage in MongoDB.

The service has been extended to ingest strategic reports from national CERTs (e.g., CERT-EU, NCSC) and private vendors. To handle these binary assets efficiently, the system applies an hash-based deduplication logic; by computing the SHA-256 hash of a document's content before final persistence, the service identifies and discards redundant copies of the same report, often cross-published by different agencies.

### 4.1.4. Instant Messaging Platforms

Differently from standard web scraping, these modules require specialized API interactions and state management to navigate the continuous streams of chat history effectively.

**Discord Monitoring** The Discord collection module is designed to monitor a curated list of invited servers focused on reverse engineering, malware analysis, and incident response. Instead of simulating a user's browser, the crawler interacts directly with the **Discord Gateway API (v9)**. Authentication is handled via a user token injected securely at runtime. The spider iterates through a predefined list of Channel IDs, executing

authenticated GET requests to the `/messages` endpoint. To ensure efficiency and prevent data duplication, the crawler implements a local state mechanism (using SQLite) that tracks the `last_message_id` processed for each channel. This allows the system to perform incremental “catch-up” cycles, fetching only the messages posted since the previous execution window, effectively transforming the chat stream into a structured timeline of intelligence events.

**Telegram Monitoring** Telegram module addresses the challenge of monitoring public and private channels often used by hacktivist groups and cybercrime actors. Due to the platform’s robust anti-scraping measures, this crawler is built upon the *Telethon* library, which is an asynchronous Python wrapper for the Telegram `MTPROTO` API. The authentication flow requires a valid `api_id`, `api_hash`, and a persistent session string stored in a file. Unlike the REST-based approach of Discord, the Telegram crawler establishes a direct, encrypted session with the Telegram data centers. It iterates through target channels and utilizes Telethon’s `GetHistoryRequest` to retrieve message batches. Similar to the Discord module, it employs a local “sync” database to record the highest `message_id` seen. This ensures that subsequent runs only retrieve the “delta” of new conversations, minimizing bandwidth usage and respecting Telegram’s strict rate limits.

#### 4.1.5. Data Lake Implementation

The staging area for raw data is implemented on **MongoDB** (v6.0). The choice of a NoSQL Document Store over a Relational Database for this specific layer is justified by three engineering requirements.

The system requires **Schema Flexibility** (Schema-on-Read) to handle the variable nature of the data; the structure of a scraped Dark Web HTML page differs radically from a JSON object returned by the CISA API, and MongoDB allows for the ingestion of these disparate structures into a single collection without complex migrations.

The ingestion phase is write-heavy, and MongoDB’s ability to defer index updates combined with its efficient BSON storage format provides superior insertion performance compared to strictly transactional SQL systems during high-velocity crawls.

Finally, the platform leverages MongoDB’s Time-To-Live (TTL) indexes to automatically manage storage costs by expiring and purging raw HTML artifacts older than a configured retention period, automating data lifecycle management.

## 4.2. Intelligence Engine (Transform)

The Intelligence Engine represents the computational core of the platform, where raw data extracted from the Data Lake is transformed into structured, actionable intelligence. To satisfy the latency requirements of processing high-volume data streams, the architecture has been fully migrated to a GPU-Accelerated Stack, utilizing the NVIDIA CUDA ecosystem to offload the massive matrix operations required by Transformer models.

This transformation process is not static, it changes depending on the type of data taken into consideration. The process is managed by a centralized orchestrator known as the **Collector**, which acts as the functional bridge between storage and computation. While the Collector handles the data flow and task prioritization, the heavy lifting of inference is distributed across a suite of specialized microservices specifically **NER**, **Classifier**, **Summarizer**, and **Preprocessing**. These services share a common deployment infrastructure based on **FastAPI** exposed via **Uvicorn**, and are containerized using the NVIDIA Container Toolkit to allow direct hardware passthrough for near-native performance. It is worth noting that the output evaluations of the **NER** and **Classifier** services are detailed in Chapter 5, where the focus shifts to the quantitative results and performance of the models, evaluating most relevant metrics (eg. F1-score, PR-AUC), depending on the specific achieved task.

### Resource Management and Asynchronous Inference

Deep learning models require a delicate balance between maximizing Video RAM (VRAM) usage and maintaining high API responsiveness. Argus architecture addresses this challenge by decoupling network management from heavy computational loads.

At the infrastructural level, the **Docker Compose** configuration ensures that the intelligence cluster has direct access to the host's NVIDIA drivers. The application logic employs a **Semaphore Guarded Inference** mechanism, effectively implementing a resource throttling strategy [13] to prevent VRAM exhaustion. The system strictly limits active tasks using the **MAX\_CONCURRENT\_INFERENCE** variable, which is calibrated specifically to the available hardware capabilities.

While the **FastAPI** application manages HTTP requests asynchronously, blocking GPU operations are discharged via **asyncio.to\_thread**. This ensures that the event loop remains free to accept and queue new incoming connections, even when the GPU is fully saturated with tensor processing.



## NLP Service Architecture

The NLP module architecture is structured as a sequential pipeline of independent microservices, designed to ensure scalability and efficient hardware resource management. The processing logic is partitioned into three distinct functional components, allowing for strategic separation of deterministic normalization tasks (CPU-bound) from intensive neural inference operations (GPU-bound). In particular, the system comprises the services described in the following sections.

**Preprocessing Service.** The Preprocessing service is structured as a modular normalization engine operating within a CPU environment. Its internal architecture centers on the `PreprocessingEngine` class, which acts as an orchestrator for three core components. The first is the *Knowledge Base Manager*, responsible for loading and persisting in memory the heuristic rules defined in the configuration file `knowledge_base.yaml`. The second is the *Text Normalization* module, a collection of deterministic procedures for text cleaning, such as HTML tag removal, Unicode NFKC normalization, and whitespace management. Finally, the *Keyword Matching* module performs comparisons between normalized text and the Knowledge Base dictionaries, allowing for preliminary metadata enrichment without consuming GPU computational resources.

**Named Entity Recognition (NER) Service.** The NER service adopts a hybrid architecture within the `NER_Engine` class, designed to combine contextual analysis with rigid pattern extraction. The structure articulates into two parallel branches coordinated by an aggregation module. The neural branch manages the lifecycle of a Transformer model (based on the RoBERTa architecture [42]), encapsulating the tokenizer and neural network weights for the semantic identification of complex entities. Conversely, the deterministic branch consists of a Regular Expression (Regex) engine optimized for extracting fixed-format structured data, such as IP addresses, file hashes (MD5, SHA256), CVE identifiers, and URLs. A final *Entity Aggregation* module receives outputs from both branches, applying deduplication and conflict resolution logic to generate a normalized JSON payload.

The neural branch utilizes the `xlm-roberta-large` architecture [12, 42], which was selected following an extensive comparative analysis against alternative models such as `xlm-roberta-base` and `DarkBERT`. While the *base* variant offered reduced inference latency, it failed to achieve the necessary granularity for the complex 61-entity (BIOES prefixes included) classification schema, frequently misidentifying specialized cybersecurity labels. Conversely, despite `DarkBERT` specialized pre-training on dark web data, it demonstrated limited generalization capabilities when applied to the specific linguistic

structures of the DNRTI and APTNER datasets. These datasets, which form the core of our training and evaluation, consist of formal technical reports and highly structured indicators that differ significantly from the informal communications found in dark web environments. Consequently, `xlm-roberta-large` was selected as the optimal backbone for its ability to reconcile formal security prose with technical indicator extraction.

The model is fine-tuned on a specialized dataset (Sec 5.2.1) containing 61 distinct entity classes (including BIOES prefixes). To address the inherent class imbalance, where critical labels such as *SHA1* or *VULID* are rare, we implemented a targeted *oversampling* strategy: minority classes were replicated 3 times before tokenization, in order to ensure sufficient model sensitivity.

We can summarize the implementation details and architectural parameters as follows:

Training was conducted for 15 epochs with a learning rate of  $2 \times 10^{-5}$  and a *cosine* scheduler (0.1 warmup ratio). We applied a weight decay of 0.05 and a *label smoothing* factor of 0.1 to maximize robustness against potential noise in the ground truth.

To accommodate memory requirements of the large architecture, the system utilizes *Automatic Mixed Precision* (FP16) and *Gradient Accumulation* with 8 steps. This configuration leads to an effective batch size of 32, ensuring high-performance inference while staying within the hardware VRAM limits.

We developed an early stopping mechanism with patience of 3 evaluation rounds. The selection of the final weights was driven by the F1-score reported through the *segeval* framework, ensuring the best balance between precision and recall across all 61 classes.

**Classification Service.** The Classification service structure is defined by the *Classifier-Inference* class, which operates as a Singleton to ensure that the neural model stably occupies the assigned Video RAM (VRAM), preventing expensive reloading overhead between requests.

The core of the classification logic is a Stacking Ensemble architecture [68, 74], designed to mitigate the variance of the single-model predictions when dealing with the high-dimensional feature space of CTI reports. The ensemble integrates two distinct predictors: a neural branch based on the `microsoft/deberta-v3-small` transformer and a statistical baseline utilizing a `LinearSVC` with TF-IDF vectorization. This hybrid approach was selected to combine the deep semantic understanding of the transformer, which captures complex contextual dependencies, with the robustness of the linear support vector classifier in detecting explicit keyword-based patterns.

The neural model, `deberta-v3-small`, was fine-tuned using a stratified 5-fold cross-

validation strategy to ensure generalization stability across different data distributions. The training configuration employed a learning rate of  $2 \times 10^{-5}$  with a cosine annealing scheduler and a warmup ratio of 0.1 over 10 epochs. To optimize computational resources without compromising the effective batch size, we used Gradient Accumulation with a factor of 4, with a base batch size of 8, resulting in an effective batch size of 32. Mixed-precision training (FP16) was adopted via `torch.cuda.amp` to accelerate the forward and backward passes while reducing the memory footprint on the GPU.

The outputs from both the neural and linear branches, in particular the calibrated probability distributions, are then concatenated and fed into a Logistic Regression meta-learner. This second-level classifier is responsible for learning the optimal weighting schema between the two models, effectively correcting individual biases, based on the specific considered situation.

#### 4.2.1. Custom Sliding Window Algorithm

In order to address the input length limitations of transformer architectures, typically capped at 512 tokens [15], the system implements a custom Sliding Window strategy in the *NERPipeline* Python class [5]. Instead of simple truncation, the algorithm segments the input text into overlapping chunks governed by a window size of 400 tokens and a stride of 80 tokens. This overlap ensures that entities located at the boundary of a chunk are not split, preserving their semantic context.

The logic behind the sliding window algorithm, is formalized in Algorithm 4.1.

---

##### Algorithm 4.1: Token-Based Sliding Window with Stride

---

**Require:** Document  $D \neq \emptyset$ , Tokenizer  $\mathcal{T}$ , Window size  $w \in \mathbb{N}^+$ , Stride  $s \in \mathbb{N}^+$  with  $s < w$

**Ensure:** Set of text chunks  $\mathcal{C} = \{(c_i, \delta_i)\}_{i=1}^{|\mathcal{W}|}$  where  $c_i$  is chunk text and  $\delta_i$  is global char offset

1:

##### Step 1: Tokenization with Offset Mapping

2:  $\mathcal{O} \leftarrow \mathcal{T}.\text{encode}(D)$   $\triangleright \mathcal{O} = [(o_1^s, o_1^e), \dots, (o_n^s, o_n^e)]$  where  $o_i^s, o_i^e \in [0, |D|]$

3:  $n \leftarrow |\mathcal{O}|$

4:

##### Step 2: Window Index Generation

5:  $\text{step} \leftarrow w - s$   $\triangleright$  Effective window advancement

6:  $\mathcal{W} \leftarrow \emptyset$

```

7:  $i \leftarrow 0$ 
8: while  $i < n$  do
9:    $j \leftarrow \min(n, i + w)$ 
10:   $\mathcal{W} \leftarrow \mathcal{W} \cup \{(i, j)\}$ 
11:   $i \leftarrow i + \text{step}$ 
12: end while
13:

```

### Step 3: Token-to-Character Span Conversion

```

14:  $\mathcal{C} \leftarrow \emptyset$ 
15: for each  $(i, j) \in \mathcal{W}$  do
16:    $\alpha \leftarrow o_i^s$  ▷ Character start: first token's start offset
17:    $\beta \leftarrow o_{j-1}^e$  ▷ Character end: last token's end offset
18:    $c \leftarrow D[\alpha : \beta]$  ▷ Extract substring
19:    $\mathcal{C} \leftarrow \mathcal{C} \cup \{(c, \alpha)\}$ 
20: end for
21: return  $\mathcal{C}$ 

```

---

### Notation:

- $\mathcal{O}[i] = (o_i^s, o_i^e)$ : offset mapping for token  $i$ , where  $o_i^s$  and  $o_i^e$  are character indices
- $w$ : maximum tokens per window (typical: 400-512)
- $s$ : stride, i.e., token overlap between consecutive windows (typical:  $w/5$  to  $w/4$ )
- $\text{step} = w - s$ : effective window advancement in tokens
- $(i, j) \in \mathcal{W}$ : token-level window boundaries ( $i$  inclusive,  $j$  exclusive)
- $(\alpha, \beta)$ : corresponding character-level span in  $D$
- $\delta_i = \alpha$ : global character offset for chunk  $c_i$ , enables local-to-global coordinate mapping

### Key Properties:

- **Complete Coverage**: Every token in  $D$  appears in at least one window.
- **Controlled Overlap**: Windows adjacent to each other  $\omega_k$  and  $\omega_{k+1}$  share exactly  $s$  tokens.
- **Boundary Recovery**: Spanning window boundaries of entities are captured in overlapping regions.

- **Coordinate Alignment:** Global offsets  $\{\delta_i\}$  enable the mapping of chunk-local predictions to document-global positions.

**Example:**

- Input:  $n = 1000$  tokens,  $w = 400$ ,  $s = 80$
- Output: step = 320,  $|\mathcal{W}| = 4$  windows
- Windows:
  - $\omega_1 : [0, 400)$
  - $\omega_2 : [320, 720)$
  - $\omega_3 : [640, 1000)$  (*Reaches end*)
  - $\omega_4 : [960, 1000)$  (*Residual tail coverage*)
- Overlaps: Tokens  $[320, 400)$  shared by  $\omega_1$  and  $\omega_2$ .

**Complexity:**

- **Time:**  $O(n)$  for tokenization +  $O(|\mathcal{W}|) = O(n/\text{step})$  for window extraction
- **Space:**  $O(|\mathcal{W}| \cdot w \cdot \bar{k})$  where  $\bar{k}$  is average characters per token
- **Number of Windows:**  $|\mathcal{W}| = \lceil \frac{n}{w-s} \rceil$

**Comparison with Standard Approaches:**

| Approach              | Coverage                   | Boundary Handling                      |
|-----------------------|----------------------------|----------------------------------------|
| Fixed Truncation      | Partial (first $w$ tokens) | Poor (entities cut off)                |
| Hierarchical Chunking | Complete                   | Medium (chunk-level only)              |
| <b>Sliding Window</b> | <b>Complete</b>            | <b>Excellent (token-level overlap)</b> |

The inference results are subsequently aggregated using a custom post-processing logic designed to resolve overlaps. If the same entity is detected in two overlapping windows, the instance with the higher confidence score is retained; if an entity is split across chunks, the logic reconstructs the full token span based on positional indices. Finally, the pipeline injects deterministic Regex matches for rigid indicators (e.g., CVEs, IPs), merging them with neural predictions to maximize precision.

### 4.2.2. Generative Summarization

The Summarization Service is powered by Llama-3.2 [46] and served via `llama-cpp-python` with CUDA/cuBLAS support. In our deployment we load the GGUF checkpoint `Llama-3.2-3b-instruct-q4_k_m.gguf` (quantization `Q4_K_M`), chosen to reduce the memory footprint while preserving sufficient quality for technical summarization. We use a 16K context window (`n_ctx=16384`) to accommodate long CTI reports with fewer splits and less cross-chunk coherence loss, and enable GPU offload (`n_gpu_layers=30`) to keep most computations on the GPU and avoid CPU-GPU transfer overhead. Generation is capped at 2048 new tokens to bound latency and prevent overly verbose outputs, while conservative decoding (`temperature=0.1`, `top_p=0.9`, `repeat_penalty=1.1`) is used to discourage hallucinations and repetition, prioritizing factual, CTI-dense summaries.

## Map-Reduce Architecture

CTI reports often exceed the context window of the deployed model. Simple truncation is not acceptable because it may discard critical details (e.g., IoCs or CVE references) that can appear anywhere in the text. To address this, the summarization component (module `summarizer_engine.py`, class `CTISummarizer`) implements a Map-Reduce style pipeline using `LangChain` with a local `LlamaCpp` backend.

The Phase 1 consist of Recursive splitting (character-based). The document is split with `RecursiveCharacterTextSplitter`, using environment-configurable parameters (`SUM_CHUNK_SIZE_CHARS` and `SUM_CHUNK_OVERLAP_CHARS`; defaults 20000 and 800). The splitter prioritizes natural separators (e.g., paragraphs, newlines, spaces), while overlap preserves local redundancy to reduce context loss across adjacent chunks. The Phase 2 is relative to the Map (extraction) part. Each chunk is processed independently with an extraction prompt (`DEFAULT_MAP_PROMPT`, method `_map_summarize_chunks`) to produce an intermediate high-fidelity “CTI brief” focused on technical facts (actor, tooling, TTPs, impact, IoCs). Although chunks are conceptually parallelizable, the current implementation processes them sequentially to ensure stability and resource control. Then we proceed with the Phase 3: Iterative Reduce (synthesis). Briefs are reduced in batches (parameter `SUM_REDUCE_BATCH_SIZE`, default 12) using the synthesis prompt `DEFAULT_REDUCE_PROMPT` (method `_reduce_briefs_once`). Reduction is iterated until a single text remains (method `_reduce_until_one`) or until a maximum number of rounds is reached (`SUM_MAX_REDUCE_ROUNDS`, default 12). If multiple segments remain after the limit, a final compression step concatenates the remaining segments and produces a final summary.

## Prompt-Based Hallucination Mitigation Strategy

To mitigate hallucinations and over-generalization during compression, the pipeline uses two distinct prompts, one for Map and one for Reduce, with explicit negative constraints. The core idea is to separate a first extraction stage from a second synthesis stage, consisting in compress, without losing “hard” technical data.

The Map prompt enforces an information-extraction behavior: it forbids inventing facts and requires preserving specific technical details (e.g., CVE IDs, dates, volumes, tools). It also enforces a terse output style with no generic introductions and no bullet lists.

```

1 Rules:
2 - Do NOT invent facts. Use only what is explicitly present in the chunk.
3 - CRITICAL: You MUST PRESERVE specific technical details:
4   * CVE IDs (e.g., CVE-2023-XXXX)
5   * Exact Dates and Timelines
6   * Data Volumes and Exfiltration tools
7   * Attack Vectors and Protocols
8 - No bullet lists. No introductions like "Here is the summary".

```

**Listing 4.1:** The 'Map' Prompt: Extraction Constraints

The Reduce prompt controls how briefs are synthesized and helps prevent semantic dilution by requiring the preservation of hard data (CVEs, dates, volumes, and specific malware or tooling names) while explicitly discouraging vague generalizations that replace concrete values. The output is kept compact and organized as a narrative flow, producing a summary that provides clearer context for the subsequent pipeline components.

```

1 DEFAULT_REDUCE_PROMPT = """
2 You are a Cyber Threat Intelligence (CTI) analyst.
3 You will receive multiple CTI briefs. Produce ONE final concise summary,
4   but maintain technical density.
5 Rules:
6 - Do NOT invent facts.
7 - PRESERVE HARD DATA: Ensure that CVEs, Dates, Data Volumes, and
8   specific Malware names found in the briefs are included in the final
9   summary.
10 - Do not generalize specific details.
11 - No bullet lists. No generic introductions.
12 - Prefer a compact narrative: who/what/when/how/impact/mitigations.
13 """

```

**Listing 4.2:** The 'Reduce' Prompt: Synthesis Constraints

## Deterministic Output Sanitization

Instruction-tuned models may introduce conversational prefixes or filler (e.g., “here is the summary”, “as an analyst...”), which reduces technical density and introduces noise in downstream storage. To ensure a professional and machine-parsable output, the system applies deterministic post-processing via `_clean_llm_output`: a set of regex rules removes only common conversational patterns when they appear at the beginning of the text, then performs minimal cleanup of quoting and normalizes whitespace and newlines. This keeps both intermediate briefs and the final summary consistent and CTI-dense.

## 4.3. Persistence and Observability (Load)

The final stage of the pipeline addresses the storage of structured intelligence and the real-time monitoring of the infrastructure. The implementation follows a pattern of strict separation between the transactional write-path and the analytical read-path.

### 4.3.1. Hybrid Data Warehouse

Structured intelligence is stored in the Data Warehouse (PostgreSQL 15), accessed via the `asynccpg` driver. The database design follows a hybrid relational model: the schema uses standard Third Normal Form (3NF) for core entities and relationships, while leveraging the `JSONB` data type to store heterogeneous source metadata without overly rigid constraints.

The architecture relies on three primary pillars: the `intel_items` table serves as the central fact repository for processed intelligence; the `extracted_entities` table acts as a normalized registry for unique IoCs and threat actors, enabling many-to-many pivots; finally, a set of Knowledge Base tables (`kb_*`) maps static definitions to the MITRE ATT&CK framework. This integration is particularly useful for alias resolution.

The detailed structure of the Data Warehouse schema is available in appendix B. This design leverages PostgreSQL’s GIN (Generalized Inverted Index) capabilities to allow high-performance querying on nested JSON fields, such as searching for a specific spider name within metadata, with performance comparable to native text columns. Furthermore, the pre-computed `TSVECTOR` column enables full-text mining across the entire intelligence corpus without external search engines.



### 4.3.2. Control Plane Architecture

The Argus Control Plane is the central command hub for the platform, it is the interface between the human analyst and the underlying collection infrastructure. To ensure scalability and maintainability, the architecture is strictly split into two decoupled layers: a presentation layer responsible for user interaction and visualization, and a data orchestration layer that manages logic and data retrieval.

#### Presentation Layer

The user interface is implemented as a Single Page Application (SPA) with a client-side rendering approach that keeps the backend logic fully decoupled. The stack is based on React 18 and TypeScript. The consistent use of static typing helps enforce API data contracts and reduces the risk of runtime errors.

To keep the frontend lean, the architecture avoids heavy external state-management libraries, relying instead on the native React Context API, together with a Vite build stack and Tailwind CSS for utility-first styling. For production, the application is packaged via Docker multi-stage builds and served as static assets through a lightweight Nginx instance, which also handles client-side routing.

Functionally, the application is organized into two primary contexts:

The General Dashboard tracks the global state of the spiders, providing real-time monitoring of the crawling infrastructure. The Intelligence View interface, that renders the structured output of the analysis pipeline, is dedicated to data consumption.

#### Data Orchestration Layer

The Dashboard API is a **FastAPI-based** asynchronous gateway acting as a unified aggregation layer, architecturally split to handle two distinct data domains.

**1. Infrastructure Aggregation Engine.** Acting as a *federated state aggregator*, this component monitors distributed crawlers by querying Scrapy and Scheduler instances in parallel via `asyncio`. Since Scrapy does not expose granular metrics, it includes an optimized log stream parsing engine that fetches only the tail of execution logs via HTTP *Range* requests and extracts real time metrics with Regex. A persistent JSON-based filesystem cache stores statistics for completed jobs, avoiding redundant processing overhead.

**2. Scientific Data Access Layer.** This layer manages intelligence retrieval via `asynpg`

connection pooling. It adopts an *In-Database Processing* strategy, using complex SQL Common Table Expressions (CTEs) to dynamically normalize entities (e.g., alias mapping) directly within the query execution. System stability is enforced through Concurrency Control semaphores that limit heavy analytical workloads, while search operations leverage PostgreSQL’s native Full-Text Search capabilities, avoiding external dependencies.

## 4.4. Security and Human-in-the-Loop

Navigating the adversarial landscape of the Surface and Dark Web requires sophisticated resilience against anti-scraping countermeasures. To guarantee continuous data acquisition, the architecture employs a multi-layered evasion strategy that addresses network blocking, dynamic rendering, and identity fingerprinting through the following specialized mechanisms.

### Adaptive Ban Mitigation and Backoff Middleware

To improve the resilience of the collection engine against anti-bot protections and rate limiting, the system includes an Adaptive Ban Mitigation Middleware (`ban_backoff`). This component acts as an automated defensive layer that monitors the health of crawling sessions in real time.

The middleware combines a small set of defensive mechanisms. First, it performs heuristic block detection by checking the downloader output for common triggers, including HTTP status codes such as `403 Forbidden` and `429 Too Many Requests`, as well as HTML patterns associated with CAPTCHA challenges or “Access Denied” pages. When a block is detected, it applies an exponential backoff strategy: rather than letting the spider fail, it intercepts the response and schedules an increasing delay before the next attempt, effectively cooling down the IP reputation. In addition, within the `scrapyd_tor` environment, the middleware integrates with the Tor proxy to trigger a circuit/session refresh (IP rotation) when blocking persists.

### Dynamic Content Handling

A critical implementation challenge for Surface Web sources, such as the Google Cloud Threat Intelligence Blog, is the prevalence of Single Page Applications (SPAs) where content is rendered client-side via JavaScript. Standard HTTP requests fail to retrieve the DOM in these scenarios. To address this, the system integrates **Scrapy-Playwright**.

Unlike Selenium, Playwright operates asynchronously, communicating directly with the browser's DevTools Protocol. The implementation leverages a specialized `DownloadHandler` that launches a headless Chromium instance; the spider logic specifically instructs the browser to wait for essential DOM mutations such as the appearance of the main content article container before passing the fully rendered HTML back to the Scrapy engine for parsing.

## TLS Fingerprint Randomization

The implementation replaces the standard Python SSL context with "Scrapy-Impersonate". in order to bypass anti-bot protections implemented by third-party providers. This library utilizes a modified boringssl binding to emulate the exact TLS handshake signatures, including specific Cipher Suites and Extension orders, of a legitimate Chrome browser. This technique effectively prevents the crawler from being flagged and blocked based on its JA3 network fingerprint.

## Human-Like Crawling Behavior

To evade detection by advanced anti-bot systems and respect the operational stability of target servers especially fragile hidden services the crawling engine enforces a strict Human-Emulation Policy. Instead of aggressive, machine-speed request bursts, the spiders are engineered to mimic the browsing patterns of a human analyst.

The strategy relies on stochastic delays (jitter). In particular, all spiders implement a randomized `DOWNLOAD_DELAY`: instead of using a fixed interval (e.g., exactly 2.0 seconds), the scheduler injects a Gaussian “jitter” (typically in the order of  $\pm 0.5$  to 1.5 seconds) between consecutive requests, reducing the predictable timing fingerprints that firewalls often use to flag automated scripts. In addition, the system applies context-aware pauses in long-running sessions. For example, after processing a batch of 50–70 messages, the **Discord Crawler** triggers a random “coffee break” sleep interval (50–70 seconds), mimicking a user reading content before scrolling further. Similarly, the **Dread** spider enforces a `_human_pause` between thread transitions to keep the concurrency profile low.

## Human-in-the-Loop Warmup Strategy

Many targets, particularly in the Dark Web (e.g., Pirates Forum) and protected Surface Web platforms (e.g., BlackHatWorld), perform advanced anti-bot protections like Cloudflare Turnstile, graphical CAPTCHAs, or mandatory login walls. To bypass these barriers without relying on unreliable third-party solving services, the system implements

a "Human-in-the-Loop" Warmup Strategy.

This mechanism leverages a dedicated Selenium Grid container equipped with a noVNC interface, allowing the operator to visually interact with the remote browser session. The workflow is orchestrated through a sequence of distinct phases. It begins with Detection & Trigger: when a spider detects a blocking condition such as an HTTP 403 response or specific HTML keywords like "Just a moment" it automatically triggers the `warmup_selenium.py` routine.

After that, the process moves to visual interaction. The system pauses the automated flow and logs a critical alert containing a URL to the VNC dashboard. The operator can connect to this interface and manually perform the required steps, such as solving the challenge or logging in with valid credentials. Once the operator completes the interaction, the session-harvesting phase begins: the script automatically exports the session cookies and the browser User-Agent string to a persistent JSON file (e.g., `cookies.json`).

Finally, via Seamless Injection, the spider reloads these fresh credentials into its Scrapy session context. For subsequent requests, it spoofs the browser's identity accurately, bypassing the protection checks by presenting a valid, human-generated token.

## 5 | Experimental Results and Analysis

Following the architectural design and engineering implementation detailed in the previous chapters, this chapter presents the empirical validation of the Argus platform. The primary objective of this analysis is to quantify the system's efficacy not merely as a data collection tool, but as a strategic asset capable of generating actionable intelligence. This is achieved by systematically comparing the evaluated sources according to the metrics defined in 5.3. Specifically, the evaluation applies different metrics to assess the semantic quality, timeliness, and strategic value of the collected data sources. The analysis answers the Research Questions (RQs) defined in Section 1 by comparing heterogeneous source categories, from unstructured Dark Web forums to structured OSINT feeds, under a unified normalized schema.

The experimental campaign was conducted over a continuous window of one month (January 2026), in which the infrastructure was able to collect and structure data from January 2023 to January 2026. During this period, the system ingested and processed a heterogeneous corpus of approximately 500,000 potential threat artifacts. This longitudinal dataset provides the statistical significance necessary to compare source behaviors and validate the system's resilience under real-world conditions. All experiments were executed on a dedicated server equipped with an Intel(R) Xeon(R) Gold 6418H CPU (32 physical cores, 32 threads), 64 GB of RAM, 2 TB of allocated storage, and an NVIDIA L40 GPU to accelerate the AI enrichment pipeline.

To guarantee replicability of the results and address potential biases inherent in comparing diverse data formats (e.g., concise bulletins, verbose reports), we established a strict validation framework.

## 5.1. Dataset Composition

The analysis is performed on a dataset stratified into seven primary source categories: *Structured Feeds (Intelligence)*; *Vendor, Institutional, and Government Reports (Report)*; *Cybersecurity Related Bulletins (Bulletin)*; *Dark Web and Clean Web Forums (Forum)*; *Social Community (Telegram and Discord)* and *Ransomware Groups' Data Leak Sites (DLS)*. The ingestion pipeline remained active 24/7, with no manual filtering applied prior to the database persistence, ensuring that the analyzed corpus reflects the true noise levels of the monitored environments.

## 5.2. AI Component Validation

This section presents the experimental validation of the individual Artificial Intelligence components integrated into the Argus CTI pipeline. Before discussing the aggregate intelligence value (Section 5.3), it is essential to demonstrate the reliability of the underlying extraction and classification engines.

### 5.2.1. Named Entity Recognition (NER) Performance

The extraction capability was evaluated using a fine-tuned XLM-RoBERTa model [12]. The evaluation focused on measuring the model's ability to distinguish between semantic entities (e.g., Threat Actors) and technical indicators (e.g., Hashes), providing the empirical justification for the adoption of the Hybrid Architecture.

### Dataset and Training Setup

The model was trained on a dataset derived from the unification of the **DNRTI** and **APTNER** datasets [48]. The resulting dataset consists of 17,568 sentences, annotated using the BIOES (Beginning, Inside, Out, End, Single) scheme [58].

The data was partitioned into strict subsets:

- **Training Set:** 14,054 sentences (80%).
- **Validation Set:** 1,756 sentences (10%).
- **Test Set:** 1,758 sentences (10%).

A deep analysis of the training data revealed a significant class imbalance, with the 'O' (Outside) tag representing approximately 82.2% of total tokens. Among the entities of interest, classes such as 'S-APT' (1.45%) and 'S-MAL' (1.08%) were relatively frequent,

whereas technical identifiers like ‘S-SHA1’ (19 instances) and ‘B-ENCR’ (3 instances) were extremely rare. To mitigate this, a Targeted Oversampling strategy was applied to the minority classes during the training phase, artificially increasing the training set size to 14,200 sentences.

The token-level analysis showed an average sentence length of 24.7 tokens, and 99.8% of sentences were shorter than 128 tokens. Based on this, we set the model’s maximum sequence length to  $L = 130$ , which improves inference speed while preserving virtually all the content in the dataset.

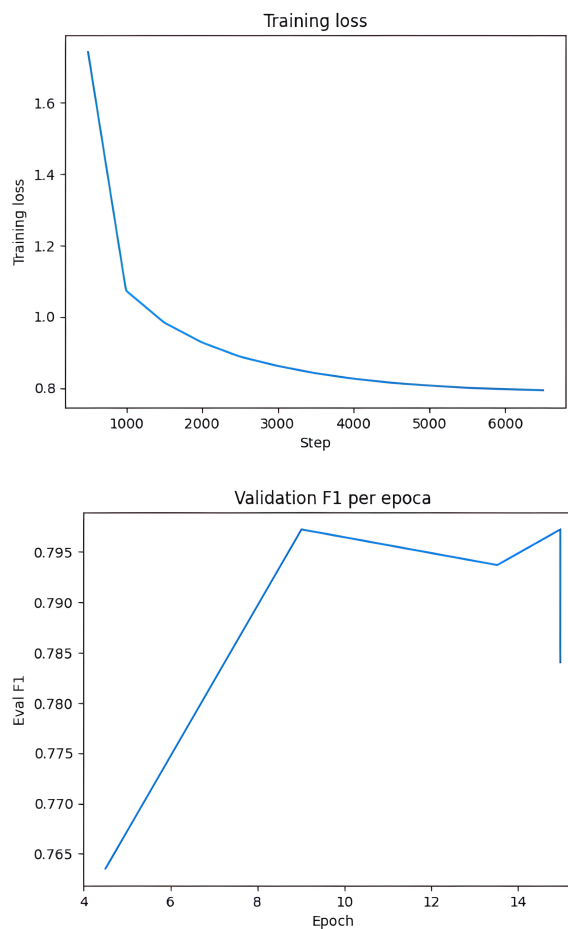


Figure 5.1: Training and Validation Loss curves over 15 epochs.

## Quantitative Results (Neural Baseline)

The model was evaluated on the held-out Test Set after 15 training epochs. The global performance metrics are reported below:

| Metric                     | Value (Test Set) |
|----------------------------|------------------|
| Eval Loss                  | 0.9604           |
| Accuracy                   | 91.99%           |
| Precision                  | 77.11%           |
| Recall                     | 79.74%           |
| <b>F1-Score (Weighted)</b> | <b>79.60%</b>    |

Table 5.1: Global performance metrics of the Neural NER model.

While the global F1-Score of 79.6% indicates a robust baseline, a detailed analysis of class-specific performance reveals a significant division between semantic and rigid entities.

| Entity Type          | Precision | Recall | F1-Score     | Support | Performance Note                               |
|----------------------|-----------|--------|--------------|---------|------------------------------------------------|
| <b>APT</b> (Actor)   | 0.896     | 0.934  | <b>0.915</b> | 2381    | <i>Excellent Semantic Understanding</i>        |
| <b>SECTEAM</b>       | 0.943     | 0.948  | <b>0.945</b> | 574     | <i>Excellent Context Recognition</i>           |
| <b>SHA2</b> (Hash)   | 0.945     | 0.964  | <b>0.954</b> | 990     | <i>High Reliability</i>                        |
| <b>IP</b>            | 1.000     | 1.000  | <b>1.000</b> | 133     | <i>Perfect (Limited Test Set)</i>              |
| <b>URL</b>           | 1.000     | 0.867  | 0.929        | 98      | <i>High Precision, some misses</i>             |
| <b>MAL</b> (Malware) | 0.655     | 0.648  | 0.652        | 1761    | <i>Moderate (Ambiguity with Tools)</i>         |
| <b>TOOL</b>          | 0.537     | 0.526  | 0.531        | 1559    | <i>Low (Confused with legitimate software)</i> |
| <b>SHA1</b> (Hash)   | 0.571     | 0.554  | <b>0.563</b> | 101     | <i>Critical Failure (Tokenization issues)</i>  |

Table 5.2: Detailed classification report by entity type.

Ablation Study:

The results provide the validation for the adoption of the **Hybrid Architecture** (Neural + Regex) proposed in Chapter 4. The data highlights two critical behaviors:

The model excels at identifying context-dependent entities: the F1-Score of 91.5% for APTs demonstrates that the XLM-RoBERTa architecture successfully captures the semantic shades required to identify Threat Actors, a task that Regular Expressions cannot perform due to the lack of fixed patterns.

Conversely, the model performs worse on rigid identifiers. In particular, the SHA1 class reaches an F1-score of only 56.3%. A plausible reason is the subword tokenization step: long alphanumeric hashes are often broken into fragments with little semantic structure, which makes it harder to recover the correct entity span.

At the end we can state that relying only on the Neural model would result in missing approximately 44% of SHA1 indicators (Recall 0.554), which is unacceptable for a CTI



platform. The integration of the deterministic Regex engine theoretically elevates the recall for structured classes (SHA1, MD5, IP, URL) to near 100%. The neural model is thus reserved for the tasks where it outperforms rule-based logic (APTs, Malware families), while the Regex layer acts as a safety net for rigid identifiers, correcting the specific weaknesses evidenced by this evaluation.

### 5.2.2. Threat Classification Model Selection

The design of the Relevance Gate was driven by a rigorous empirical validation. We aimed to answer two fundamental questions:

1. **Data Strategy:** Is it better to learn from massive, noisy datasets (Big Data) or small, expertly curated ones (Small Data)?
2. **Model Architecture:** Does the architectural complexity (Stacking Ensemble) yield statistically significant improvements over simpler baselines (SVC) or pure Transformers (DeBERTa)?

## Experimental Setup

To avoid bias, we evaluated all three architectures (Baseline, Transformer, Ensemble) under three distinct data scenarios. We chose the PR-AUC (Area Under the Precision-Recall Curve) metric, as it is commonly used for imbalanced datasets where the positive class (Relevant CTI) is rare.

- **Exp A (Large Weakly Labeled):** Training on 40k weakly labeled samples, Testing on 600 Gold Standard.
- **Exp B (Reduced Weakly Labeled):** Training on 6k weakly labeled samples, Testing on 600 Gold Standard.
- **Exp C (Small Manually Annotated):** Training on 600 Gold Standard items (using Stratified 5-Fold Cross Validation for internal validation), Testing on the 40k dataset (to test generalization on the wild distribution).

## Results: The "No Free Lunch" Validation

The comparative results are summarized in Table 5.3. The validation empirically demonstrates that increasing data quantity without ensuring label quality (Exp A and B) yields suboptimal performance compared to a "Small Data" strategy focused on high-quality, manual annotations (Exp C).

| Scenario                       | Model Configuration               | PR-AUC      | Recall      | F1-Score    |
|--------------------------------|-----------------------------------|-------------|-------------|-------------|
| <b>Exp A: Big Data (40k)</b>   | Ensemble (Test: 600)              | 0.66        | 0.57        | 0.55        |
| <b>Exp B: Subset (6k)</b>      | Ensemble (Test: 600)              | 0.58        | 0.51        | 0.37        |
| <b>Exp C: Small Data (600)</b> | <b>Stacking Ensemble (Winner)</b> | <b>0.78</b> | <b>0.87</b> | <b>0.77</b> |

**Table 5.3:** Comprehensive Model Selection Matrix. The table highlights that the model trained on the small, high-quality dataset (Exp C) consistently outperforms models trained on larger, noisier datasets (Exp A, Exp B) across all key metrics.

**Analysis of Results:** The results favor the "Small Data" approach. Exp C (600 labels) achieved a PR-AUC of 0.78, outperforming the "Big Data" approach in Exp A (40k noisy labels), which saturated at 0.66. This confirms that in complex CTI domains, model performance is bounded by label noise rather than dataset size.

Exp B (6k subset) performed worse than both other scenarios (PR-AUC 0.58, F1 0.37). This suggests that reducing the dataset size without manual cleaning merely reduces the statistical signal while retaining the noise, leading to the poorest generalization.

In the end, as shown in Figure 5.2, we can state that the Stacking Ensemble in Exp C proved incredible results. While the Transformer alone was less stable on the small dataset (PR-AUC 0.621), the Stacking Ensemble combined the baseline signal effectively, raising PR-AUC to 0.774 and reaching a Recall of 87%, which is important for minimizing false negatives in security operations.

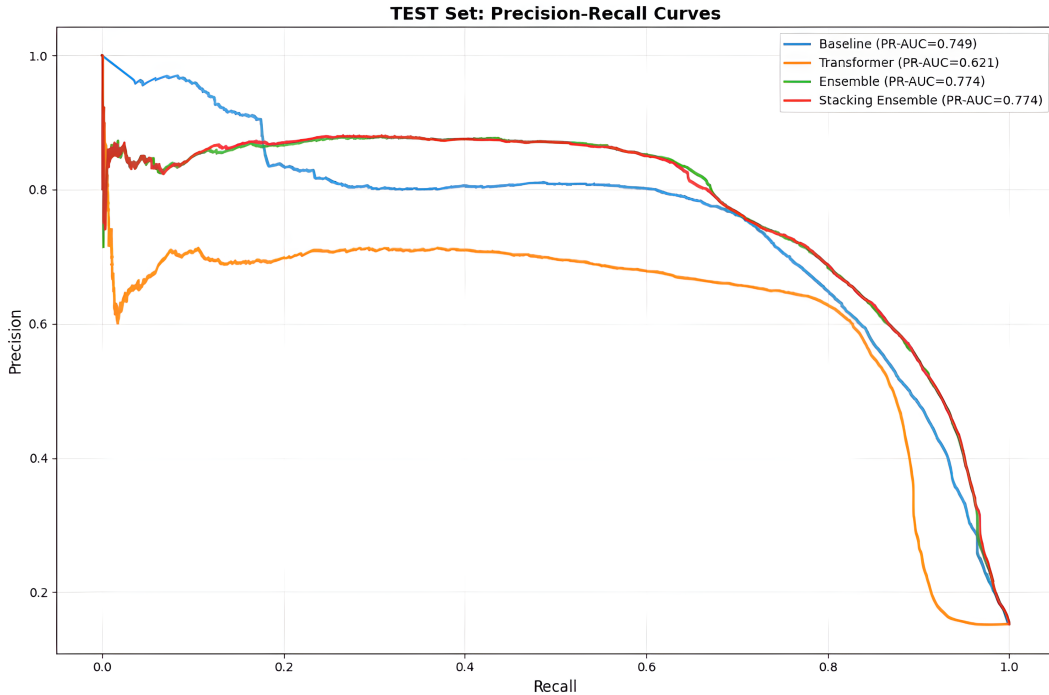


Figure 5.2: Precision-Recall curves for Experiment C (Test Set). The Stacking Ensemble (Red line) compensates for the instability of the single Transformer (Orange) on small data, achieving the highest area under the curve (0.774) and outperforming the Baseline (Blue).

This rigorous ablation study confirms that the Stacking Ensemble trained on the smaller, hand-labeled dataset, is the optimal configuration, strictly adhering to the "No Free Lunch" principle [74] by verifying performance through experimentation rather than assuming that larger datasets yield better results.

### 5.3. Comparative Source Analysis

Having validated the reliability of the individual extraction components, this section presents the core comparative analysis of the intelligence ecosystem. The objective is to quantify the specific contribution of each source category, ranging from unstructured Dark Web forums to structured OSINT feeds, within the integrated threat landscape.

To ensure a rigorous comparison between heterogeneous data formats, the analysis adopts a normalized evaluation framework. A direct comparison of raw volumes would inherently bias the results towards high-frequency, low-quality channels (e.g., social media streams). Therefore, the evaluation metrics prioritize **Information Density**, **Semantic Exclusivity**, and **Temporal Celerity**. By normalizing the extracted entities against document

length and enforcing strict canonicalization for threat actors, the platform isolates the true strategic value of each stream. The following subsections detail the experimental findings across five dimensions, moving from quantitative volume assessments to complex relational attribution, and concluding with a consolidated discussion answering the Research Questions defined in Chapter 2.

The empirical evaluation is structured into five different evaluated phases:

The **Volume and Density Analysis (Sec. 5.3.1)** evaluates the quantity of information provided. We compare sources based on *Total Entity Volume* and *Information Density* (Entities per 1K Words) to distinguish between sources that provide high-volume noise and those offering concentrated intelligence.

Then we evaluate **Source Reliability: Exclusivity and Overlap [40, 57] (Sec. 5.3.2, 5.3.3)** analyzing the uniqueness of the data. We measure *Exclusivity* (entities found only in a specific source type) and *Source Overlap*. Crucially, overlap is analyzed at two levels: "Raw Overlap" on atomic indicators and "Semantic Overlap" on APT-TTP relationships, to determine if sources corroborate each other on technical details or strategic attribution.

Then we proceed with the **Timeliness and Celerity [51, 56] (Sec. 5.3.4)** evaluation, which quantifies the "Lead Time" and detection speed. By determining the Global First Seen timestamp ( $t_{min}$ ) for every normalized entity across the entire dataset, we calculate the First-to-Report Ranking. This metric aggregates the frequency with which each source category captured the threat artifact, establishing an empirical hierarchy of anticipation capabilities independent of official disclosure delays.

After that, we study **Strategic Coverage and Attribution (Sec. 5.3.5, 5.3.6)**, analyzing the scope of intelligence. We classify extracted data to determine which sources specialize in technical aspects (Blind Spot Analysis) versus strategic attribution, measuring the number of valid *APT-TTP Links* established by each source type.

In the end we draw up the **Longitudinal Analysis and Lifecycle Case Study (Sec. 5.3.7)** that is a comprehensive analysis of all extracted notions about the analyzed sources during the monitored period (2023-2025) and qualitative end-to-end analysis of a specific campaign, tracing its lifecycle from the first signal on the Dark Web to its analysis in vendor reports, demonstrating the platform's correlation capabilities in a real-world scenario.

Before diving into the analysis results, we will address a critical challenge in CTI analysis, which is the ambiguity of terms. For the purpose of this study, we adopt the following definitions:

- **Entity:** Any of the extracted artifact, belonging to one of the 21 different labels defined by the dataset in Section 5.2.1; the most relevant ones are the categories of *Indicator of Compromise* (IP, URL, Hash), *Threat Actor* (APT name), *CVE* (Vulnerabilities), *Malware Family*, or *Attack Technique* (MITRE ID).
- **Canonicalization (Alias Handling):** To calculate exclusivity and overlap accurately, entities are normalized to a canonical form. Specifically for Threat Actors, where naming conventions vary significantly (e.g., "APT28" vs. "Fancy Bear"), we apply a soft-matching algorithm based on the MITRE ATT&CK alias dictionary to group synonyms under a single unique identifier.
- **Metric Normalization:** To avoid bias towards verbose sources, volume metrics are calculated not only in absolute terms but also as Density, defined as the number of unique entities extracted per 1,000 words of analyzed text.

### 5.3.1. Volume and Density Analysis

#### 2025 Analysis

The 2025 window represents the most mature phase of the data collection campaign. The quantitative analysis of this period highlights a distinct structural divergence between automated feeds and human-generated content.

**A. Total Entity Volume:** As illustrated in Figure 5.3, the volume distribution is heavily biased towards the **Intelligence** category, which alone contributes with **881,593 entities**. This magnitude is intrinsic to the nature of OSINT feeds (e.g., URLhaus, ThreatFox), which operate as high-frequency streams of atomic indicators (IPs, Hashes) extracted deterministically.

Conversely, the **DLS** (Data Leak Sites) category appears as a minority in absolute terms ( $\approx 480$  items). This scarcity is attributable to a confluence of strategic and technical factors. Strategically, as noted by Paladini et al. (2024) [51], threat actors are increasingly abandoning the instability of Tor-based infrastructure in favor of more resilient platforms like Telegram. Technically, the low volume is also impacted by a specific Ingestion Bias. The analysis framework strictly filters items based on their **reference\_date** to ensure temporal consistency. However, a significant portion of unstructured DLS pages lacks explicit, machine-parsable timestamps. Consequently, valid leak posts without a clear publication date are systematically excluded from the time-windowed analysis, leading to an under-representation of the actual DLS activity compared to sources with structured metadata.

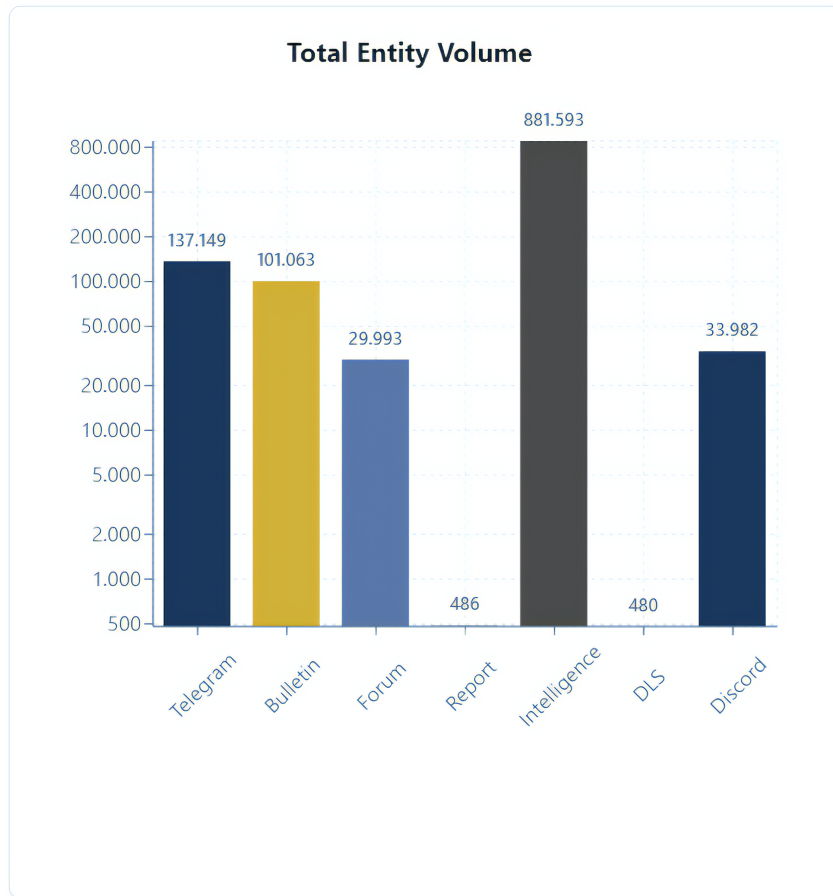


Figure 5.3: Total Entity Volume (2025). The Intelligence feeds confirms the high-volume, low-context nature of OSINT streams.

**B. Entities per Item:** Figure 5.4 reveals the "richness" of single documents. Here, the hierarchy flips, revealing a significant functional homology between Telegram channels and Security Bulletins:

- **Telegram (26.96) and Bulletins (24.61):** These two categories exhibit nearly identical richness metrics. This similarity suggests that the monitored Telegram channels (often automated “infosec bots”) effectively act as “real-time bulletins”. Similar to official vendor bulletins (e.g., Broadcom, Microsoft) that list many vulnerable product versions and CVEs within a single advisory, these Telegram channels disseminate consolidated lists of targets or indicators. The NER pipeline captures both as high-density clusters of technical entities.
- **Forum (11.60):** While forums contain technical data, the extraction is naturally diluted by the conversational structure of the threads (greetings, signatures, non-technical replies), resulting in a lower average count per item.

- **DLS (3.00):** This exact integer value empirically validates the Deterministic Pipeline constraint described in the methodology. The ingestion logic for Ransomware sites forces the extraction of exactly three core entities per post: the *Threat Actor* (Group Name), and the *Network Indicators* (Tor URL/DOM). There is no probabilistic variance here; the pipeline strictly maps the structured fields of the leak site.

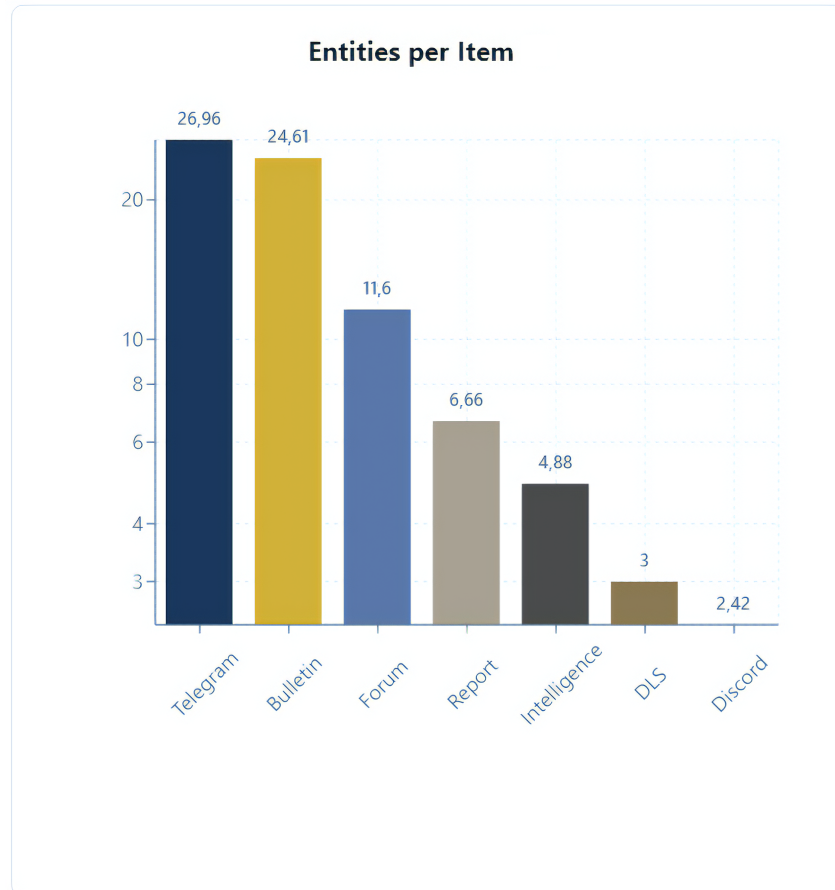


Figure 5.4: Average Entities per Item (2025). The convergence between Telegram and Bulletins, both acting as high-density dissemination vectors.

**C. Information Density - Entities per 1K Words) [9]:** To normalize against verbosity, we analyze the density metric (Figure 5.5), which serves as a proxy for the "Signal-to-Noise Ratio".

- **Intelligence (502.65):** Here the theoretical maximum is achieved. Since feed items often contain *only* the indicator (e.g., an IP address) with zero narrative text, the ratio of entities to words approaches 1:2, resulting in the end into an extreme density.
- **Reports (27.56):** Despite containing high-value intelligence, PDF reports are

highly verbose. The ratio indicates that for every valid IoC extracted, there are approximately 35 words of narrative context (executive summaries, methodology descriptions). While valuable for strategic understanding, they represent a low-density source for automated ingestion compared to the raw signal of Intelligence feeds.

- **Forums (17.92):** Confirmed as the lowest density source. The probabilistic NER model, correctly discards the majority of the conversational noise, extracting entities only from the rare technical segments embedded in the discussions.

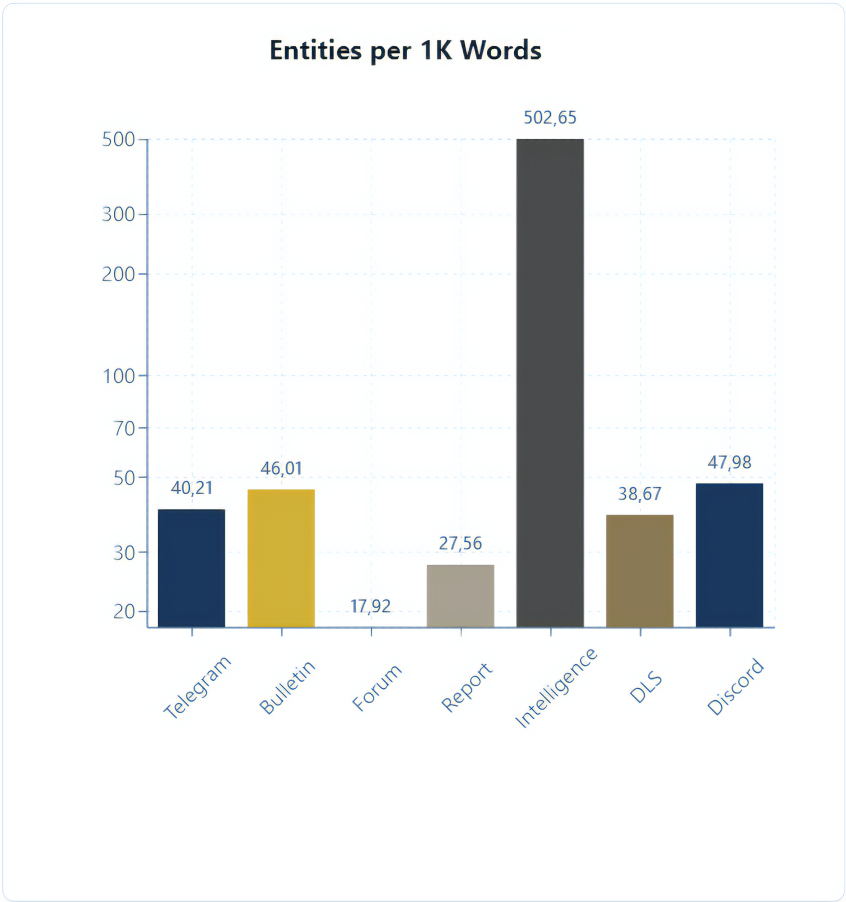


Figure 5.5: Information Density in 2025. Intelligence feeds provide pure signal, while Forums and Reports are diluted by narrative text.

2024 Analysis

The quantitative analysis of the 2024 window provides a baseline for understanding the evolutionary trends of the threat landscape. Unlike the fragmented ecosystem observed in 2025, the 2024 data highlights a period of peak activity for traditional Tor-based infrastructures before their subsequent destabilization.



**A. Total Entity Volume:** Figure 5.6 illustrates the volume distribution. While the Intelligence category remains the dominant source (225,050 entities) due to the continuous stream of atomic indicators, the most significant insight emerges from the **DLS** (Data Leak Sites) category. In 2024, the platform ingested 3,567 entities from Ransomware Leak Sites, higher than the volume recorded in 2025 ( $\approx 480$ ). This contrast empirically validates the "Short Lifecycle Hypothesis" regarding underground infrastructure. The 2024 data captures a phase where major Ransomware-as-a-Service (RaaS) groups maintained stable, accessible .onion blogs. The subsequent collapse of this volume in 2025 is not attributable to a decrease in attacks, but to the high volatility of these hidden services, which were subjected to law enforcement takedowns, extensive bans, and infrastructure decay, forcing the strategic migration towards Telegram discussed in the previous section.

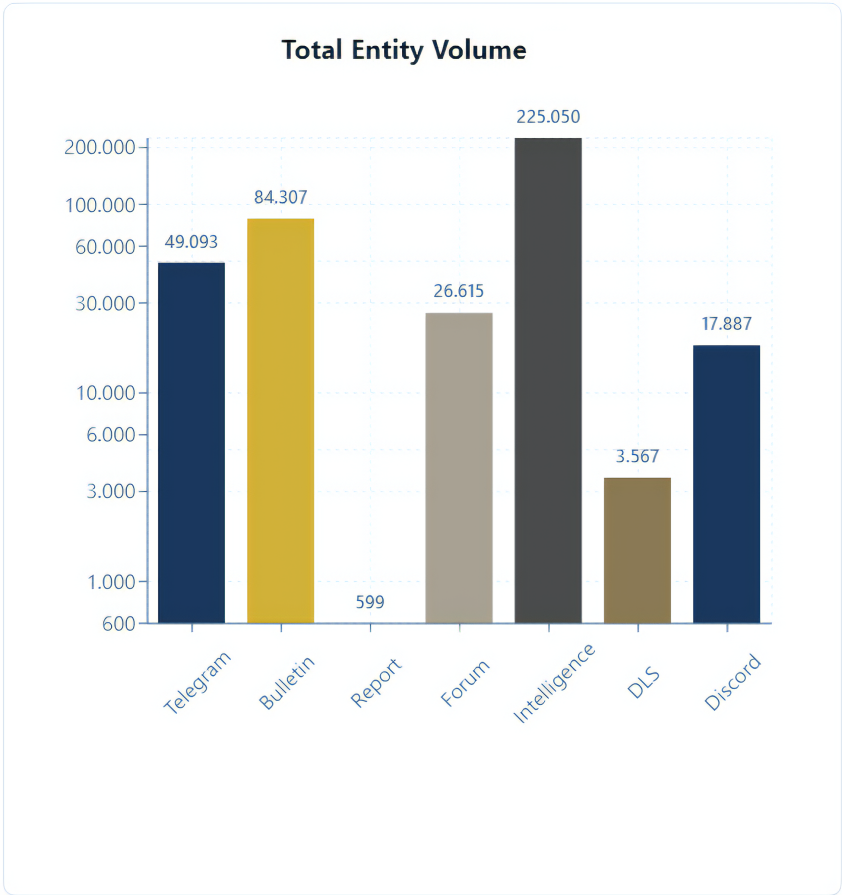


Figure 5.6: Total Entity Volume (2024). The significant volume of DLS (3,567), reflecting a period of stability for Tor-based leak sites before the 2025 decline.

**B. Entities per Item:** The per-item analysis (Figure 5.7) validates the structural consistency of the ingestion pipelines, specifically regarding the deterministic constraints:

- **Telegram (23.16) and Bulletin (20.10):** The functional homology between these two sources is evident in 2024 as well. Both operate as high-density dissemination vectors, where a single item acts as a container for lists of indicators or vulnerabilities.
- **DLS (3.00):** The metric remains fixed at exactly 3.00, confirming the rigidity of the Deterministic Pipeline for DLS. Regardless of the volume fluctuations (high in 2024, low in 2025), the extraction logic remains invariant.
- **Discord (1.77):** This source shows the lowest richness. Unlike the bot-driven nature of the monitored Telegram channels, the Discord data predominantly consists of chat logs and short user interactions, resulting in a sparse distribution of entities per message.

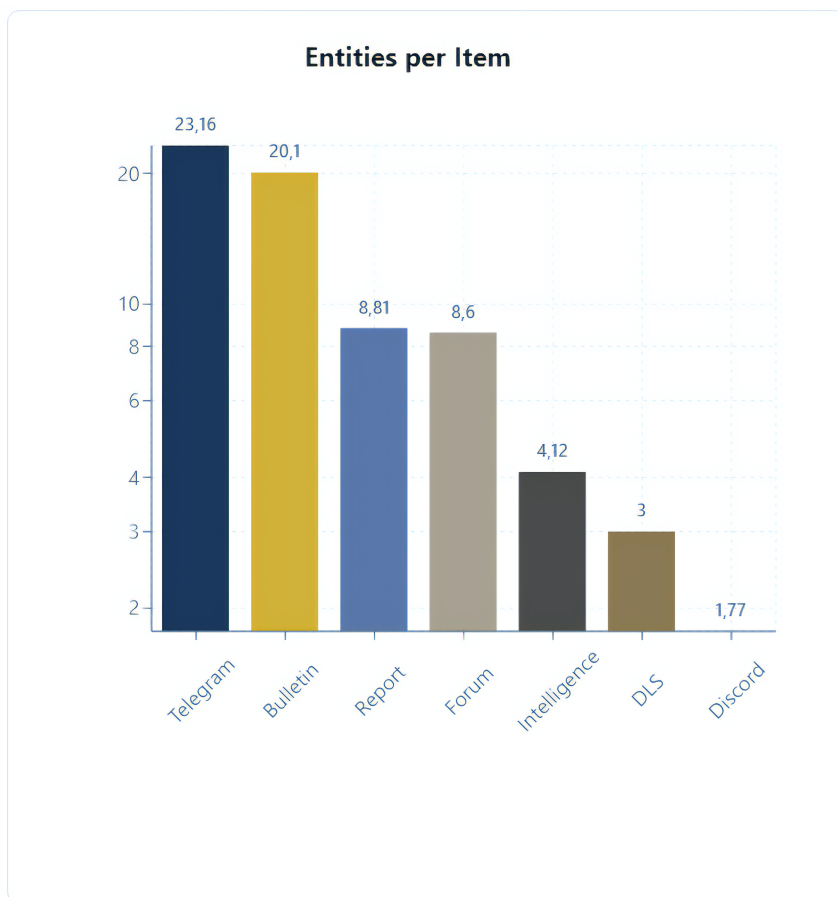


Figure 5.7: Average Entities per Item (2024). The DLS metric remains fixed at 3.0, confirming the deterministic nature of the extraction pipeline regardless of total volume.

**C. Information Density - Entities per 1K Words):** The density analysis for 2024 (Figure 5.8) highlights the "Signal-to-Noise" characteristics of strategic reporting versus

technical feeds.

- **Intelligence (469.26):** Providing raw technical signal with minimal linguistic overhead, it confirms as the highest density source.
- **Reports (4.29):** This metric is much lower compared to 2025. This anomaly suggests that the reports ingested in 2024 were predominantly Strategic or Executive Summaries (e.g., "Year in Review" documents). These documents are characterized by extensive narrative text, policy discussions, and trend analysis, with relatively few technical indicators (IoCs) for each page. While their strategic value is high, their "density" for automated extraction is minimal, requiring the ingestion of thousands of words to retrieve few valid entities.
- **Bulletin (48.95):** Bulletins maintain high density, confirming their role as technical advisories (concise text, high volume of CVEs).

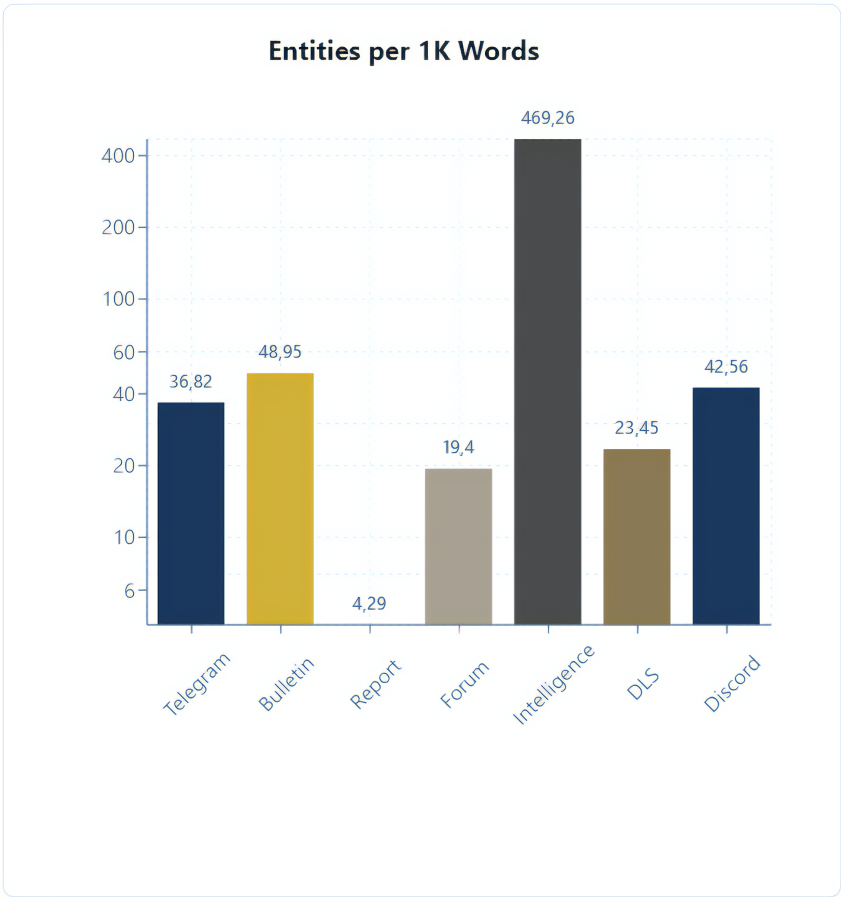


Figure 5.8: Information Density in 2024. The low density of Reports (4.29), indicating a prevalence of narrative-heavy strategic documents.

## 2023 Analysis

The analysis of the 2023 window provides the historical baseline for the study. This dataset serves as a control group to evaluate the persistence and recoverability of intelligence artifacts over time. The divergence in volume between stable institutional sources and volatile underground infrastructures reaches its maximum here.

**A. Total Entity Volume:** Figure 5.9 reveals a volume distribution diametrically opposed to the real-time streams of 2025.

- **Bulletin (77,364):** The high volume confirms the Historical Persistence of official sources. Security advisories published in 2023 remain indexed, accessible, and immutable on the Surface Web, allowing for near-complete historical reconstruction.
- **DLS (60):** This extremely low value (compared to 3,567 in 2024) illustrates the Survivorship Bias inherent in Dark Web forensics. Unlike Bulletins, Ransomware Leak Sites have a short "Time-to-Live". Most .onion sites active in 2023 are currently defunct due to server seizure or abandonment. Consequently, retrospective ingestion fails to recover these artifacts, capturing only the negligible fraction of posts that persisted on rare, long-living mirrors.
- **Intelligence (15,302):** The volume is significantly lower than in 2025 (881k). This reflects the Ephemeral Nature of IoCs. Structured feeds (e.g., URLhaus) are designed to track "active" threats; indicators from 2023 are typically rotated out of production feeds as they become obsolete (cleaned or sinkholed), leading to a natural decay in historical availability.

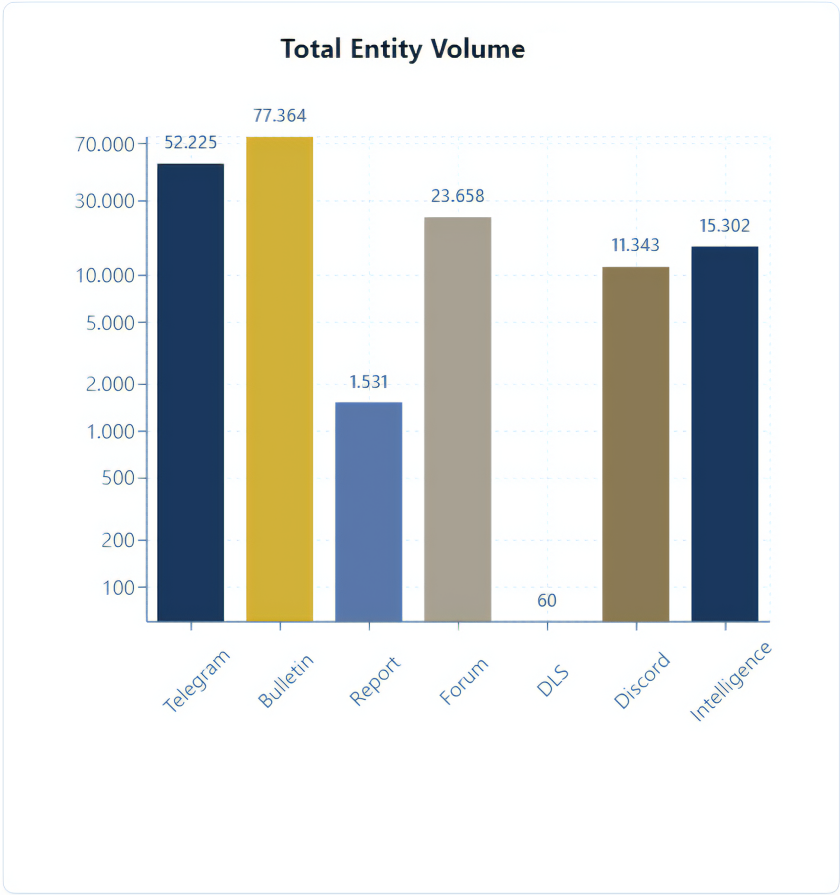


Figure 5.9: Total Entity Volume (2023). Bulletins dominate due to high persistence, while DLS volumes (60) are negligible due to the decay of .onion infrastructure.

**B. Entities per Item:** Figure 5.10 confirms the stability of the sources, from a structural perspective, across the entire longitudinal study (2023-2025).

- **Telegram (23.97) and Bulletin (21.06):** The functional homology is preserved. Even in historical data, Telegram channels and Official Bulletins maintain their role as high-density lists.

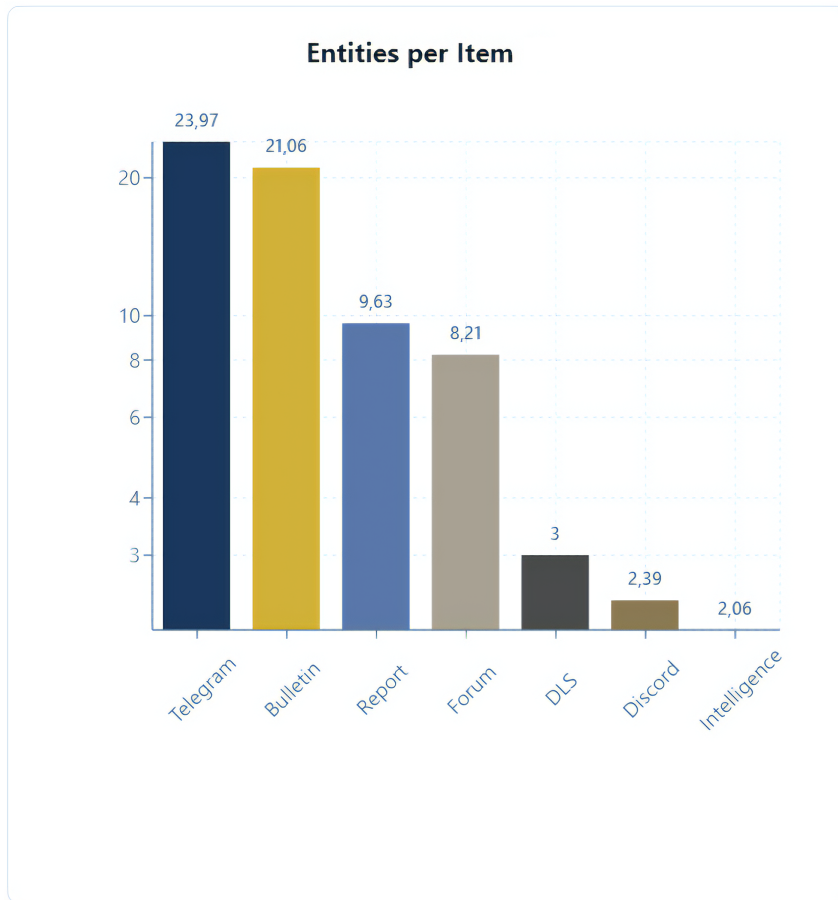


Figure 5.10: Average Entities per Item (2023). The consistency of the metrics (DLS at 3.00) validates the stability of the extraction pipelines across the years.

**C. Information Density - Entities per 1K Words):** The density analysis (Figure 5.11) for 2023 highlights an incredible deviation for the Intelligence category.

- **Intelligence (194.24):** While still high, the density is less than half of the 2025 peak (502.65). This suggests that the few historical feed items that persist in archives are likely enriched with more metadata or descriptions compared to the raw "IP-only" streams ingested in real-time, slightly diluting their density.
- **Report (6.09):** Confirmed as the lowest density source (consistent with 2024). Historical reports ingested from 2023 act most as narrative archives rather than sources of actionable technical indicators.

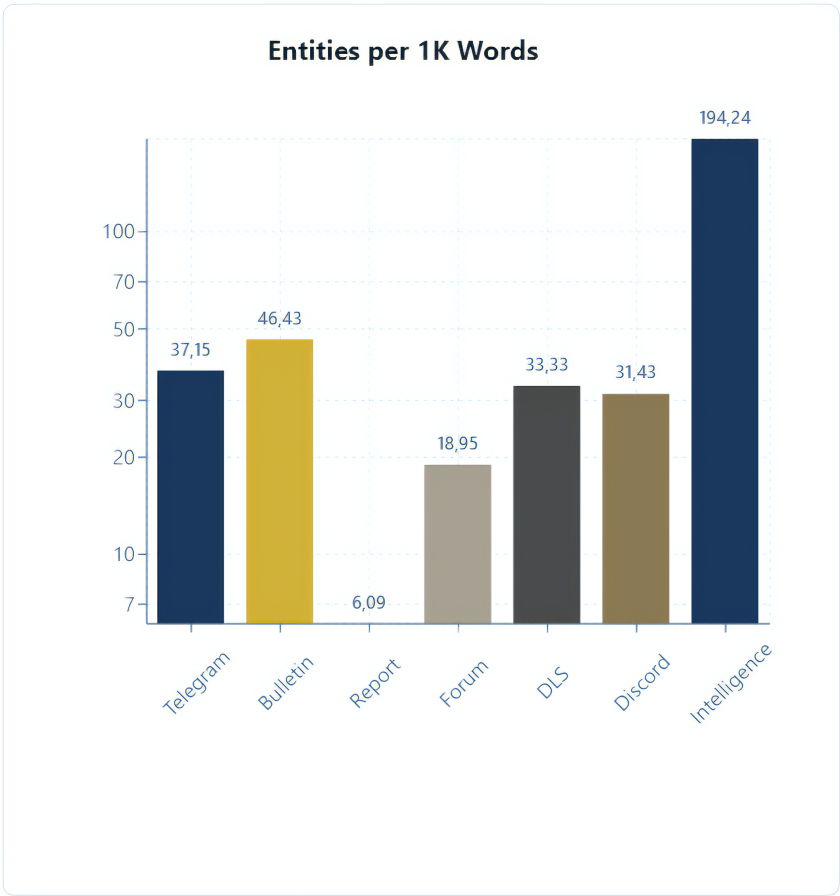


Figure 5.11: Information Density in 2023. Reports show minimal density (6.09), confirming their role as narrative and strategic documents rather than technical feeds.

5.3.2. Source Exclusivity and Uniqueness

2025 Analysis

The analysis of **Source Exclusivity** measures the number of entities that appear *uniquely* in a specific source category. The 2025 data (Figure 5.12) reveals a sharp dichotomy between sources that generate raw data and sources that contextualize existing information.

**A. The Atomic Exclusivity of Intelligence Feeds:** The Intelligence category dominates the metric with 515,248 exclusive entities. However, this magnitude must be interpreted through the lens of the *Atomic Nature* of these indicators. These feeds ingest massive quantities of ephemeral hashes (MD5/SHA256) and dynamic IP addresses. By definition, a specific file hash generated by a polymorphic malware variant is mathematically unique and unlikely to appear in a high-level PDF report or a forum discussion. Therefore, this high exclusivity represents "Technical Novelty" (new artifacts) rather than

"Strategic Novelty" (new threat actors or campaigns).

**B. The Primary Source Battle:** The most significant strategic insight emerges from the comparison between human-generated sources. Telegram (29,825) surpasses Bulletins (26,175) in terms of unique contributions. This inversion of the traditional hierarchy suggests that Telegram has consolidated its role as a Primary Source. It is no longer merely an "echo chamber" for discussions found elsewhere; it is the environment where new victim names, fresh credentials, and zero-day proofs-of-concept are published *first*. If these entities were merely reposts of official Bulletins, the overlap would be high and the exclusivity low. The high exclusivity score confirms that Telegram generates independent, unique intelligence that is not yet present in official institutional advisories.

**C. The "Derivative Nature" of Reports:** The most counter-intuitive result is the negligible exclusivity of Reports (126), the lowest among all categories. This finding empirically characterizes Reports as Derivative and Contextual Sources. A detailed threat analysis report typically analyzes a campaign that has *already* occurred. Consequently, the entities it cites (Malware names, victim sectors, C2 IPs) have likely already been ingested by the high-speed Intelligence feeds or discussed in Bulletins days before. The value of a Report, therefore, lies not in the *novelty* of its atomic entities, but in the *relational context* it provides connecting them. The pipeline correctly identifies that while the Report adds knowledge, it rarely adds unique, never-before-seen entity strings to the global database.



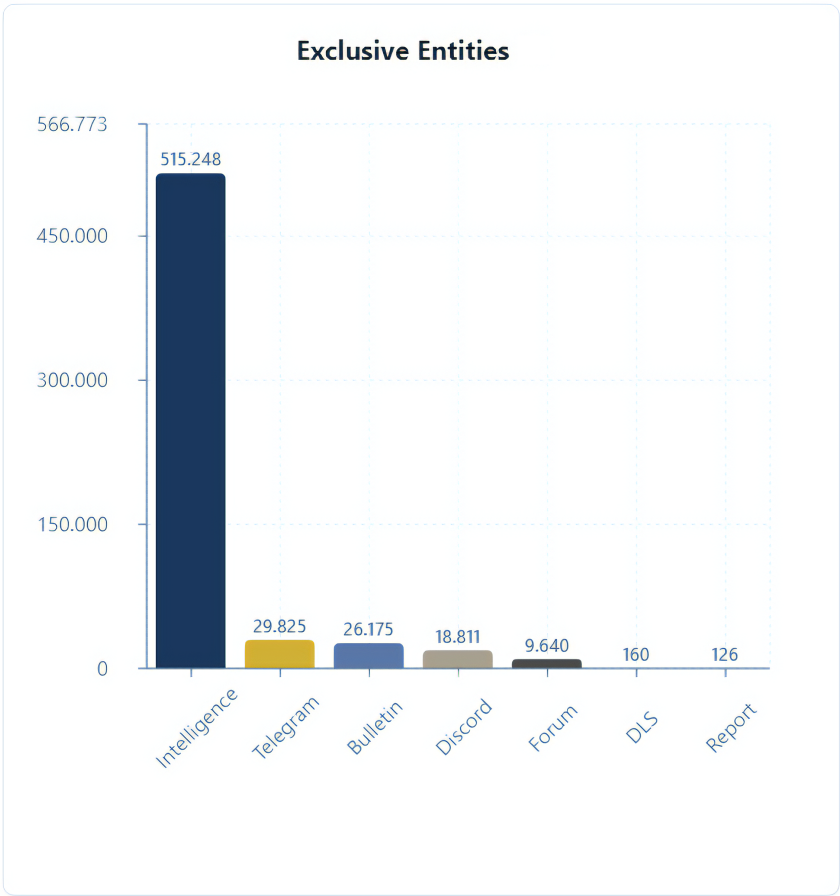


Figure 5.12: Exclusive Entities (2025). Telegram (29,825) outperforms Bulletins, proving its role as a generator of unique intelligence. Reports show minimal exclusivity (126), confirming their role as contextualizers of known threats.

2024 Analysis

We can use the 2024 dataset as a benchmark for the "Traditional" threat landscape structure, before the strategic shifts observed in the subsequent year. The exclusivity metrics (Figure 5.13) highlight a distinct hierarchy where institutional sources still maintained a lead over emerging underground channels.

**A. The Traditional Hierarchy - Bulletins Lead Telegram:** In strong contrast to 2025, the 2024 data shows Bulletins (25,978) significantly outperforming Telegram (11,851) in terms of unique entities. This comparison offers a dynamic view of the threat landscape’s evolution. While the "Novelty" of official Bulletins remained numerically stable across the two years ( $\approx 26k$  in both 2024 and 2025), Telegram’s exclusivity tripled in the following year (from 11k to 29k). This trend empirically proves that 2024 was the transition period where Telegram began evolving from a secondary "re-post" channel

into a primary source of original intelligence. In 2024, however, the "First-to-Disclose" capability for vulnerabilities and patch data remained firmly with the official vendors.

**B. The Strategic Value of DLS:** The DLS category records 1,199 exclusive entities. While this number is absolutely lower than the technical feeds, its relative weight is substantial. Comparing this with the Total Volume measured in Section 5.3.1 (3,567 total entities), we calculate a Uniqueness Rate of  $\approx 33\%$ . This implies that one out of every three entities extracted from a Ransomware Leak Site is *never* observed in any other source category, neither in OSINT feeds, nor in Reports, nor in Forums. This metric scientifically validates the necessity of monitoring Tor-based Leak Sites: despite their technical instability, they cover a specific "Blind Spot" (typically small-business domains) that remains completely invisible to the rest of the ecosystem.

**C. Structural Stability of Derivative Sources:** Report category (195 exclusive entities) and Intelligence feeds (120,468) confirm the structural constants of the pipeline. Intelligence remains the primary engine for unique technical artifacts (IPs/Hashes), although the volume is lower than in 2025, likely due to the natural decay of historical indicators. Reports, instead, consistently exhibit the lowest exclusivity. This reinforces the conclusion that the function of a Report is not *Discovery* (finding new entities) but *Correlation* (explaining known entities). The limited variation between 2024 (195) and 2025 (126) suggests this is a stable property of the format, rather than an effect of the yearly threat landscape.

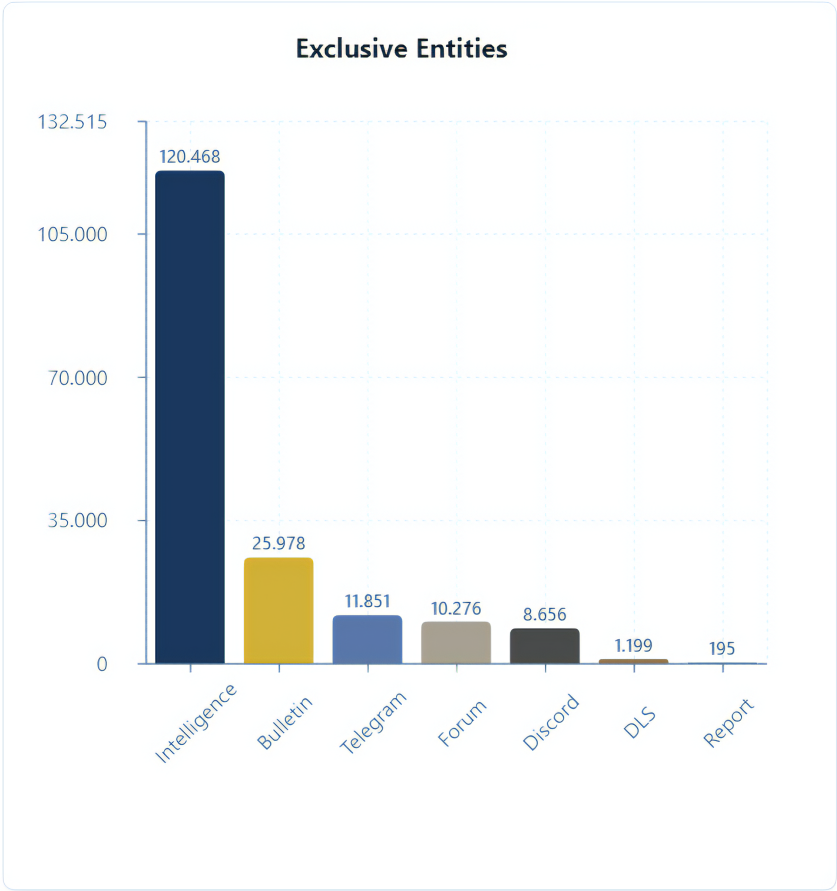


Figure 5.13: Exclusive Entities (2024). The traditional hierarchy is visible, with Bulletins (25,978) leading Telegram. DLS shows a significant contribution of 1,200 unique domains not found elsewhere.

2023 Analysis

The 2023 dataset acts as the historical control group, revealing how the "Uniqueness" of intelligence sources degrades or persists over time. The distribution in Figure 5.14 presents a radically different topology compared to the modern landscape (2025), characterized by the absolute dominance of institutional sources over ephemeral feeds.

**A. The "Archival Supremacy" of Bulletins:** The most striking finding is the primacy of **Bulletins**, which record 26,981 exclusive entities. This metric is remarkably consistent with the values observed in 2024 (25k) and 2025 (26k). This stability empirically proves the **Data Permanence of institutional advisories**. Unlike criminal infrastructure or structured feeds that decay, Vendor advisories remain indexed and accessible indefinitely. Consequently, when performing retrospective analysis on historical threats, Bulletins become the single most reliable source of unique intelligence, effectively

"outliving" the sources that might have originally detected the threat (like DLS).

**B. Telegram - The Secondary Channel:** In 2023, Telegram contributed 13,302 exclusive entities. While significant, this is less than half the volume observed in 2025 (29,825). This structural gap validates the evolutionary timeline of the threat landscape: in 2023, Telegram was still a **Secondary Channel**, likely used more for reposting content found elsewhere rather than as the "Ground Zero" for leaks. The shift to Telegram as a primary, exclusive source is a phenomenon that accelerates specifically in the 2024-2025 window.

**C. The "Digital Decay" of Technical Feeds and DLS:** The analysis of Intelligence Feeds (7,428) and DLS (26) provides a quantifiable measure of Information Decay.

In 2025, feeds provided 515k unique items; in 2023, only 7k remain unique. This indicates that 98% of technical indicators (IPs/Hashes) are ephemeral: they are either rotated out of feeds, sinkholed, or become non-unique (shared across all sources) over time. With only 26 exclusive items, the Ransomware landscape of 2023 (DLS) is effectively lost to historical reconstruction. This confirms that the unique value of Dark Web monitoring is strictly real-time. If the data is not captured at the moment of publication (before the .onion site goes offline), the exclusivity drops to near zero, leaving analysts with only the sanitized summaries found in Reports or Bulletins.

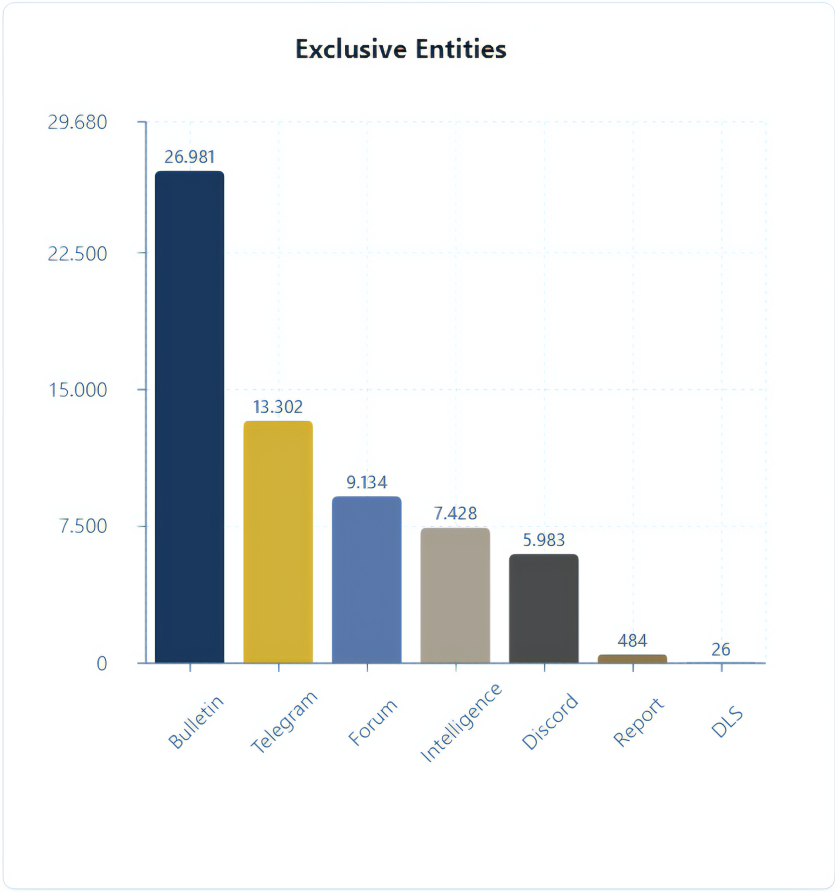


Figure 5.14: Exclusive Entities (2023). Bulletins (26,981) dominate the historical landscape, proving their archival stability. The low value for Intelligence (7,428) and DLS (26) illustrates the rapid decay of ephemeral technical data.

5.3.3. Cross-Source Overlap and Redundancy

2025 Analysis

The analysis of Cross-Source Overlap quantifies the degree of redundancy and confirmation between different intelligence streams. We utilize the Jaccard Similarity Coefficient, a set-based similarity measure defined as the ratio between the intersection and the union of two sets of entities:  $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$ , where  $J \in [0, 1]$  (0 = no overlap, 1 = identical sets) [30].

**A. The "Central Axis" - Bulletin-Telegram Convergence:** The most critical finding of the 2025 dataset is the strong structural coupling between Bulletins and Telegram. As shown in the Source Overlap table (Figure 5.15), this pair achieves the highest Jaccard Index of the entire ecosystem (16.87%), with a massive intersection of 11,064 shared entities. This metric empirically confirms the "Functional Homology" hypothesis introduced

in Section 5.3.1. The high overlap indicates that Official Vendor advisories (Bulletins) are immediately reposted and dissected by Telegram "Infosec Bots". Conversely, leaks born on Telegram (e.g., a zero-day POC) are rapidly acknowledged and assigned CVEs in official Bulletins.

In the Heatmap (Figure 5.16), the node connecting Telegram and Bulletin is the largest and most intense, representing the gravitational center of the 2025 threat landscape.

**Source Overlap**

| SOURCE A     | SOURCE B     | ENTITIES | UNIQUE A | UNIQUE B | JACCARD |
|--------------|--------------|----------|----------|----------|---------|
| Bulletin     | Telegram     | 11.064   | 36.424   | 40.232   | 16.87%  |
| Discord      | Telegram     | 3.282    | 22.484   | 40.232   | 5.52%   |
| Bulletin     | Discord      | 3.019    | 36.424   | 22.484   | 5.40%   |
| Discord      | Forum        | 1.497    | 22.484   | 11.604   | 4.59%   |
| Forum        | Telegram     | 2.119    | 11.604   | 40.232   | 4.26%   |
| Bulletin     | Forum        | 1.959    | 36.424   | 11.604   | 4.25%   |
| Forum        | Report       | 91       | 11.604   | 343      | 0.77%   |
| Discord      | Report       | 138      | 22.484   | 343      | 0.61%   |
| Bulletin     | Report       | 213      | 36.424   | 343      | 0.58%   |
| Report       | Telegram     | 204      | 343      | 40.232   | 0.51%   |
| Bulletin     | Intelligence | 2.081    | 36.424   | 517.051  | 0.38%   |
| Intelligence | Telegram     | 2.049    | 517.051  | 40.232   | 0.37%   |
| Discord      | Intelligence | 796      | 22.484   | 517.051  | 0.15%   |
| Forum        | Intelligence | 543      | 11.604   | 517.051  | 0.10%   |
| DLS          | Discord      | 7        | 170      | 22.484   | 0.03%   |
| Intelligence | Report       | 86       | 517.051  | 343      | 0.02%   |
| DLS          | Intelligence | 3        | 170      | 517.051  | 0.00%   |
| DLS          | Telegram     | 1        | 170      | 40.232   | 0.00%   |

Figure 5.15: Source Overlap Table (2025). The Bulletin-Telegram 16.87% Jaccard Similarity, we can see massive synchronization between official advisories and underground channels.

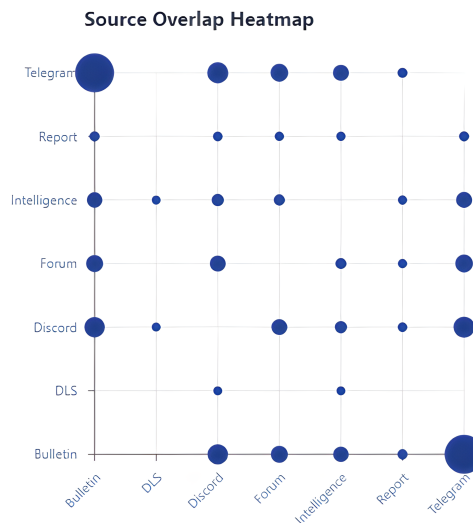


Figure 5.16: Overlap Heatmap (2025). The central axis is formed by Telegram and Bulletins, with Discord acting as a satellite community.

**B. APT-TTP Links:** We analyzed the Semantic Overlap, beyond simple entity matching (e.g., sharing the same IP): do these sources agree on *who* is attacking and *how*? Figure 5.17 specifically focuses on "APT-TTP Links" (e.g., "APT28 uses Phishing"). Once again, the Bulletin-Telegram pair leads with 1,038 shared semantic links (Jaccard 3.55%). This result is profound from a strategic point of view: it means that the unstructured chats on Telegram and the formal reports in Bulletins are telling the *same story*. They are not just listing the same indicators; they are attributing them to the same actors and techniques, validating the reliability of Telegram as a source for high-level attribution.

| Source Overlap (APT-TTP Links) |              |                 |          |          |         |
|--------------------------------|--------------|-----------------|----------|----------|---------|
| SOURCE A                       | SOURCE B     | OVERLAP (LINKS) | UNIQUE A | UNIQUE B | JACCARD |
| Bulletin                       | Telegram     | 1,038           | 19,946   | 10,333   | 3.55%   |
| Discord                        | Telegram     | 86              | 571      | 10,333   | 0.79%   |
| Intelligence                   | Telegram     | 77              | 680      | 10,333   | 0.70%   |
| Discord                        | Intelligence | 6               | 571      | 680      | 0.48%   |
| Bulletin                       | Discord      | 90              | 19,946   | 571      | 0.44%   |
| Bulletin                       | Intelligence | 68              | 19,946   | 680      | 0.33%   |
| Discord                        | Report       | 2               | 571      | 56       | 0.32%   |
| Discord                        | Forum        | 1               | 571      | 251      | 0.12%   |
| Forum                          | Telegram     | 11              | 251      | 10,333   | 0.10%   |
| Bulletin                       | Forum        | 16              | 19,946   | 251      | 0.08%   |
| Bulletin                       | Report       | 16              | 19,946   | 56       | 0.08%   |
| Report                         | Telegram     | 7               | 56       | 10,333   | 0.07%   |

Figure 5.17: Strategic Overlap (APT-TTP Links). 1,038 semantic links between Bulletins and Telegram proves that they provide consistent attribution logic, not just raw indicators.

**C. "Big Data" Feeds Isolation:** The Intelligence category (OSINT Feeds) shows very low Jaccard scores with human sources (e.g., 0.38% with Bulletins), despite sharing 2,081 entities. This is largely a “denominator problem”: since Intelligence feeds contain nearly 1 million entities (mostly unique, short-lived IPs), the overlap (2k) becomes negligible compared to the total volume. This supports the idea that, while Feeds are essential for blocking (Firewall/SIEM), they operate as a separate high-velocity layer that only rarely overlaps with the strategic narrative found in Reports or Bulletins.

2024 Analysis

The 2024 dataset captures the consolidation phase of the modern intelligence topology. The overlap metrics reveal that, while the structural connection between official and underground sources was already formed, their semantic alignment depth was still evolving.

**A. The Establishment of the Bulletin-Telegram Axis:** Data from Figure 5.18 indicates that the "Central Axis" identified in 2025 was already the dominant structure in



2024. The **Bulletin-Telegram** pair exhibits a **Jaccard Similarity of 15.50%**, sharing **6,774 entities**. This high degree of intersection (remarkably close to the 16.87% observed in 2025) suggests that the *speed of propagation* for raw indicators (IPs, Hashes, CVEs) had already reached saturation. By 2024, the "Infosec Bots" on Telegram were already efficiently mirroring official advisories, creating a unified high-frequency information loop. The visual confirmation is provided by the Heatmap (Figure 5.19), where the interaction nodes involving Telegram and Bulletins are the most prominent.

Source Overlap

📄 CSV

| SOURCE A     | SOURCE B     | ENTITIES | UNIQUE A | UNIQUE B | JACCARD |
|--------------|--------------|----------|----------|----------|---------|
| Bulletin     | Telegram     | 6.774    | 32.180   | 18.300   | 15.50%  |
| Discord      | Telegram     | 1.619    | 10.836   | 18.300   | 5.88%   |
| Discord      | Forum        | 1.121    | 10.836   | 11.767   | 5.22%   |
| Forum        | Telegram     | 1.398    | 11.767   | 18.300   | 4.88%   |
| Bulletin     | Discord      | 1.957    | 32.180   | 10.836   | 4.77%   |
| Bulletin     | Forum        | 1.753    | 32.180   | 11.767   | 4.15%   |
| Discord      | Report       | 141      | 10.836   | 461      | 1.26%   |
| Report       | Telegram     | 214      | 461      | 18.300   | 1.15%   |
| Forum        | Report       | 112      | 11.767   | 461      | 0.92%   |
| Bulletin     | Report       | 263      | 32.180   | 461      | 0.81%   |
| Bulletin     | Intelligence | 741      | 32.180   | 121.581  | 0.48%   |
| Intelligence | Telegram     | 414      | 121.581  | 18.300   | 0.30%   |
| Discord      | Intelligence | 162      | 10.836   | 121.581  | 0.12%   |
| Forum        | Intelligence | 88       | 11.767   | 121.581  | 0.07%   |
| Intelligence | Report       | 19       | 121.581  | 461      | 0.02%   |
| Bulletin     | DLS          | 1        | 32.180   | 1.203    | 0.00%   |
| DLS          | Intelligence | 4        | 1.203    | 121.581  | 0.00%   |

Figure 5.18: Source Overlap Table (2024). The Bulletin-Telegram axis has 15.50% similarity, proving that the synchronization of raw indicators was established before the 2025 volume boom.

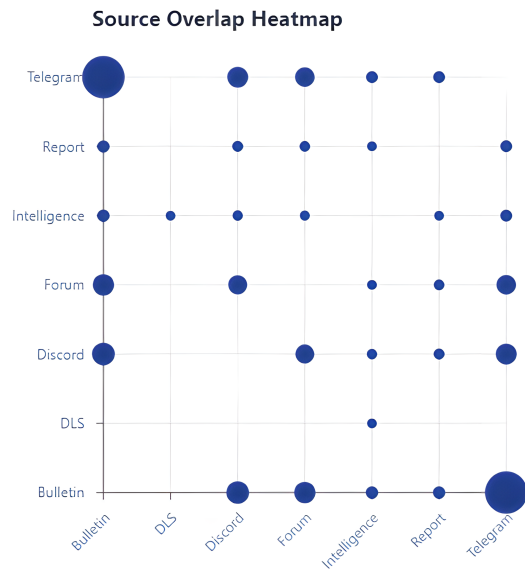


Figure 5.19: Overlap Heatmap (2024). High connectivity between Telegram, Forums, and Bulletins, contrasting with the isolation of DLS.

**B. The "Semantic Gap":** A critical evolutionary insight emerges when comparing the APT–TTP links (Figure 5.20) with the 2025 data. In 2024, the Bulletin–Telegram pair shared only 338 semantic links (Jaccard 2.49%). By contrast, 2025 shows 1,038 links, revealing a substantial “semantic gap”.

In 2024, Telegram was effective at mirroring *artifacts* (entities), but less consistent at mirroring *logic* (attribution). Automated bots were reposting CVEs and IPs, while the deeper contextual layer (e.g., linking Actor X to TTP Y) had not yet fully migrated from formal bulletins into the chat ecosystem. The tripling of this metric in 2025 suggests Telegram’s shift from a “simple feed” toward a more “strategic” source.

Source Overlap (APT-TTP Links)

| SOURCE A | SOURCE B | OVERLAP (LINKS) | UNIQUE A | UNIQUE B | JACCARD |
|----------|----------|-----------------|----------|----------|---------|
| Bulletin | Telegram | 338             | 10.177   | 3.760    | 2.49%   |
| Discord  | Report   | 1               | 128      | 152      | 0.36%   |
| Discord  | Telegram | 6               | 128      | 3.760    | 0.15%   |
| Forum    | Telegram | 6               | 240      | 3.760    | 0.15%   |
| Bulletin | Report   | 14              | 10.177   | 152      | 0.14%   |
| Report   | Telegram | 5               | 152      | 3.760    | 0.13%   |
| Bulletin | Forum    | 10              | 10.177   | 240      | 0.10%   |
| Bulletin | Discord  | 8               | 10.177   | 128      | 0.08%   |

Figure 5.20: Strategic Overlap (APT-TTP Links) in 2024. The shared links (338), lower compared to 2025 (1,038) highlights that Telegram had not yet fully matured as a source of attribution intelligence.

**C. The Isolation of DLS:** The overlap analysis definitively confirms the unique position of DLS. In the main overlap table, the Bulletin-DLS intersection is effectively null (**1 shared entity**, 0.00% Jaccard). This implies a total Isolation: the leak URLs of Ransomware sites are *never* republished in official Bulletins. This mathematically proves that DLS provides a layer of intelligence that is completely orthogonal to official sources; without direct DLS monitoring, this specific data class is entirely missed by the overlap network.

2023 Analysis

The 2023 dataset provides the historical baseline required to understand the evolution of cross-source synchronization. The overlap metrics depict an ecosystem where the "Echo Chamber" effect was already structural but semantically immature compared to the modern landscape.

**A. The "Embryonic" Axis:** As shown in Figure 5.21, the Bulletin-Telegram pair already exhibited the highest Jaccard Similarity index (14.16%) in 2023, sharing **6,454 entities**. The fact that the Jaccard index in 2023 (14.16%) is comparable to 2024 (15.50%) and 2025 (16.87%) proves that the *tendency* of Telegram channels to mirror official Bulletins is a permanent structural feature of the landscape, not a recent anomaly. However,

the absolute volume of intersection (6,454 entities) is nearly half of the 2025 volume ( $\approx 11k$ ). This reflects the lower adoption of automated scraping bots in 2023: while the "Reposting Behavior" was present, the *scale* of ingestion was significantly smaller.

### Source Overlap

| SOURCE A     | SOURCE B     | ENTITIES | UNIQUE A | UNIQUE B | JACCARD |
|--------------|--------------|----------|----------|----------|---------|
| Bulletin     | Telegram     | 6.454    | 32.770   | 19.252   | 14.16%  |
| Discord      | Forum        | 820      | 7.316    | 10.361   | 4.86%   |
| Forum        | Telegram     | 1.328    | 10.361   | 19.252   | 4.70%   |
| Discord      | Telegram     | 1.072    | 7.316    | 19.252   | 4.20%   |
| Bulletin     | Forum        | 1.509    | 32.770   | 10.361   | 3.63%   |
| Bulletin     | Discord      | 1.157    | 32.770   | 7.316    | 2.97%   |
| Report       | Telegram     | 397      | 967      | 19.252   | 2.00%   |
| Discord      | Report       | 149      | 7.316    | 967      | 1.83%   |
| Forum        | Report       | 184      | 10.361   | 967      | 1.65%   |
| Bulletin     | Report       | 483      | 32.770   | 967      | 1.45%   |
| Intelligence | Telegram     | 274      | 7.734    | 19.252   | 1.03%   |
| Bulletin     | Intelligence | 337      | 32.770   | 7.734    | 0.84%   |
| Discord      | Intelligence | 70       | 7.316    | 7.734    | 0.47%   |
| Forum        | Intelligence | 72       | 10.361   | 7.734    | 0.40%   |
| Intelligence | Report       | 19       | 7.734    | 967      | 0.22%   |

Figure 5.21: Source Overlap Table (2023). Bulletins and Telegram coupling is already evident (14.16% Jaccard).

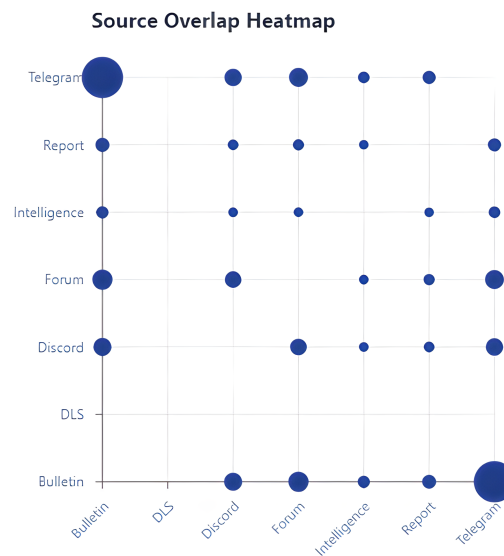


Figure 5.22: Overlap Heatmap (2023). The visualization confirms the density of the Bulletin-Telegram connection, while highlighting the total isolation of DLS (due to Survivorship Bias).

**B. The "Attribution Gap":** The most relevant finding can be seen from the APT-TTP Links analysis (Figure 5.23). In 2023, Bulletins and Telegram shared only 347 semantic links (Jaccard 3.12%). When compared to the 1,038 links observed in 2025, this demonstrates a threefold increase in "Attribution Maturity" over two years.

In 2023, Telegram acted primarily as a *News Feed* (sharing the "What" - CVEs/IPs). It had not yet evolved into a source of *Strategic Intelligence* (sharing the "Who" and "How" - APTs and TTPs). The complex attribution logic found in official Bulletins was not effectively propagating to the underground chats, creating a "Semantic Gap" that has only been closed in the most recent temporal window.

| Source Overlap (APT-TTP Links) |          |                 |          |          |         |
|--------------------------------|----------|-----------------|----------|----------|---------|
| SOURCE A                       | SOURCE B | OVERLAP (LINKS) | UNIQUE A | UNIQUE B | JACCARD |
| Bulletin                       | Telegram | 347             | 7.544    | 3.927    | 3.12%   |
| Discord                        | Telegram | 15              | 66       | 3.927    | 0.38%   |
| Report                         | Telegram | 11              | 250      | 3.927    | 0.26%   |
| Bulletin                       | Report   | 18              | 7.544    | 250      | 0.23%   |
| Bulletin                       | Discord  | 8               | 7.544    | 66       | 0.11%   |
| Bulletin                       | Forum    | 2               | 7.544    | 141      | 0.03%   |
| Forum                          | Telegram | 1               | 141      | 3.927    | 0.02%   |

Figure 5.23: Strategic Overlap (APT-TTP Links) in 2023. Telegram had not yet matured into a source capable of mirroring complex attribution logic (low volume of shared semantic links, 347).

**C. Discord and Forums, the Community Cohesion:** Unlike 2025, where Discord appears more isolated, the 2023 data shows a relatively strong coupling between Discord and Forums (Jaccard 4.86%). This suggests that in the “legacy” landscape, unstructured communities (chats and forums) were more tightly clustered, sharing low-level indicators among themselves. As the ecosystem evolved toward 2025, Telegram appears to have broken away from this cluster and aligned more closely with official bulletins, leaving Discord and forums as secondary, less synchronized communities.

5.3.4. Timeliness and Celerity Analysis

The Source Anticipation Analysis summarizes, over the full 2023–2025 window, how often one source family is observed before another on the same entities. This provides a pairwise view of which channels tend to surface *initial signals* versus those that mainly *validate, consolidate, and redistribute* information already disclosed elsewhere. Overall, Forums and Telegram frequently anticipate downstream aggregators: Forums lead Intelligence in most shared cases (80.5%), and Telegram leads both Intelligence (74.9%), Reports and Discord (65.8%; 64.5%), consistent with their role as rapid signal channels. Conversely, Bulletins remain the main driver of the standard disclosure cycle, anticipating Intelligence, Discord and Reports (73.4%; 69.4%; 68.4%). The large Bulletin–Telegram intersection (8,565 shared entities) combined with a near-neutral win rate (50.7%) indicates

strong synchronization, i.e., rapid reposting rather than a clear lead.



Figure 5.24: Source Anticipation Matrix (2023–2025). The analysis shows how often one source family anticipates another across the longitudinal dataset (ties are excluded from the evaluation).

Since this table aggregates the entire three-year period, the following sections decompose

the same celerity analysis year-by-year (2023, 2024, 2025) using the “First-to-Report” ranking and the First-Seen Matrix to highlight how these dynamics evolve over time.

## 2025 Analysis

The “Celerity” analysis determinates the temporal performance of the sources, evaluating which category is the *First-to-Report* for a specific entity. This metric is critical for shifting the CTI posture from reactive (patching after a Bulletin) to proactive (mitigating based on underground discussions).

**A. The “First-to-Report” Hierarchy:** Figure 5.25 aggregates the number of times each source was the absolute first to detect a new entity. Unsurprisingly, official Bulletins hold the top rank (9,663 times “First-to-Report”). This represents the “Standard Disclosure” process: when a vendor releases a patch and a CVE simultaneously, the Bulletin is mathematically the first and authoritative source. Telegram is the second fastest source (6,342 times “First-to-Report”), far outpacing Forums and Reports. This implies that for over 6,000 entities, the initial signal did not come from a vendor advisory but from a real-time chat message. This confirms Telegram’s role as the *de facto* “Zero-Day” channel, where proofs-of-concept (PoCs) or active exploitation claims are shared before official acknowledgment. Despite its high volume (analyzed in Sec. 5.3.1), Intelligence feeds are rarely the *first* to define an entity (1,246 times “First-to-Report”). They act as “echoes”: once a threat is active, they generate thousands of IP indicators, but the *semantic* discovery (e.g., the CVE ID itself) usually originates elsewhere.



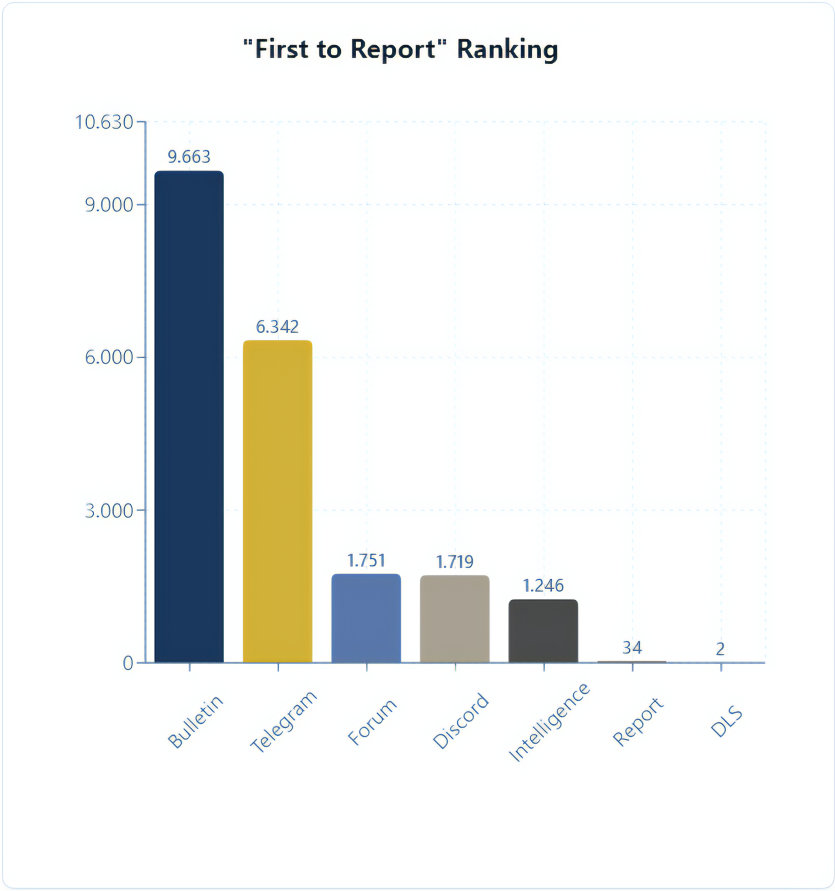


Figure 5.25: "First to Report" Ranking (2025). Bulletins lead the standard disclosure cycle, but Telegram’s high rank (6,342) confirms its capability to preempt official channels.

**B. Temporal Matrix Analysis: Synchronization vs. Anticipation**

The "First Seen Matrix" (Figure 5.26) offers a granular view of specific detection timestamps for sample CVEs. It analyzes the timestamps revealing two distinct behavioral patterns in the 2025 landscape:

**1. The "Synchronized Disclosure" Pattern:** We observe a near-perfect synchronization for entities like CVE-2025-43200 and CVE-2025-53770.

*Example (CVE-2025-53770):* Bulletin, Discord, Intelligence, and Telegram all recorded the entity on the exact same date (**20/07/2025**).

This pattern indicates a coordinated public release. The automated pipelines (Telegram Bots and Intelligence Feeds) ingested the official Bulletin immediately upon publication. In this case, the "underground" acts merely as a distribution amplifier.

**2. The "Underground Anticipation" Pattern:** The most valuable insights arise

where *Forums* significantly predate official *Bulletins*.

*Example (CVE-2025-61882):* Forum category detected this entity on 18/08/2025, while the official Bulletin appeared on 06/10/2025.

It is important to note that Forum date aligns with the initial release of the primary Thread. Although subsequent comments, such as those regarding the CVE on Dread, were published later, proactive monitoring of the original thread could have significantly mitigated the window of exposure, potentially preventing the exploitation of the vulnerability.

The evaluated scenario represents a **49-day lead time**. An analyst monitoring Forums would have been aware of this vulnerability nearly two months before the vendor issued a formal warning.

The disparity is even more extreme in the *Example of (CVE-2025-55182)*. The entity appeared in Forums on 02/03/2025, but the Bulletin is dated 03/12/2025. A massive 9-month anticipation. This suggests a scenario where a vulnerability was discussed, traded, or exploited as a "Zero-Day" in the underground long before it was patched or publicly disclosed (Zero-Day Brokerage scenario).

This matrix empirically validates the thesis: while Bulletins provide *certainty*, Forums and Telegram provide *time*. The platform’s ability to correlate these timestamps highlights "Blind Spots" where organizations relying only on official feeds would remain exposed for months.

First Seen Matrix

| ENTITY         | TYPE  | GLOBAL FIRST SEEN | BULLETIN   | DLS | DISCORD    | FORUM      | INTELLIGENCE | REPORT   | TELEGRAM   |
|----------------|-------|-------------------|------------|-----|------------|------------|--------------|----------|------------|
| CVE-2025-43200 | VULID | 12/6/2025         | 12/6/2025  | -   | 12/6/2025  | 28/6/2025  | 16/6/2025    | 1/7/2025 | 12/6/2025  |
| CVE-2025-14847 | VULID | 30/10/2025        | 27/12/2025 | -   | 26/12/2025 | 16/12/2025 | 29/12/2025   | -        | 30/10/2025 |
| CVE-2025-61882 | VULID | 18/8/2025         | 6/10/2025  | -   | 5/10/2025  | 18/8/2025  | 6/10/2025    | -        | 6/10/2025  |
| CVE-2025-59718 | VULID | 5/8/2025          | 10/12/2025 | -   | 16/12/2025 | 5/8/2025   | 16/12/2025   | -        | 9/12/2025  |
| CVE-2025-59719 | VULID | 5/8/2025          | 10/12/2025 | -   | 16/12/2025 | 5/8/2025   | 18/12/2025   | -        | 9/12/2025  |
| CVE-2025-53770 | VULID | 20/7/2025         | 20/7/2025  | -   | 20/7/2025  | -          | 20/7/2025    | 4/8/2025 | 20/7/2025  |
| CVE-2025-3248  | VULID | 5/5/2025          | 6/5/2025   | -   | 5/7/2025   | -          | 5/5/2025     | 3/6/2025 | 6/5/2025   |
| CVE-2025-55182 | VULID | 2/3/2025          | 3/12/2025  | -   | 6/12/2025  | 2/3/2025   | 5/12/2025    | -        | 5/11/2025  |

Figure 5.26: First Seen Matrix (2025). Rows highlighted in green show the earliest detection.

2024 Analysis

The 2024 window serves as the control baseline for the temporal analysis, depicting a landscape still dominated by the "Standard Disclosure Cycle". Unlike the disruption

observed in 2025, the 2024 data indicates that institutional sources maintained a distinct temporal advantage over underground communities.

**A. The Supremacy of Standard Disclosure:** As evidenced by the ranking in Figure 5.27, Bulletins were the "First to Report" source for 8,286 entities, establishing a massive lead over all other categories. This metric confirms that in 2024, the primary vector for threat dissemination was strictly top-down: Vendors identified vulnerabilities and released patches or advisories (Bulletins) before the underground communities could react.

The "First Seen" volume events in Bulletins is nearly triple that of Telegram (2,735) and five times that of Forums (1,654). This statistical distribution proves that the "Anticipation Phenomenon" observed in 2025 was not yet systemic in 2024. In this period, Forums and Chats functioned primarily as "Echo Chambers," discussing vulnerabilities *after* their official release.

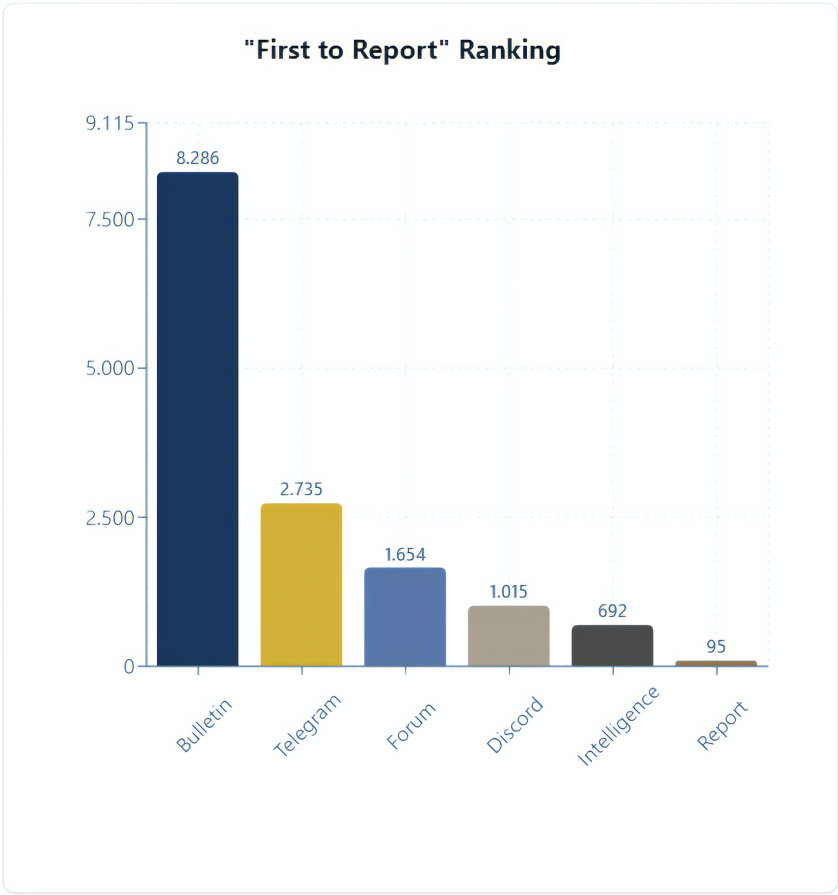


Figure 5.27: "First to Report" Ranking (2024). The dominance of Bulletins (8,286) confirms that the "Standard Disclosure" model was the norm, with underground sources mostly reacting to official news.

**B. Temporal Matrix - Synchronization and Outliers:** The "First Seen Matrix" (Figure 5.28) provides granular confirmation of this reactive posture.

For most of high-profile entities (e.g., CVE-2024-1709, CVE-2024-21412), the Global First Seen timestamp coincides perfectly with the Bulletin date.

This pattern indicates that automated scrapers (Intelligence and Telegram Bots) were ingesting data in real-time *from* official Bulletins. There is no "underground lead"; the ecosystem is synchronized on the vendor’s clock.

The dataset also reveals rare but significant "Outliers" that signal the incoming shift of 2025. The case of CVE-2024-4577 is the most prominent exception.

We have the Forum Detection on 12/02/2024, while the Bulletin Disclosure happens on 08/06/2024

While statistically rare in 2024, this 4-month anticipation demonstrates that the capability for early warning existed within the Forums, specifically for Zero-Day exploits traded before disclosure. In 2024, this was an anomaly; in 2025, as analyzed in the previous section, this "Blind Spot" coverage becomes a more frequent structural feature of the landscape.

First Seen Matrix

| ENTITY         | TYPE  | GLOBAL FIRST SEEN | BULLETIN   | DLS | DISCORD    | FORUM      | INTELLIGENCE | REPORT    | TELEGRAM  |
|----------------|-------|-------------------|------------|-----|------------|------------|--------------|-----------|-----------|
| CVE-2024-9680  | VULID | 9/10/2024         | 10/10/2024 | -   | 16/11/2024 | 26/10/2024 | 15/10/2024   | -         | 9/10/2024 |
| CVE-2024-1709  | VULID | 20/2/2024         | 20/2/2024  | -   | 23/2/2024  | -          | 22/2/2024    | 3/4/2024  | 21/2/2024 |
| CVE-2024-21412 | VULID | 12/2/2024         | 12/2/2024  | -   | 14/3/2024  | -          | 13/2/2024    | 13/2/2024 | 13/2/2024 |
| CVE-2024-4577  | VULID | 12/2/2024         | 8/6/2024   | -   | 6/6/2024   | 12/2/2024  | 12/6/2024    | -         | 7/6/2024  |

Figure 5.28: First Seen Matrix (2024). Official sources were usually the first to detect threats. CVE-2024-4577 (Forum lead) represents a rare strategic outlier.

2023 Analysis

The 2023 dataset represents the historical baseline of the study, painting an ecosystem governed by the traditional "Vendor-First" disclosure model. However, the granular analysis of detection timestamps reveals niche behaviors in unstructured communities, specifically Discord and Forums, that challenge the aggregate trends.

**A. The Hegemony of Official Disclosure:** Figure 5.29 confirms the structural dominance of institutional sources in the 2023 landscape. Bulletins were the "First to Report" source for 7,189 entities, more than doubling the contribution of Telegram (3,011). This distribution defines the "Legacy CTI Lifecycle": in 2023, the standard propagation path

involved a vendor discovering a flaw, assigning a CVE, and publishing a Bulletin. Underground sources (Telegram/Forums) typically acted as *reactive* amplifiers, discussing the vulnerability only *after* the official ID was assigned. The "Telegram Anticipation" observed in 2025 was present here only in an embryonic state.

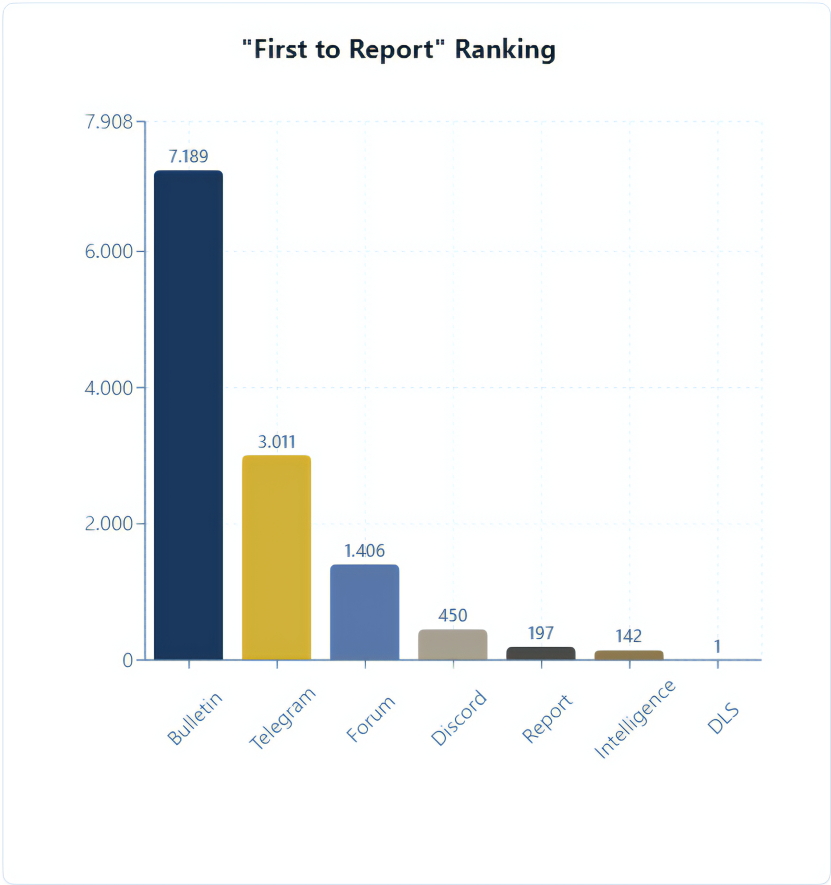


Figure 5.29: "First to Report" Ranking (2023). The landscape is dominated by Bulletins (7,189), confirming a reactive CTI model where official vendors initiate the disclosure cycle.

**B. Temporal Matrix:** While the aggregate volume goes in favour of Bulletins, the "First Seen Matrix" (Figure 5.30) highlights critical exceptions where unstructured communities broke the standard cycle.

A distinct cluster of vulnerabilities (specifically CVE-2023-41991, CVE-2023-41992, CVE-2023-41993, and CVE-2023-41061) was detected first on Discord in September/October 2023.

This clustering suggests a "Community Silo" effect. Unlike Telegram (which often acts as a broadcast channel), Discord operates through semi-private servers often dedicated to

specific hacking niches (e.g., iOS jailbreaking, game cheats, or exploit development). The data indicates that active exploitation or proof-of-concept development for these specific CVEs was occurring collaboratively within a Discord community *before* the wider threat intelligence community (Bulletins and Feeds) caught up. In 2023, Discord acted as a "Tactical Incubator" for specific exploit chains.

The matrix recovers also instances of extreme anticipation from traditional Forums, validating their role as the historical home for Zero-Days.

The critical MOVEit Transfer vulnerability (Case Study -CVE-2023-34362) was detected in Forums on 24/04/2023, while the official Bulletin appears in the dataset on 26/09/2023 (a gap of 5 months).

Another example is relative to CVE-2023-37450, that was detected in Forums on 22/04/2023, with the Bulletin following on 25/07/2023 (a gap of 3 months).

These massive lead times confirm the fact that in 2023, while Forums had lower volume than Bulletins, they possessed the highest "Quality of Anticipation". 5-month lead time on a critical vulnerability like MOVEit represents the difference between a prevented breach and a catastrophic ransomware incident.

First Seen Matrix

| ENTITY         | TYPE  | GLOBAL FIRST SEEN | BULLETIN   | DLS | DISCORD    | FORUM     | INTELLIGENCE | REPORT     | TELEGRAM  |
|----------------|-------|-------------------|------------|-----|------------|-----------|--------------|------------|-----------|
| CVE-2023-37450 | VULID | 22/4/2023         | 25/7/2023  | -   | 18/7/2023  | 22/4/2023 | 13/7/2023    | -          | -         |
| CVE-2023-41064 | VULID | 11/9/2023         | 11/9/2023  | -   | 12/9/2023  | -         | 11/9/2023    | -          | -         |
| CVE-2023-38606 | VULID | 25/7/2023         | 25/7/2023  | -   | 26/12/2023 | -         | 26/7/2023    | -          | -         |
| CVE-2023-3519  | VULID | 19/7/2023         | 23/7/2023  | -   | 21/7/2023  | -         | 19/7/2023    | -          | -         |
| CVE-2023-32434 | VULID | 23/6/2023         | 25/7/2023  | -   | 19/9/2023  | -         | 23/6/2023    | -          | -         |
| CVE-2023-32435 | VULID | 23/6/2023         | 25/7/2023  | -   | 26/12/2023 | -         | 23/6/2023    | -          | -         |
| CVE-2023-34362 | VULID | 24/4/2023         | 26/9/2023  | -   | -          | 24/4/2023 | -            | -          | 26/9/2023 |
| CVE-2023-47565 | VULID | 17/12/2023        | 17/12/2023 | -   | -          | -         | 21/12/2023   | -          | -         |
| CVE-2023-49897 | VULID | 17/12/2023        | 17/12/2023 | -   | -          | -         | 21/12/2023   | -          | -         |
| CVE-2023-42916 | VULID | 4/12/2023         | 12/12/2023 | -   | -          | -         | 4/12/2023    | -          | -         |
| CVE-2023-42917 | VULID | 4/12/2023         | 12/12/2023 | -   | -          | -         | 4/12/2023    | -          | -         |
| CVE-2023-49103 | VULID | 30/11/2023        | -          | -   | 4/12/2023  | -         | 30/11/2023   | -          | -         |
| CVE-2023-38831 | VULID | 3/11/2023         | 8/11/2023  | -   | -          | -         | -            | 3/11/2023  | -         |
| CVE-2023-4956  | VULID | 17/10/2023        | 17/10/2023 | -   | -          | -         | 18/10/2023   | -          | -         |
| CVE-2023-42793 | VULID | 4/10/2023         | -          | -   | -          | -         | 4/10/2023    | 13/12/2023 | -         |
| CVE-2023-41991 | VULID | 21/9/2023         | -          | -   | 21/9/2023  | -         | 25/9/2023    | -          | -         |
| CVE-2023-41992 | VULID | 21/9/2023         | -          | -   | 21/9/2023  | -         | 25/9/2023    | -          | -         |
| CVE-2023-41993 | VULID | 21/9/2023         | -          | -   | 21/9/2023  | -         | 25/9/2023    | -          | -         |
| CVE-2023-41061 | VULID | 7/9/2023          | -          | -   | 7/9/2023   | -         | 11/9/2023    | -          | -         |
| CVE-2023-35674 | VULID | 6/9/2023          | 6/9/2023   | -   | -          | -         | 13/9/2023    | -          | -         |
| CVE-2023-2868  | VULID | 26/5/2023         | 8/6/2023   | -   | -          | -         | 26/5/2023    | -          | -         |
| CVE-2023-28204 | VULID | 18/5/2023         | -          | -   | 18/5/2023  | -         | 22/5/2023    | -          | -         |
| CVE-2023-32373 | VULID | 18/5/2023         | -          | -   | 18/5/2023  | -         | 22/5/2023    | -          | -         |
| CVE-2023-29336 | VULID | 9/5/2023          | -          | -   | 18/8/2023  | -         | 9/5/2023     | -          | -         |
| CVE-2023-27350 | VULID | 21/4/2023         | 20/4/2023  | -   | -          | -         | 21/4/2023    | -          | -         |
| CVE-2023-28205 | VULID | 18/4/2023         | -          | -   | 11/5/2023  | -         | 18/4/2023    | -          | -         |
| CVE-2023-30190 | VULID | 21/3/2023         | 21/3/2023  | -   | 22/7/2023  | -         | -            | -          | -         |
| CVE-2023-23397 | VULID | 21/3/2023         | 21/3/2023  | -   | -          | -         | -            | -          | 17/7/2023 |
| CVE-2023-23529 | VULID | 14/2/2023         | -          | -   | 14/2/2023  | -         | 14/2/2023    | -          | -         |
| CVE-2022-41080 | VULID | 6/1/2023          | 6/1/2023   | -   | -          | -         | 10/1/2023    | -          | -         |

Figure 5.30: First Seen Matrix (2023). Specific exploit development community active prior to official disclosure in Discord.

5.3.5. Strategic Coverage and Blind Spots

2025 Analysis

The "Strategic Coverage" analysis evaluates the semantic diversity of the intelligence sources. By categorizing the extracted entities into threat classes (e.g., APT, Malware, Tool, Vulnerability), we identify the specific intelligence domain where each source provides visibility, and conversely, where it leaves analysts blind.

**A. "Specialist" vs. "Generalist" Dichotomy:** Figure 5.31 reveals a functional distinction between automated feeds and human-curated channels.

The Intelligence category exhibits an extreme specialization. It dominates the **MAL (Malware)** class, with volumes (> 100,000) orders of magnitude higher than any other source. However, it shows a critical Blind Spot regarding VULID (Vulnerabilities) and ACT (Tactics, Techniques, and Procedures). This confirms that OSINT feeds are engineered to stream technical artifacts (hashes, signatures) but fail to provide the strategic



context of *who* is attacking or *which* vulnerability is being exploited.

The Data Leak Sites category presents the most restricted coverage profile, visible in the chart as a single bar for APT (Threat Actor). Due to the deterministic nature of the scraper, which maps the "Group Name" field to the APT entity, this source is hyper-specialized. Relying only on DLS provides high fidelity on active Ransomware groups but leaves the analyst completely blind to the tools (TOOL) or vulnerabilities (VULID) used to perpetrate the breach.

**B. Telegram as a "Full-Spectrum" Source:** The most significant finding for 2025 is the coverage profile of Telegram. Unlike Intelligence (skewed to Malware) or DLS (skewed to Actors), Telegram displays a Balanced Polymathic Coverage. It registers high volumes across all strategic categories: It rivals Bulletins in tracking hacking tools (TOOL), acting as a marketplace for exploits; it effectively mirrors official advisories in tracking CVEs (VULID/VULNAME) and it provides context on threat actors (ACT/APT).

This "Full-Spectrum" capability empirically validates why Telegram has displaced other sources: it is the only single channel where an analyst can observe the *Actor*, the *Weapon* (Tool), and the *Target* (Vulnerability) simultaneously.

**C. The "Weaponization" Channels:** An interesting insight emerges from comparing Discord and Forums with Bulletins. While Bulletins dominate the *definitions* of vulnerabilities (VULID), Discord and Forums show a disproportionately high ratio of TOOL entities relative to their total volume. This indicates that these unstructured communities are not discussing the *administrative* aspect of threats (CVE IDs), but rather the *practical and technical* aspect (Exploits, Scripts, Panels). They are the primary sources for monitoring the "Weaponization" of a threat, covering a blind spot that official Bulletins, which focus on remediation, often omit.



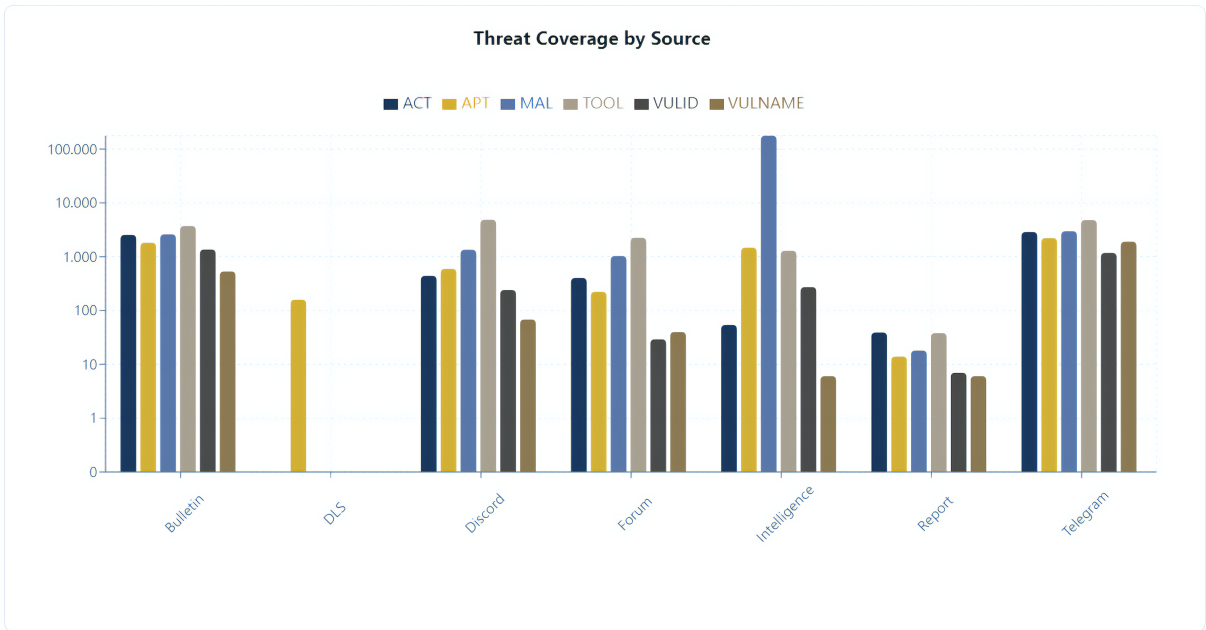


Figure 5.31: Threat Coverage by Source (2025). Intelligence (Malware-focused) and DLS (APT-focused), contrasted with the broad, full-spectrum coverage of Telegram and Bulletins.

### 2024 Analysis

The 2024 dataset depicts a threat landscape where the "Semantic Leadership" is still firmly held by institutional sources, although the underground communities exhibit distinct specialization patterns regarding the weaponization of threats.

**A. The Institutional Hegemony of Bulletins:** In strong contrast to the balance observed in 2025, the 2024 coverage profile (Figure 5.32) shows Bulletins as the undisputed "Generalist" leader. They register the highest volumes across the critical strategic triad: ACT (TTP), MAL (Malware), and VULID (Vulnerability). This indicates that in 2024, the holistic description of a threat, connecting the technique to the malware used and the vulnerability exploited, was primarily the domain of official vendor advisories. Underground sources like Telegram provided supporting data but had not yet aggregated sufficient volume to challenge the comprehensive scope of Bulletins.

**B. Discord, The "Toolkit" Repository:** A distinct functional anomaly appears in the Discord category. While its overall volume is moderate, Discord exhibits a disproportionately high peak for TOOL, which serves as its dominant category ( $\approx 4,000$  entities), surpassing even its malware mentions.

This coverage profile empirically characterizes Discord in 2024 as a **"Tactical deposit"**.

It was not the primary venue for high-level APT attribution (low APT count) or vulnerability disclosure (low VULID count). Instead, it functioned as a collaborative environment for sharing the operational scripts, panels, and exploit kits (Tools) required to execute attacks.

**C. The Static Role of Intelligence Feeds:** The Intelligence category confirms its structural rigidity. Consistent with the 2025 findings, it shows a massive spike in MAL (Malware) and strong APT coverage, but essentially zero visibility into TOOL or VULID. This supports the hypothesis that OSINT feeds are “indicator-centric” rather than “context-centric”: they answer “What is the hash?” but typically do not capture “How is it used?” (Tool) or “What does it target?” (Vulnerability), leaving these strategic dimensions to human-curated sources.

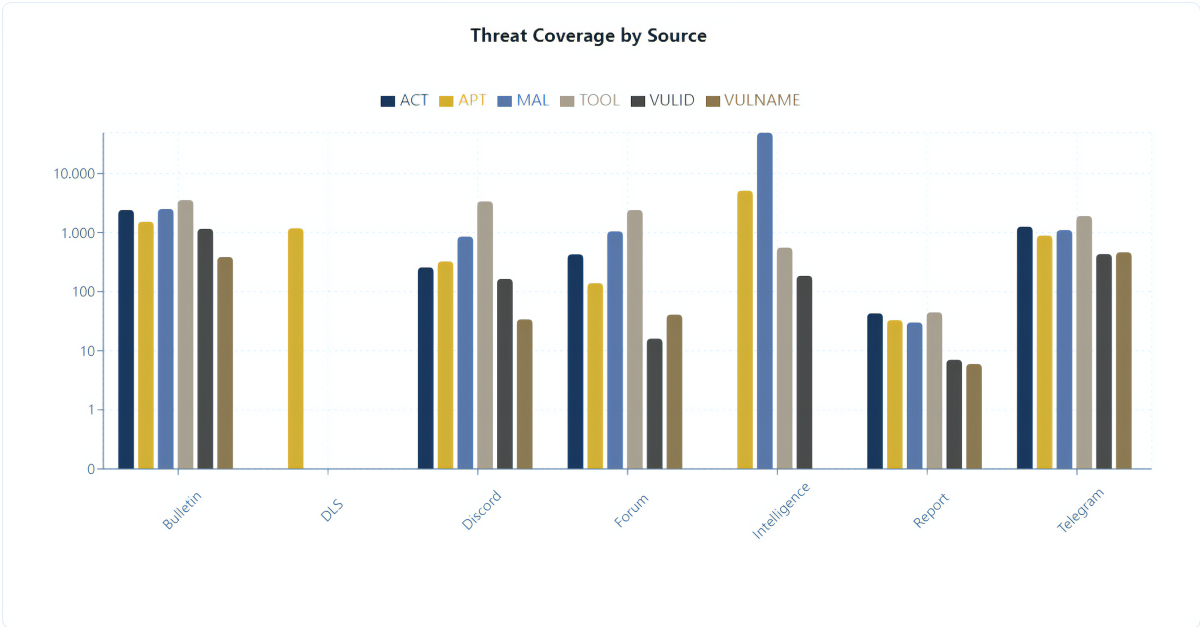


Figure 5.32: Threat Coverage by Source (2024). Bulletins leadership covering Actors, Malware, and Vulnerabilities. Discord shows a specific specialization in TOOLS, surpassing other underground sources in this domain.

## 2023 Analysis

2023 dataset serves as the historical baseline for the semantic distribution of threat intelligence. Unlike the fragmented landscape of later years, 2023 coverage metrics depict a centralized ecosystem where official advisories provided the most holistic view of threats, while underground communities exhibited sharp functional specializations.

**A. The Semantic Completeness of Bulletins:** Figure 5.33 identifies Bulletins as the

"Semantic Standard" of 2023. This category is the only one maintaining high and balanced volumes across all critical dimensions: APT (Actor), ACT (TTP), MAL (Malware), TOOL (Tools), and VULID (Vulnerabilities). This balance implies that in the "Legacy" landscape, the contextual linking of a threat, answering "Who used What against Which vulnerability", was predominantly performed by official vendors. An analyst relying solely on Bulletins in 2023 would have a complete, potentially delayed, strategic picture.

**B. Discord - The "Operational Warehouse":** While Discord generally has lower volumes than Telegram, its ratio of Tool-related entities to other categories is significantly higher.

This suggests that in 2023, Discord communities were functionally distinct from the more "news feed" nature of Telegram. Discord acted as an operational warehouse for exchanging offensive capabilities (scripts, exploit kits, panels). While bulletins described the *vulnerability* (VULID), Discord users were actively sharing the *means* to exploit it (TOOL), addressing a specific tactical blind spot of official reporting.

**C. The Structural Blind Spots of Intelligence Feeds:** Intelligence activity is concentrated in two separate phases, resulting in a distribution with two distinct peaks, dominated exclusively by APT and MAL entities. The almost total absence of VULID and TOOL entities confirms that this limitation is structural and not temporal. Across all three years (2023-2025), OSINT feeds have functioned strictly as "Attribution and Identification" engines. They effectively answer "Who is it?" and "Is it malicious?", but they consistently fail to explain the "Methodology" of the attack, leaving the analyst blind to the vulnerability vectors and tooling involved unless supplemented by human-curated sources like Bulletins or Telegram.

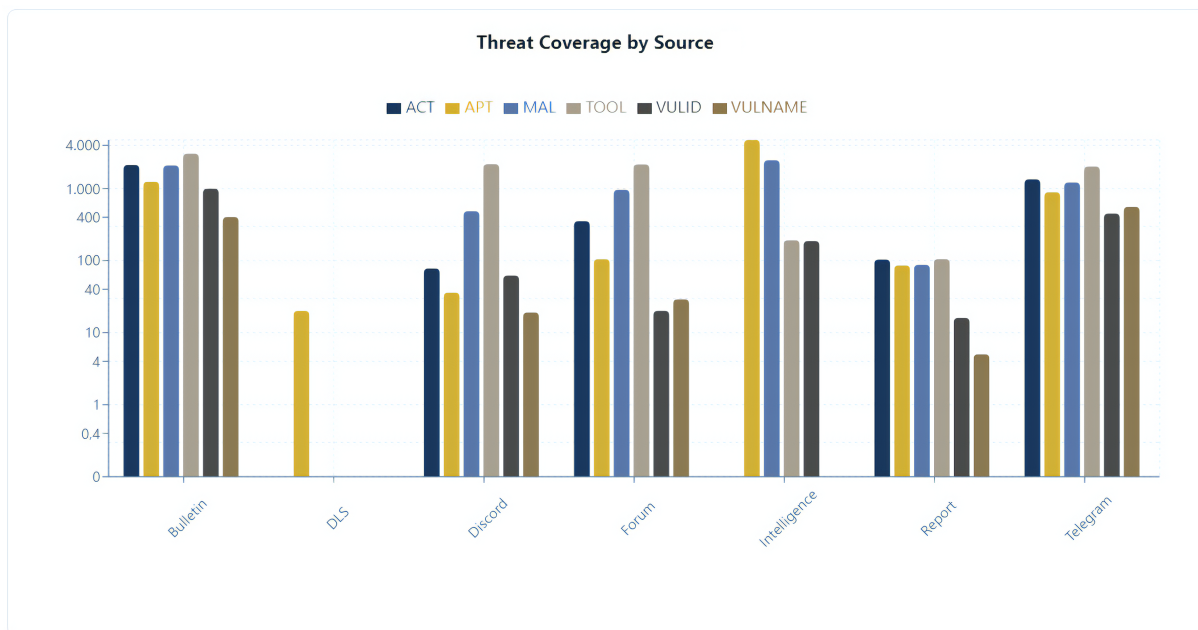


Figure 5.33: Threat Coverage by Source (2023). Bulletins display the most balanced coverage. Discord stands out for its high density of TOOLS, while Intelligence feeds remain restricted to Actors and Malware.

### 5.3.6. Attribution Capabilities

#### 2025 Analysis

“Attribution Capabilities” analysis measures how well a platform links a specific threat event to a named threat actor. In practice, Argus counts an attribution link whenever a single intelligence item (e.g., post, bulletin, report) contains both an extracted ACT entity (Attack Pattern) and an extracted APT entity (Actor Label). The co-occurrence of ACT and APT within the same item is treated as a candidate event-to-actor association, and the counts are aggregated across items to quantify how often each source provides actionable attribution context.

**A. The "Attribution Parity":** Figure 5.34 reveals a remarkable structural equivalence between the top two sources (Bulletin and Telegram).

Official advisories (1,630 Linked APTs) maintain the lead. This is expected, as formal attribution (often involving geopolitical considerations) is a primary function of major vendors. The critical insight is how closely Telegram (1,386 Linked APTs) tracks the official volume. In previous years (see subsequent sections), the gap was much wider. The fact that Telegram now provides attribution for nearly 1,400 distinct actors confirms its evolution into a "Claims of Responsibility" channel. Hacktivist groups, Ransomware

affiliates, and independent researchers use Telegram to explicitly claim attacks effectively turning the chat ecosystem into a primary attribution engine that rivals official sources in breadth.

**B. "Attribution Gap" of Technical Feeds:** The analysis of the Intelligence category highlights a key limitation of automated OSINT. Despite producing the highest volume of raw entities (881k+), Intelligence feeds linked only 91 APTs. This contrast captures the “contextual blindness” of technical feeds: they are very effective at streaming short-lived indicators (What/Where), but they provide little attribution context (Who). An analyst relying only on feeds would be flooded with alerts while remaining largely blind to the adversary’s identity.

**C. The "Functional Focus" of Communities:** Discord (95) and Forums (81) show low attribution capabilities. This aligns with the findings in Section 5.3.5 regarding their focus on *Tools* and *Exploits*. The users in these communities are primarily interested in the "means" of the attack (the "How"), rather than the geopolitical identity of the attacker. Consequently, while they are excellent sources for finding a new exploit script, they are poor sources for understanding the threat actor landscape.

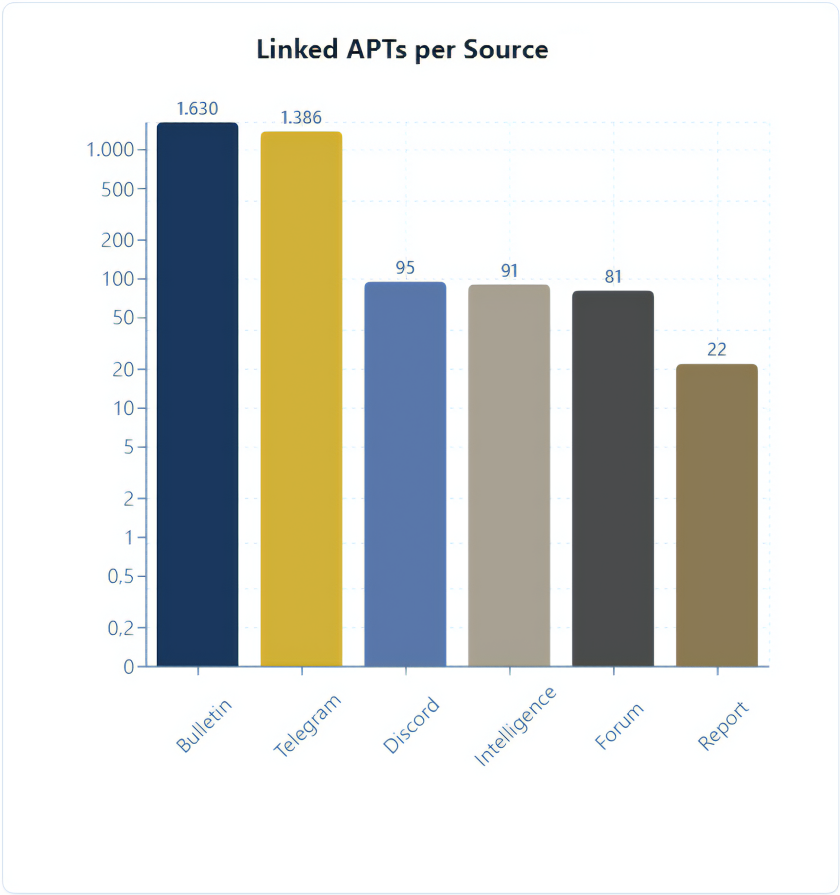


Figure 5.34: Linked APTs per Source (2025). Bulletins (1,630) and Telegram (1,386) coupling, they are the two primary engines for attribution. Intelligence feeds show minimal capability in this domain.

2024 Analysis

2024 dataset depicts a transitional phase in strategic attribution. Unlike the near-equivalence observed in 2025, the 2024 landscape is characterized by a distinct "Attribution Gap" between established institutional authorities and the emerging underground ecosystem.

**A. The Institutional Hegemony of Bulletins:** As shown in Figure 5.35, Bulletins maintain a dominant lead with 1,255 Linked APTs. This metric is nearly double that of the nearest competitor (Telegram, 699). This distribution reflects the traditional CTI hierarchy of 2024: the complex task of attributing a cyberattack to a specific criminal group was still predominantly the reserve of well-resourced threat intelligence teams publishing formal advisories. In 2024, the "Truth" regarding an attacker’s identity flowed top-down.

**B. Telegram, The Sole Contender:** While significantly trailing Bulletins, Telegram (699 Linked APTs) effectively isolates itself from the rest of the underground.

Its attribution volume is considerably higher than Forums (82) and Discord (47). This signals that by 2024, the "center of gravity" for hacktivist claims and ransomware shaming had already migrated away from web forums. While Telegram had not yet reached the parity seen in 2025, it had already established itself as the exclusive underground channel for attribution. If a threat actor wanted to claim credit for an attack in 2024, they were already doing it on Telegram, not on XSS or BreachForums.

**C. The Strategic Irrelevance of Other Sources:** The combined attribution capability of Forums, Discord, and Reports is negligible (< 100 each). This supports the functional specialization hypothesis: these communities act as tactical hubs. A user on Discord is more likely to share a “Nitro Generator” or a specific exploit script, but typically lacks the visibility or interest to connect that tool to “APT29”. As a result, these sources remain largely “attribution-neutral”, providing the *means* of the attack but rarely the *identity* of the attacker.

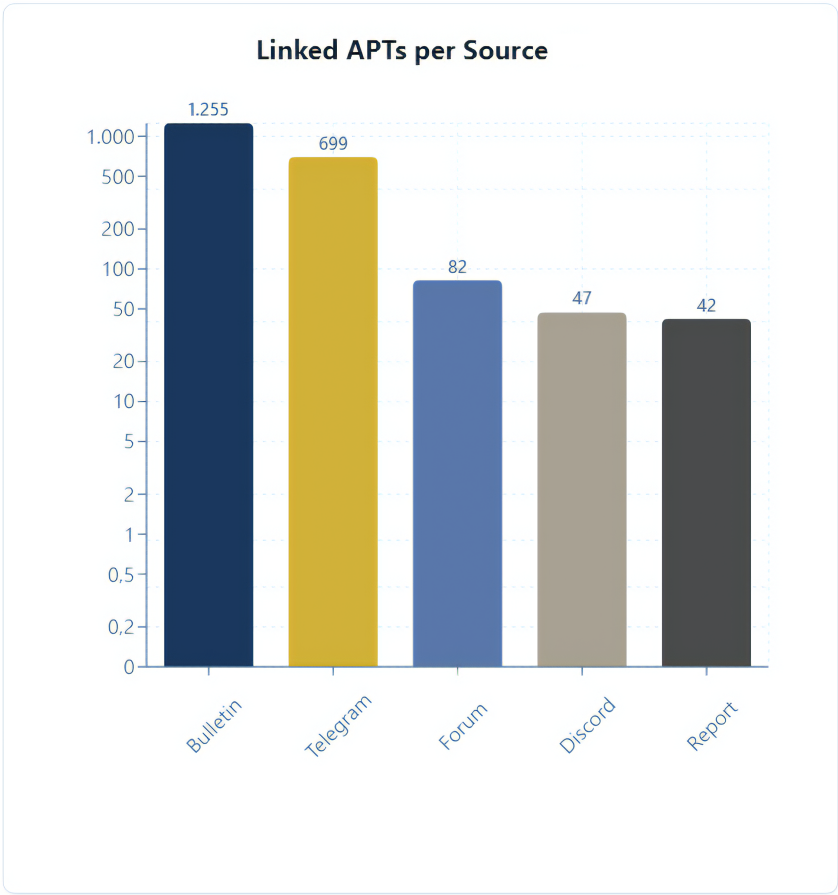


Figure 5.35: Linked APTs per Source (2024). Bulletins (1,255) lead significantly, but Telegram (699) has already substituted traditional Forums as the primary underground source for actor identification.

## 2023 Analysis

2023 dataset serves as the control group for the longitudinal study, it illustrates the traditional hierarchy of threat attribution before the structural shifts observed in later years. The data characterizes an ecosystem where "Strategic Truth", the identity of the adversary, was established top-down by institutional authorities.

**A. The importance of Institutional Attribution:** Figure 5.36 confirms that in 2023, Bulletins were the primary engine for actor identification, recording **1,112 Linked APTs**. This volume is nearly double that of the nearest competitor (Telegram, 667). This distribution reflects the "Legacy CTI Model": attribution requires significant visibility and forensic resources, which in 2023 were predominantly possessed by major vendors and government agencies. Consequently, the "Who" of cyberattacks was a narrative controlled by official advisories.

**B. Telegram, The Emerging Echo Chamber:** In 2023, Telegram linked 667 APTs. While this establishes it as the second-most important source, the qualitative nature of this volume differs from 2025.

In 2025, Telegram rivals Bulletins (reaching nearly 1,400 links). In 2023, it captured only  $\approx 60\%$  of the official volume.

This suggests that in 2023, Telegram functioned more as a repost source (from official attributions) rather than a primary source (where actors claim responsibility directly). The Hacktivist Boom and the migration of Ransomware PR sites to Telegram, which drove the 2025 numbers, were not yet systemic in 2023.

**C. The "Attribution Void" in Technical Communities:** The analysis of Discord (15) and Forums (59) highlights a massive "Strategic Blind Spot" in unstructured communities. With negligible attribution counts, these sources are confirmed to be strictly tactical (not-strategic). A researcher monitoring Discord in 2023 would find code snippets, exploit proofs-of-concept, and tools (as seen in Section 5.3.5), but would find effectively zero information regarding *who* was using them. The users in these communities operate with an "Identity-Agnostic" mindset, focusing on the efficacy of the exploit rather than the geopolitics of the campaign.



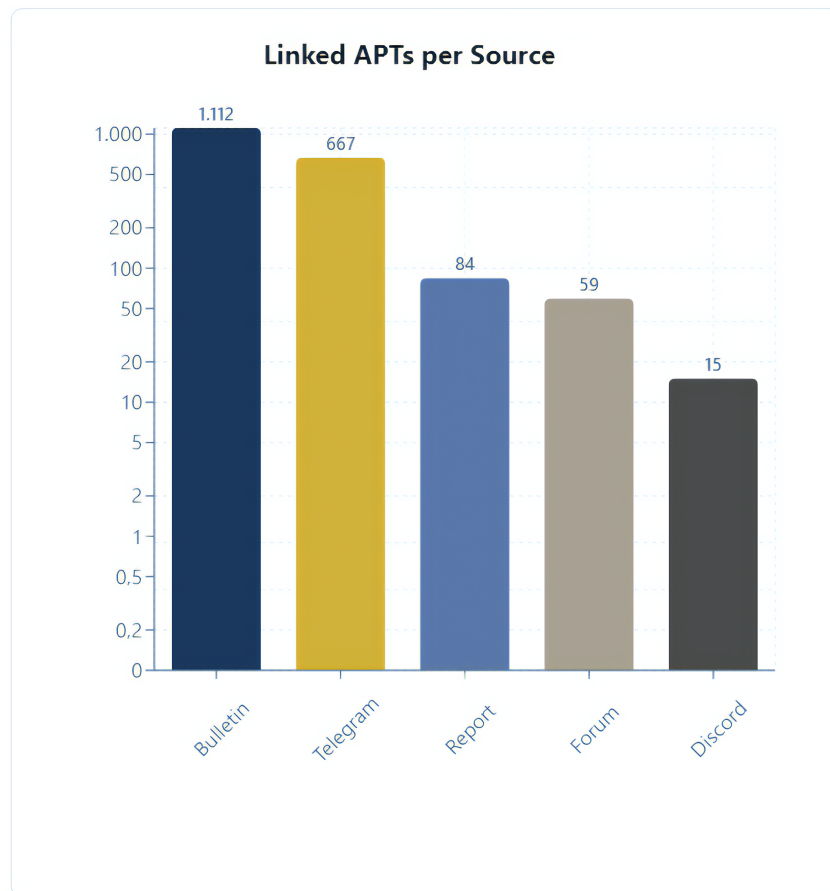


Figure 5.36: Linked APTs per Source (2023). Dominance of Bulletins (1,112), illustrating a traditional model where attribution is an institutional function. Communities like Discord (15) show zero strategic visibility.

### 5.3.7. Case Study: The MOVEit Transfer Campaign (CVE-2023-34362)

To empirically validate the "Celerity" and "Coverage" metrics discussed in previous sections, we performed a vertical analysis on the **MOVEit Transfer** supply chain attack. This event, attributed to the **Cl0p (Lace Tempest)** ransomware cartel, represents a paradigm shift in cybercrime, transitioning from encryption-based attacks to pure "Zero-Day Extortion".

The 2020–2026 analysis highlights a massive temporal gap between the underground availability of the exploit and the institutional reaction.

## Chronological Reconstruction - The "Time-to-Insight" Gap

By correlating timestamps across different source categories, we were able to reconstruct the lifecycle of the threat. The collected data reveals a critical "Pre-Disclosure Window" where the vulnerability was known and traded in underground communities before vendor acknowledgment.

**A. The "Proof of Concept" (April 2023):** The platform detected a critical signal dated April 24, 2023 (Forum category). The post, titled "POC for CVE-2023-34362 affecting MOVEit Transfer - SQL Injection", explicitly references the SQL Injection vector and the specific target software.

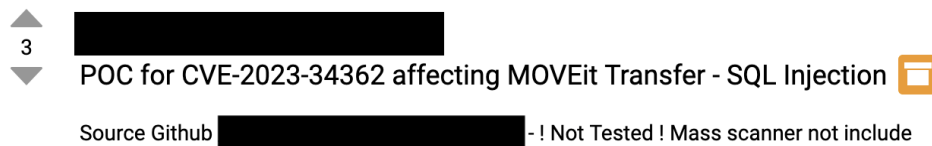


Figure 5.37: Dread post sharing a PoC for MOVEit Transfer.

This detection occurred 37 days before the first Bulletin was issued. This implies that a functional Proof-of-Concept (PoC) was circulating in the wild more than a month before the mass exploitation began.

**B. The Mass Exploitation Phase (May 2023):** Recorded around May 27, 2023. Intelligence feeds and retrospective reports identify this date as the start of data theft operations.

Following the exploitation weekend, Telegram channels began populating with victim names (e.g., British Airways, BBC, Shell), functioning as a real-time "Leak Ticker" before official confirmations.

**C. The Institutional Reaction (June 2023):** The first Bulletins appear in the dataset on June 2, 2023, with CISA adding the vulnerability to its Known Exploited Vulnerabilities catalog [11].

Table 5.4 summarizes the detection delta.

| Phase                       | Date       | Source       | Event Description                          |
|-----------------------------|------------|--------------|--------------------------------------------|
| Pre-Exploitation            | 24/04/2023 | Forum        | Release of SQLi PoC for MOVEit             |
| The Beginning of the Attack | 27/05/2023 | Intelligence | Start of Mass Exploitation (Retrospective) |
| Disclosure                  | 31/05/2023 | Bulletin     | Progress Software issues advisory          |
| Official Alert              | 02/06/2023 | Bulletin     | CISA Catalog Update                        |
| Peak Leak                   | June 2023  | Telegram     | Mass posting of victim data (Shell, BBC)   |

Table 5.4: Timeline Analysis of the MOVEit Campaign. Note the 37-day gap between the Forum PoC and the CISA Alert.

Cl0p Ransomware Group Evolution (2023-2026)

Thanks to the platform, we analyzed the evolution of the threat actor Cl0p (also tracked as TA505, FIN11, Lace Tempest). The analysis confirms the "Strategic Parity" hypothesis (Sec. 5.3.6), where Telegram acts as the primary channel for attribution claims.

Here it is the reconstructed chain:

- **Identity Confirmation:** While Bulletins use technical designators like "Lace Tempest" or "TA505", Telegram channels provide the explicit self-attribution. The group actively used Telegram to claim responsibility for high-profile breaches, including the US Department of Energy and Shell.
- **Persistence (2024-2026):** The longitudinal data proves that the MOVEit campaign was not an isolated event but a TTP (Tactics, Techniques, and Procedures) shift. The dataset records Cl0p exploiting new Zero-Days in Gladinet CentreStack (CVE-2025-11371) in December 2025. By January 2026, Cl0p is linked to attacks on Korean Air using yet another zero-day (CVE-2025-61882 - 2026 Escalation).

This recurrence pattern (approximately one major zero-day campaign per year) suggests that Cl0p has increasingly “industrialized” a zero-day brokerage model, shifting from opportunistic encryption toward targeted exploitation of file-transfer appliances.

## Impact Assessment and Blind Spots

The coverage analysis of the MOVEit case study highlights the specific "Blind Spots" of relying only on official sources.

Official Bulletins focused heavily on the *vulnerability* (CVE-2023-34362) and the *patch*. However, they rarely listed the specific compromised entities in real-time. On the other hand, the dataset shows that Telegram and Reports provided a granular list of victims, including:

- **Critical Infrastructure:** Siemens Energy, Schneider Electric.
- **Government:** US Department of Energy, Colorado Department of Health.
- **Education:** New York City Department of Education, Harvard University.

An analyst relying only on CISA alerts would know *how* the attack happened, but not *who* was impacted.

We can also state that the dataset is able to capture the "Long-Tail" effect of the breach. While the initial vulnerability was patched in June 2023, data leaks continued well into late 2023 and 2024. For instance, the Delta Dental breach (7 million records) was disclosed in December 2023, six months after the initial exploit. This validates the necessity of long-term monitoring of Underground sources to assess the true blast radius of a supply chain attack.

## Conclusion of the Case Study

The MOVEit Transfer incident serves as empirical proof of the platform's value proposition: The Forum source provided a 37-day strategic warning (April 24 vs. May 31); Telegram confirmed the actor (Cl0p) and the victims significantly faster than official forensic reports; The 2024-2026 data indicates that this TTP (Zero-Day Exploitation of File Transfer tools) has become the standard operating procedure for the Cl0p cartel, requiring continuous monitoring of niche software vulnerabilities.

### 5.3.8. Discussion and Answers to Research Questions

The presented analysis allows for a formal synthesis of the findings. By correlating the quantitative metrics of Celerity, Coverage, and Overlap with the qualitative insights from the Case Study, this section addresses the Research Questions defined in Section 2.4. The aggregate evidence supports a redefinition of the threat intelligence lifecycle, characterizing it as a multi-layered ecosystem where the trade-off between "Richness of Content"

and "Speed" defines the functional role of each source.

### RQ1: Temporal Dynamics and "Time-to-Insight"

**Underground sources provide a measurable "Strategic Lead" over official channels, ranging from 24 hours to several months, identifying distinct phases of the threat lifecycle.**

The analysis of the "First-to-Report" metrics and the MOVEit Case Study (Section 5.3.7) confirms that the platform successfully reconstructs the threat lifecycle, validating the existence of a "Celerity Gap" through two distinct anticipation patterns:

- **The "Strategic Lead":** As evidenced in Figure 5.24 and by the detection of CVE-2024-4577 (4-month anticipation) and the MOVEit SQL Injection PoC (37-day anticipation), Forums act as the "Incubators" of threats. The temporal gap here is maximal. This validates the hypothesis that monitoring these communities allows analysts to detect the *capability development* phase (Genesis) long before the *mass exploitation* phase begins.
- **Telegram, the "Tactical Lead":** In the 2025 dataset, Telegram emerged as the second-fastest source (>6,000 First Seen events), often anticipating official bulletins by days. This reflects the "zero-day" window in which exploitation is active before a patch is broadly deployed. Telegram functions as a "broadcast" layer, bridging the gap between early underground discovery and wider public awareness.

Overall, the system proved capable of tracing the full propagation path: from the *proof-of-concept* on forums, to *active exploitation and leaks* on Telegram, and finally to *formal remediation* in bulletins. While official bulletins remain the source of "certainty" (authoritative patches), unstructured sources provide the "time" needed for proactive hardening.

### RQ2: Source Topology and Functional Specialization

**The threat landscape is not hierarchical but "polycentric", with strict functional specialization, where emerging platforms have evolved from passive echo chambers into primary intelligence generators.**

The comparative analysis of Exclusivity (Section 5.3.2) and Coverage (Section 5.3.5) refutes the existence of a single "best" source. Instead, the data reveals a topology composed of distinct functional nodes:

- **The Evolution of Telegram (From Echo to Origin):** Addressing the spe-

cific query regarding Telegram’s role, the longitudinal data documents a structural shift. While in 2023 it functioned primarily as a secondary dissemination channel (Echo Chamber), by 2025 it surpassed official Bulletins in *Exclusivity* (29k vs 26k unique entities). This confirms its evolution into a **Primary Source**, serving as the "Ground Zero" for hacktivist claims and real-time leak disclosure.

- **The "Blind Spot" Coverage (DLS and Intelligence):** The analysis confirms that distinct source typologies cover largely non-overlapping strategic areas.
  - **DLS:** Despite low volumes, Data Leak Sites maintain a  $\approx 33\%$  uniqueness rate (Section 5.3.2). They provide the only visibility into *actor .onion domains*, a data class that is typically absent from official bulletins.
  - **Intelligence Feeds (Sensors):** These sources are highly specialized in *malware* artifacts but largely blind to *vulnerabilities* and *tools*. They act as “binary sensors” (high volume, low context) rather than strategic informants.
- **The "Weaponization" Hubs:** As highlighted in the Coverage analysis, unstructured communities (Discord and Forums) exhibit a disproportionate focus on the TOOL category. They do not compete with Bulletins on "Vulnerability Definitions" (CVEs), but strictly cover the "Operational Implementation" (Exploits and Scripts), filling the gap between a theoretical vulnerability and an active attack.

The findings demonstrate that Source Overlap is high only along the "Central Axis" (Telegram-Bulletin), while other sources (DLS, Feeds) remain isolated with each other. Consequently, a comprehensive CTI posture cannot be achieved by relying on a single dominant source type, but requires a **Hybrid Ingestion Strategy** to cover the full spectrum of Actor, Victim, Tool, and Vulnerability.

### RQ3: Strategic Attribution and Semantic Linking

The automated pipeline successfully bridged the gap between raw data and strategic intelligence, validating a "Strategic Parity" between official and underground sources while exposing the "Contextual Blindness" of high-volume feeds.

The evaluation of the Attribution Capabilities (Section 5.3.6) and Semantic Overlap (Section 5.3.3) answers the third research question by establishing three key findings regarding the automated generation of strategic intelligence:

- **Validation of "Strategic Parity":** The most relevant result is the near-equivalence in attribution volume observed in 2025 between Bulletins (1,630 Linked APTs) and

Telegram (1,386 Linked APTs). This empirically proves that unstructured underground sources have evolved to provide the same level of strategic depth as official advisories. Both channels demonstrate a comparable capability to contextually link Threat Actors to their specific Tactics, Techniques, and Procedures (TTPs), effectively bridging the gap between raw indicators and high-level attribution logic without relying only on institutional reporting.

- **The "Big Data" Paradox:** The analysis definitively resolves the question regarding the value of structured feeds. Despite generating the highest absolute volume (>880k entities), the Intelligence category contributed negligible attribution data (< 100 links). This answers the RQ by demonstrating that "More Data" does not equal "Better Intelligence." Automated feeds are efficient for *Blocking* (technical indicators) but structurally incapable of *Understanding* (strategic context), necessitating the integration of NLP-driven analysis of unstructured sources.
- **Semantic Consistency:** The high Semantic Overlap (Jaccard 3.55% on APT-TTP links) between the Bulletin-Telegram pair confirms the validity of the extraction logic. The fact that these two diverse sources independently corroborate over 1,000 specific Actor-Technique relationships validates the system's ability to transform noisy text into a coherent, verifiable strategic knowledge graph.

#### RQ4: Longitudinal Analysis

**The threat landscape exhibits a "High-Velocity Mutation" where underground channels evolve from passive receivers to primary sources within 24 months, while technical infrastructure suffers from rapid decay, making real-time ingestion a strict requirement.**

The multi-year analysis from 2023 to 2025 identified structural shifts that reshape the stability of the intelligence ecosystem. The most significant finding concerns the maturation of Telegram. In the 2023 dataset, this channel operated mainly as a secondary "echo chamber", with an exclusivity count roughly half that of official sources. By 2025, however, the picture had reversed: Telegram surpassed official bulletins in unique entity generation (29,825 vs 26,175) and roughly tripled its semantic linking (APT-TTP) capability. Taken together, these results support its evolution from a passive distribution vector to a primary source of strategic intelligence, narrowing the traditional institutional advantage in attribution.

Simultaneously, the data exposes the critical volatility of underground infrastructure. Comparing the peak volume of Ransomware Leak Sites (DLS) data recorded during mon-



itoring of 2024 posts, against the retrospective low of the 2023 dataset, let us quantify the survivorship bias inherent in Dark Web forensics: we cannot easily reconstruct data from that period, unless we consider to evaluate structured feed that collected that data at time. This drastic decay proves that threat data on Tor-based infrastructure has a short "Time-to-Live"; without real-time stateful ingestion, this intelligence becomes historically unrecoverable due to domain seizures and abandonment.

In contrast to this volatility, the Bulletin category demonstrated remarkable stability, maintaining a constant volume of approximately 26,000 exclusive entities across all three analyzed years. This consistency establishes institutional advisories as the archival backbone of the ecosystem: while underground sources provide necessary speed and anticipation, official vendors provide the data permanence required for long-term historical analysis. Ultimately, the ecosystem is defined by mutability, where the center of gravity for intelligence generation has visibly migrated from web-based Forums to instant messaging platforms, necessitating a platform architecture capable of adapting to the rapid migration of threat actors.

Simultaneously, the longitudinal data highlights the functional marginalization of Discord within the broader threat-intelligence topology. In the 2023 dataset, Discord acted as a "tactical incubator", closely coupled with legacy forums and able to anticipate official vulnerability disclosures (e.g., the CVE-2023-41991 cluster). As the ecosystem moved toward 2025 and Telegram consolidated its role as a central hub for both technical leaks and strategic attribution, Discord was pushed to the periphery. It shifted from an integrated early-warning component to an isolated, highly specialized repository focused almost exclusively on exchanging offensive capabilities (TOOLS and exploit scripts), with limited relevance for high-level attribution or synchronization with official bulletins.

Moreover, the analysis quantifies the "digital decay" of technical indicators: comparing the active 2025 structured feed volume (over 500,000 unique entities) with the 2023 historical residue (only 7,000 remaining entities) shows a 98% signal loss. This confirms that while strategic narratives (bulletins) persist, atomic indicators (IPs, hashes) are inherently short-lived, requiring high-frequency, real-time ingestion to retain value before it fades. In the end, the Argus platform demonstrates that automated attribution is feasible, provided that the ingestion strategy prioritizes *Semantic Richness* (Telegram and Bulletins) over *Raw Volume* (Structured Feeds). The system successfully evolved from a simple "Indicator Collector" to a "Strategic Correlation Engine," capable of answering the critical questions of "Who" and "How" without exclusive reliance on manual reporting.



## 6 | Conclusions

This thesis presented the design, implementation, and empirical validation of Argus, a distributed platform for the automated collection and semantic enrichment of Cyber Threat Intelligence across heterogeneous sources. The work started from two structural problems that limit CTI in practice: the fragmentation of intelligence across incompatible ecosystems, and the low signal-to-noise ratio of unstructured underground data. By engineering a unified ingestion strategy that spans structured OSINT feeds, institutional bulletins, vendor reporting, and unstructured communities on the surface and dark web, Argus demonstrates that consolidation is not only an architectural improvement but a prerequisite to perform rigorous, longitudinal comparisons of source value. The experimental campaign, conducted over a multi-year dataset spanning 2023 to 2025, shows that an automated pipeline can transform high-volume raw data into structured knowledge that supports both tactical triage and strategic reasoning.

From an engineering perspective, the primary contribution is the realization of an end-to-end system that is resilient to the constraints of adversarial collection. Argus concretize a distributed ETL approach in which acquisition and analysis are decoupled, enabling continuous ingestion even when the downstream AI pipeline is the bottleneck. This design choice is critical in high-latency and high-failure environments such as Tor-based infrastructure, where the cost of missing ephemeral content is often higher than the cost of delayed processing. The system therefore provides a practical blueprint for building CTI platforms that remain observable, extensible, and scalable under realistic conditions.

From a methodological perspective, the thesis validates that accurate CTI extraction benefits from a hybrid strategy rather than relying on a single paradigm. The results confirm that deterministic extraction remains essential for rigid indicators, while Transformer-based models are necessary to capture contextual entities such as threat actors and malware families. By combining these layers with a relevance filtering stage, the pipeline becomes robust to both tokenization failures typical of purely neural extraction and to the overwhelming noise that characterizes underground communities. This hybrid approach is not presented as an abstract design preference, but as an empirically justified

compromise between recall, precision, and functional cost.

The most significant contribution is the quantitative characterization of how CTI value is distributed across the modern source ecosystem. The comparative analysis shows that the landscape is not hierarchical, but polycentric and functionally specialized. Specifically, while Intelligence feeds dominate the raw volume with over 880,000 extracted entities, their contribute to strategic context is negligible (<100 attributed APTs). In contrast, the results indicate a convergence between official and underground sources on high-level attribution: Telegram has evolved to reach strategic parity with Official Vendor Bulletins, linking 1,386 distinct Threat Actors compared to the 1,630 identified by vendors. This shift has direct implications: a CTI posture built only on traditional structured feeds is efficient for detection and blocking, but remains strategically blind without complementary monitoring of semantically rich unstructured channels.

Finally, the timeliness analysis highlights the time-to-insight advantage provided by unstructured communities, which is measurable and highly meaningful. While Bulletins remain the primary disclosure channel ( $\approx 9,600$  first reports), Telegram secured the second rank with over 6,300 anticipation events, significantly outperforming traditional reports. This translates, from a qualitatively perspective, into critical lead times: the evaluation of the MOVEit campaign demonstrated a 37-day window between the proof-of-concept on Forums and the official CISA alert. This does not reduce the value of official announcements, which remain the source of authoritative remediation, but it changes the optimal workflow: certainty often arrives late, while early warnings arrive elsewhere. The thesis therefore supports the conclusion that proactive defense requires monitoring of both of them, because time and authority are no longer co-located in a single source category.

In the end, the longitudinal analysis (2023-2025) demonstrates that source relevance is not a static property but evolves functionally over time. The ecosystem shows high fluidity: Telegram transitioned from a secondary "echo chamber" in 2023 to a primary strategic source in 2025, effectively tripling its attribution capabilities to rival official vendors. Conversely, infrastructure like Data Leak Sites exhibits rapid decay, confirming that intelligence value is often ephemeral. Consequently, a robust CTI platform cannot rely on static ingestion logic, it requires continuous, stateful monitoring to adapt to these structural shifts, ensuring that the system evolves alongside the threat actors it tracks. In this sense, Argus is not only a retrospective measurement tool: it can be maintained as a living observatory to continuously track how the relevance of each source category shifts over time and to early detect when a channel becomes strategically important. At the same time, this also highlights a natural limitation of any fixed source portfolio: new ecosystems and communities will inevitably emerge, meaning that long-term effectiveness

depends on the platform's ability to onboard additional sources and evolve its ingestion strategy accordingly.



## 7 | Limitation and Future Work

### 7.1. Limitations

While the results confirm the effectiveness of Argus as both an operational platform and a research instrument, it is important to state several limitations to correctly interpret the scope of the findings and the area to which the conclusions can be generalized.

The first concerns the availability of historical data, especially in the dark web context. Building a reliable “historical” view is inherently difficult because many sources remove content, change structure, or disappear entirely: sites and onion services can be taken down, relocated, or shut down, and complete, verifiable archives are rarely available. As a consequence, retrospective coverage cannot be ensured with the same completeness as continuous monitoring, and some past information may simply no longer be retrievable.

A second limitation concerns source accessibility and the ethical constraints that follow from it. A significant portion of underground forums and communities impose entry barriers such as payments, invite-only access, vouching procedures, or other actions required to gain admission. In many cases, these mechanisms make access not only technically challenging, but also ethically and legally problematic, as they may imply financially supporting criminal ecosystems.

A third limitation relates to the variety of the source set. Although the number of integrated sources is already substantial and sufficient to reveal structural differences across categories (feeds, bulletins, social channels, forums, DLS), the CTI ecosystem is highly fragmented and continuously evolving. To further maximize coverage and generalizability, the platform would benefit from expanding the number of monitored sources, including emerging communities, additional vendors, and a broader range of regional and multi-lingual channels, thereby reducing the risk that some results reflect peculiarities of the observed sample.

A fourth limitation concerns the dependence on labeled data for supervised components, in particular the relevance filter. Even with a robust pipeline, the ability to separate truly

informative content from noise is bounded by the quality and coverage of the curated hand-labeled dataset. As new threat categories emerge, communication patterns shift, and terminology evolves, labels must be updated to keep the dataset aligned with the domain. This introduces a maintenance cost that grows with source heterogeneity and the speed at which narratives change.

A fifth limitation concerns the error propagation relative to the automated extraction. The quantitative analyses presented in this thesis, such as source exclusivity and overlap, are downstream derivatives of the Named Entity Recognition (NER) pipeline. While the hybrid architecture minimizes errors, no extraction system achieves perfect precision or recall. Consequently to this, minor extraction artifacts (e.g., false positives in dictionary-based matching or missed entities in noisy text) may propagate into the aggregate statistics, introducing a potential margin of error in the fine-grained counts.

A sixth class of limitations is related to semantic blind spots intrinsic to the current modality assumptions. The pipeline is text-centric and therefore cannot fully capture intelligence exchanged through images, screenshots, and other media, which are common in chat-based underground communities. Similarly, linguistic drift remains a persistent challenge: cybercrime slang and multilingual mixing evolve quickly, and even strong multilingual backbones can struggle with ambiguous terminology and rapidly changing jargon without periodic domain adaptation.

The last class of limitations concerns the temporal validity of the findings. As we have stated in the longitudinal analysis (RQ4, Section 5.3.8), the threat landscape is characterized by high-velocity mutation: sources that act as primary intelligence generators today (e.g., Telegram) may revert to secondary roles or be replaced by new platforms in the future. Therefore, the functional hierarchy and topology established in this study represent a snapshot of the ecosystem observed between 2023 and 2026. At the same time, this also highlights a natural limitation of any fixed source portfolio: new ecosystems and communities will inevitably emerge or change, which means that long-term effectiveness depends on the platform's ability to take on additional sources and evolve its ingestion strategy accordingly.

## 7.2. Future Work

These limitations naturally translate into a concrete roadmap for both research and engineering. One clear, high-impact direction is extending Argus beyond plain text to support multimodal extraction and full attachment handling. In real-world underground ecosystems, a lot of the most useful evidence is not written directly in posts, but comes

as screenshots, PDFs, documents, archives, or other files shared in chats and forums. These attachments often include concrete “proof” (e.g., leak lists, configuration files, logs, indicators, operational notes) that would otherwise be missed. At the same time, this step should be approached carefully: attachments can easily contain sensitive, illegal, or harmful material. For that reason, any attachment pipeline should be built with ethical and legal constraints in mind from the start, defining clear rules on what is allowed to be collected. A practical extension would be a dedicated attachment-handling module that downloads files safely, classifies them, applies OCR to images, and performs text extraction from PDFs and common document formats when appropriate. The extracted text can then be fed back into the existing relevance filtering and NER stages, keeping the architecture modular while significantly expanding the system’s coverage.

The second direction focuses on expanding and diversifying the set of monitored sources. Although the current number of integrated sources is already substantial and sufficient for meaningful comparative analyses, the CTI ecosystem remains highly fragmented and continuously evolving. To increase coverage and improve generalizability, Argus should be extended to include additional communities, vendors, and regional or multilingual channels. In particular, systematically integrating high-velocity social media sources such as “X” (formerly “Twitter”) would allow the platform to capture early signals, campaign traces, and near-real-time discussions that often precede formal reporting.

The third direction concerns the long-term maintenance of supervised components, especially the relevance filter, in a pragmatic way. Rather than expanding the labeled dataset indiscriminately, the platform could prioritize what to label by focusing analyst effort on the most informative or ambiguous samples, improving performance with less manual work. In parallel, periodic model and normalization updates would help keep extraction aligned with evolving slang, new actor names, and rapidly changing narratives, reducing drift over time.

The fourth direction is to leverage Argus outputs as a structured knowledge base for CTI-focused question answering and reasoning. Entities, relationships, timelines, and source references can populate a curated knowledge base (e.g., an indexed repository or a knowledge graph) that supports a CTI-specialized LLM. The model could then answer analyst questions such as attribution hypotheses, cross-source correlations, and temporal comparisons, while remaining grounded in verified evidence. A key requirement is enforcing grounding and traceability: responses should include explicit references to support claims with sources from the knowledge base, minimizing hallucinations and making the system more suitable for operational use.

In conclusion, the effectiveness of modern CTI does not depend only on collecting as much data as possible, but on combining diverse and complementary sources. Integrating high-volume feeds, official reporting, and unstructured high-velocity channels reduces blind spots and enables a more complete and timely view of emerging threats. Argus supports this integration by transforming fragmented information into structured and traceable knowledge, instantly actionable to efficiently understand threat patterns and monitored source quality.



## Bibliography

- [1] V. Adewopo and B. Gonen. Scraping the deep web: A 3-dimensional framework for cyber-threat intelligence. Paper presented at IT Expo 2020, University of Cincinnati, 2020. URL <https://itexpo.live/2020/research/13>. Accessed: 2024-02-03.
- [2] M. Allegretta, G. Siracusano, R. Gonzalez, and M. Gramaglia. Are crowd-sourced CTI datasets ready for supporting anti-cybercrime intelligence? *Computer Networks*, 234:109920, 2023. doi: 10.1016/j.comnet.2023.109920. URL <https://doi.org/10.1016/j.comnet.2023.109920>.
- [3] D. R. Arikkat, S. B. T., S. Nicolazzo, A. Nocera, V. P., R. R. K. A., and K. R. Cti dataset construction from telegram, 2025. URL <https://arxiv.org/abs/2509.20943>.
- [4] G. Ayoade, S. Chandra, L. Khan, K. Hamlen, and B. Thuraisingham. Automated threat report classification over multi-source data. In *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pages 236–245, 2018. doi: 10.1109/CIC.2018.00040.
- [5] J. Baek, J. O, S. Jeong, and W. Kim. A study on improving the automatic classification performance of cybersecurity mitre att&ck tactics using nlp-based modernbert and bertopic models. *Electronics*, 14(22), 2025. ISSN 2079-9292. doi: 10.3390/electronics14224434. URL <https://www.mdpi.com/2079-9292/14/22/4434>.
- [6] M. Bailey, D. Dittrich, E. Kenneally, and D. Maughan. The menlo report. *IEEE Security & Privacy*, 10(2):71–75, 2012. doi: 10.1109/MSP.2012.52.
- [7] S. J. Bell and P. Komisarczuk. An analysis of phishing blacklists: Google safe browsing, openphish, and phishtank. In *Australasian Computer Science Week 2020 (ACSW '20)*. ACM, 2 2020. doi: 10.1145/3373017.3373020.
- [8] D. Bianco. The pyramid of pain, 2014. URL <http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html>. Conceptual model for CTI indicators.
- [9] N. A. Bondareva, A. E. Bondarev, S. V. Andreev, and I. G. Ryzhova. Informa-

- tion density of objects in digital environment: Theoretical foundations. *Scientific Visualization*, 17(4):87–98, 2025.
- [10] M. Burrows. The chubby lock service for loosely-coupled distributed systems. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI)*, pages 335–350. USENIX, 2006. URL <https://research.google/pubs/pub27897/>.
- [11] CISA. Alert (aa23-158a): Cl0p ransomware gang exploits cve-2023-34362 moveit transfer, 06 2023. URL <https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-158a>. Cybersecurity Advisory.
- [12] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale, 2020. URL <https://arxiv.org/abs/1911.02116>.
- [13] D. Crankshaw, X. Wang, G. Zhou, M. J. Franklin, J. E. Gonzalez, and I. Stoica. Clipper: A low-latency online prediction serving system, 2017. URL <https://arxiv.org/abs/1612.03079>.
- [14] Cybersecurity and Infrastructure Security Agency (CISA). Binding operational directive 22-01: Reducing the significant risk of known exploited vulnerabilities. U.S. Government directive, 11 2021. URL <https://www.cisa.gov/binding-operational-directive-22-01>. Defines operational requirements around the KEV catalog.
- [15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL <https://arxiv.org/abs/1810.04805>.
- [16] J. Ding, W. Peng, Y. Lu, Z. Wang, and Y. Feng. Reto: Reinforcement learning enhanced terminology optimization for cyber threat intelligence summarization. *Neurocomputing*, 654:131339, 2025. ISSN 0925-2312. doi: <https://doi.org/10.1016/j.neucom.2025.131339>. URL <https://www.sciencedirect.com/science/article/pii/S0925231225020119>.
- [17] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina. Microservices: yesterday, today, and tomorrow, 2017. URL <https://arxiv.org/abs/1606.04036>.
- [18] R. A. Ellinitakis, K. Fysarakis, P. Bountakas, and G. Spanoudakis. Uncovering hidden threats: Automated, machine learning-based discovery & extraction of cyber

- threat intelligence from online sources. In *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2024. doi: 10.1109/CSR61664.2024.10679473.
- [19] European Union Agency for Cybersecurity (ENISA). Enisa threat landscape 2023. Technical report, ENISA, 10 2023. URL <https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023>. Annual report on the state of the cybersecurity threat landscape.
- [20] P. Evangelatos, C. Iliou, T. Mavropoulos, K. Apostolou, T. Tsikrika, S. Vrochidis, and I. Kompatsiaris. Named entity recognition in cyber threat intelligence using transformer-based models. In *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 348–353, 2021. doi: 10.1109/CSR51186.2021.9527981.
- [21] Á. Feal, P. Vallina, J. Gamba, S. Pastrana, A. Nappa, O. Hohlfeld, N. Vallina-Rodriguez, and J. Tapiador. Blocklist babel: On the transparency and dynamics of open source blocklisting. *IEEE Transactions on Network and Service Management*, 18(2):1334–1349, 2021. doi: 10.1109/TNSM.2021.3075552.
- [22] Filigran. Opencti: Open cyber threat intelligence platform, 2024. URL <https://github.com/OpenCTI-Platform/opencti>. Knowledge graph-based CTI platform strictly adhering to STIX 2.1.
- [23] W. E. Forum. Global cybersecurity outlook 2025. Technical report, World Economic Forum, 2025. URL <https://www.weforum.org/publications/global-cybersecurity-outlook-2025/>.
- [24] J. Gharibshah, T. C. Li, A. Castro, K. Pelechrinis, E. E. Papalexakis, and M. Faloutsos. Mining actionable information from security forums: the case of malicious ip addresses, 2018. URL <https://arxiv.org/abs/1804.04800>.
- [25] G. Hohpe and B. Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, Boston, MA, 2003. ISBN 978-0321200686.
- [26] S. Hussain and T. Stachelhaus. Towards reproducible ransomware analysis. In *Proceedings of the Workshop on Cyber Security Experimentation and Test (CSET)*. ACM, 2023. doi: 10.1145/3607505.3607510. URL <https://dl.acm.org/doi/10.1145/3607505.3607510>.
- [27] M. Husák, M. Čermák, T. Jirsík, and P. Čeleda. Https traffic analysis and client identification using passive ssl/tls fingerprinting. *EURASIP Journal on Information Security*, 2016(1):1–14, 2016. doi: 10.1186/s13635-016-0030-7.

- [28] T. C. T. Intelligence. Discord and telegram: Democratization of cybercrime. Technical report, Thales Group, 2024. URL <https://cds.thalesgroup.com/en/hot-topics/discord-and-telegram-democratization-cybercrime>.
- [29] ISACA. Building a threat-led cybersecurity program. *ISACA White Papers*, 2025. URL <https://www.isaca.org/resources/white-papers/2025/building-a-threat-led-cybersecurity-program>.
- [30] P. Jaccard. Etude de la distribution florale dans une portion des alpes et du jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 37:547–579, 01 1901. doi: 10.5169/seals-266450.
- [31] V. Jain, S. Alam, S. Krishnamurthy, and M. Faloutsos. C2store: C2 server profiles at your fingertips. *Proceedings of the ACM on Networking*, 1:1–21, 11 2023. doi: 10.1145/3629132.
- [32] M. Janiszewski, M. Rytel, P. Lewandowski, and H. Romanowski. Creating vulnerabilities and exploits database of iot devices. In *Proceedings of the 2022 European Interdisciplinary Cybersecurity Conference, EICC '22*, page 91–92, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450396035. doi: 10.1145/3528580.3532990. URL <https://doi.org/10.1145/3528580.3532990>.
- [33] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, Mar. 2023. ISSN 1557-7341. doi: 10.1145/3571730. URL <http://dx.doi.org/10.1145/3571730>.
- [34] Y. Jin, E. Jang, J. Cui, J.-W. Chung, Y. Lee, and S. Shin. Darkbert: A language model for the dark side of the internet, 2023. URL <https://arxiv.org/abs/2305.08596>.
- [35] E. Jollès, S. Gillard, D. Percia David, M. Strohmeier, and A. Mermoud. Empirical evidence of the collective intelligence sharing dynamics on ThreatFox. In *Critical Information Infrastructures Security (CRITIS 2022), Revised Selected Papers*, Lecture Notes in Computer Science. Springer, 2023. doi: 10.1007/978-3-031-35190-7\_10. URL [https://link.springer.com/chapter/10.1007/978-3-031-35190-7\\_10](https://link.springer.com/chapter/10.1007/978-3-031-35190-7_10).
- [36] M. Khder. Web scraping or web crawling: State of art, techniques, approaches and application. *International Journal of Advances in Soft Computing and its Applications*, 13:145–168, 12 2021. doi: 10.15849/IJASCA.211128.11.
- [37] M. Kleppmann. *Designing Data-Intensive Applications: The Big Ideas Behind Re-*

- liable, Scalable, and Maintainable Systems*. O'Reilly Media, Sebastopol, CA, 2017. ISBN 978-1449373320.
- [38] D. Kouzis-Loukas. *Learning Scrapy*. Packt Publishing Ltd, 2016.
- [39] B. D. Le, G. Wang, M. Nasim, and A. Babar. Gathering cyber threat intelligence from twitter using novelty classification, 2019.
- [40] V. G. Li, M. Dunn, P. Pearce, D. McCoy, G. M. Voelker, S. Savage, and K. Levchenko. Reading the tea leaves: a comparative analysis of threat intelligence. In *Proceedings of the 28th USENIX Conference on Security Symposium*, SEC'19, page 851–867, USA, 2019. USENIX Association. ISBN 9781939133069.
- [41] X. Liao, K. Yuan, X. Wang, Z. Li, L. Xing, and R. Beyah. Acing the ioc game: Toward automatic discovery and analysis of open-source cyber threat intelligence. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 755–766, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394. doi: 10.1145/2976749.2978315. URL <https://doi.org/10.1145/2976749.2978315>.
- [42] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL <https://arxiv.org/abs/1907.11692>.
- [43] L. Maschmeyer. Raising a red flag: Evidence from western cyber conflict reporting. *Journal of Strategic Studies*, 2021. URL <https://www.tandfonline.com/doi/full/10.1080/01402390.2021.1907542>.
- [44] S. Massengale and P. Huff. Assessing and prioritizing ransomware risk based on historical victim data, 2025. URL <https://arxiv.org/abs/2502.04421>.
- [45] R. McMillan. Definition of threat intelligence. Research Note G00262483, Gartner, 2014.
- [46] Meta AI. Introducing meta llama 3: The most capable openly available llm to date, 2024. URL <https://ai.meta.com/blog/meta-llama-3/>. Model Card and Technical Release.
- [47] T. Mimoto, K. Kita, Y. Gempei, T. Isohara, S. Kiyomoto, and T. Tanaka. Cyber threat intelligence report summarization with&nbsp;named entity recognition. In *Advances in Information and Computer Security: 20th International Workshop on Security, IWSEC 2025, Fukuoka, Japan, November 25–27, 2025, Proceedings*, page 468–485, Berlin, Heidelberg, 2025. Springer-Verlag. ISBN 978-981-95-

- 4673-2. doi: 10.1007/978-981-95-4674-9\_24. URL [https://doi.org/10.1007/978-981-95-4674-9\\_24](https://doi.org/10.1007/978-981-95-4674-9_24).
- [48] I. Mouiche and S. Saad. Ti-nermergerv2: Automating the integration of threat intelligence ner datasets via stix standard. *Computers & Security*, 163:104838, 2026. ISSN 0167-4048. doi: <https://doi.org/10.1016/j.cose.2026.104838>. URL <https://www.sciencedirect.com/science/article/pii/S0167404826000143>.
- [49] E. Nunes, A. Diab, A. Gunn, E. Marin, V. Mishra, V. Paliath, J. Robertson, J. Shakarian, A. Thart, and P. Shakarian. Darknet and deepnet mining for proactive cybersecurity threat intelligence. In *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, pages 7–12, 2016. doi: 10.1109/ISI.2016.7745435.
- [50] OASIS CTI Technical Committee. Stix™ version 2.1. Technical report, OASIS Standard, 2021. URL <https://docs.oasis-open.org/cti/stix/v2.1/stix-v2.1.html>.
- [51] T. Paladini, L. Ferro, M. Polino, S. Zanero, and M. Carminati. You might have known it earlier: Analyzing the role of underground forums in threat intelligence. In *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, Padua, Italy, 2024. ACM. doi: 10.1145/3678890.3678930.
- [52] P. Panagiotou, C. Iliou, K. Apostolou, T. Tsikrika, S. Vrochidis, P. Chatzimisios, and I. Kompatsiaris. Towards selecting informative content for cyber threat intelligence. In *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 354–359, 2021. doi: 10.1109/CSR51186.2021.9527909.
- [53] S. Pastrana, D. R. Thomas, A. Hutchings, and R. Clayton. Crimebb: Enabling cybercrime research on underground forums at scale. In *Proceedings of the 2018 World Wide Web Conference, WWW '18*, page 1845–1854, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee. ISBN 9781450356398. doi: 10.1145/3178876.3186178. URL <https://doi.org/10.1145/3178876.3186178>.
- [54] W. Peng, J. Ding, W. Wang, L. Cui, W. Cai, Z. Hao, and X. Yun. Ctisum: A new benchmark dataset for cyber threat intelligence summarization, 2025. URL <https://arxiv.org/abs/2408.06576>.
- [55] P. Phandi, A. Silva, and W. Lu. SemEval-2018 task 8: Semantic extraction from CybersecUurity REports using natural language processing (SecureNLP). In M. Apidianaki, S. M. Mohammad, J. May, E. Shutova, S. Bethard, and M. Carpuat, editors, *Proceedings of the 12th International Workshop on Semantic Evaluation*, pages 697–

- 706, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/S18-1113. URL <https://aclanthology.org/S18-1113/>.
- [56] A. Pinto and A. Maxwell. Measuring the effectiveness of threat intelligence feeds. In *Defcon 22*, 2014.
- [57] A. Ramsdale, S. Shiaeles, and N. Kolokotronis. A comparative analysis of cyber-threat intelligence sources, formats and languages. *Electronics*, 9(5), 2020. ISSN 2079-9292. doi: 10.3390/electronics9050824. URL <https://www.mdpi.com/2079-9292/9/5/824>.
- [58] L. Ratnov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, page 147–155, USA, 2009. Association for Computational Linguistics. ISBN 9781932432299.
- [59] Reactive Streams Initiative. Reactive streams 1.0.0 release announcement. <https://www.reactive-streams.org/>, 2015. Specification and initiative homepage (accessed 2026-02-08).
- [60] S. S. Roy, E. P. Vafa, K. Khanmohammadi, and S. Nilizadeh. Darkgram: A large-scale analysis of cybercriminal activity channels on telegram, 2025. URL <https://arxiv.org/abs/2409.14596>.
- [61] C. Sabottke, O. Suci, and T. Dumitras. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting Real-World exploits. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 1041–1056, Washington, D.C., Aug. 2015. USENIX Association. ISBN 978-1-939133-11-3. URL <https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/sabottke>.
- [62] C. Sauerwein, I. Pekaric, M. Felderer, and R. Breu. An analysis and classification of public information security data sources used in research and practice. *Computers & Security*, 82:140–155, 2019. ISSN 0167-4048. doi: <https://doi.org/10.1016/j.cose.2018.12.011>. URL <https://www.sciencedirect.com/science/article/pii/S0167404818304978>.
- [63] S. L. Schröer, N. Canevascini, I. Pekaric, P. Widmer, and P. Laskov. The dark side of the web: Towards understanding various data sources in cyber threat intelligence. In *2025 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 79–89, 2025. doi: 10.1109/EuroSPW67616.2025.00015.



- [64] ScrapingAnt. Header mutation fuzzing: Discovering the minimal identity to bypass bot detection. *ScrapingAnt Blog*, 2025. URL <https://scrapingant.com/blog/header-mutation-fuzzing-discovering-the-minimal-identity-to>.
- [65] N. Shimizu and M. Hashimoto. Vulnerability management chaining: An integrated framework for efficient cybersecurity risk prioritization, 2025. URL <https://arxiv.org/abs/2506.01220>.
- [66] B. Stojkovski, G. Lenzini, V. Koenig, and S. Rivas. What’s in a cyber threat intelligence sharing platform? a mixed-methods user experience investigation of misp. In *Proceedings of the 37th Annual Computer Security Applications Conference, ACSAC ’21*, page 385–398, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450385794. doi: 10.1145/3485832.3488030. URL <https://doi.org/10.1145/3485832.3488030>.
- [67] The MITRE Corporation. Mitre att&ck®: Design and philosophy, 2023. URL <https://attack.mitre.org/>. Knowledge Base of Adversary Tactics and Techniques.
- [68] Y. Tian, X. Dai, Z. Li, H. Guo, and X. Mao. Improving the accuracy of cybersecurity spam email detection using ensemble techniques: A stacking approach. *PLoS ONE*, 20(9):e0331574, 2025. doi: 10.1371/journal.pone.0331574.
- [69] Y. Tian, Y. Yu, J. Sun, and Y. Wang. From past to present: A survey of malicious url detection techniques, datasets and code repositories, 2025. URL <https://arxiv.org/abs/2504.16449>.
- [70] W. Tounsi and H. Rais. A survey on technical threat intelligence in the age of sophisticated cyber attacks. *Computers & Security*, 72:212–233, 2018. ISSN 0167-4048. doi: <https://doi.org/10.1016/j.cose.2017.09.001>. URL <https://www.sciencedirect.com/science/article/pii/S0167404817301839>.
- [71] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody. Misp: The design and implementation of a collaborative threat intelligence sharing platform. In *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security, WISCS ’16*, page 49–56, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450345651. doi: 10.1145/2994539.2994542. URL <https://doi.org/10.1145/2994539.2994542>.
- [72] X. Wang, X. Liu, S. Ao, N. Li, Z. Jiang, Z. Xu, Z. Xiong, M. Xiong, and X. Zhang. Dnrti: A large-scale dataset for named entity recognition in threat intelligence. In *2020 IEEE 19th International Conference on Trust, Security and Pri-*



- vacy in Computing and Communications (TrustCom)*, pages 1842–1848, 2020. doi: 10.1109/TrustCom50675.2020.00252.
- [73] X. Wang, S. He, Z. Xiong, X. Wei, Z. Jiang, S. Chen, and J. Jiang. Aptner: A specific dataset for ner missions in cyber threat intelligence field. In *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 1233–1238, 2022. doi: 10.1109/CSCWD54268.2022.9776031.
- [74] D. Wolpert and W. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997. doi: 10.1109/4235.585893.



# A | Configuration of the Scheduler

This appendix documents the operational configuration of the Argus CTI Orchestrator. The orchestration follows a declarative approach: execution frequencies, target backends, and runtime safeguards are defined in `config.yaml`, while the application code only interprets and executes these definitions. At runtime, `scheduler.py` loads the YAML file and registers tasks using `APScheduler`, persisting state in a SQL-backed job store. The actual task actions are implemented in `jobs.py`, which provides two handlers: (i) triggering Scrapy spiders through the `Scrapyd` API, and (ii) sending email notifications used for Human-in-the-Loop (HITL) coordination before CAPTCHA or login-protected crawls.

## A.1. Declarative Job Definitions (`config.yaml`)

The `config.yaml` file defines Argus scheduling rules in a declarative way, keeping crawl frequencies, target Scrapy backends, and safeguards outside the scheduler code. It separates *Clean Web* jobs (high-frequency, automated scraping via the `clean` endpoint) from *Dark Web* jobs (lower-frequency, operator-assisted crawling via `tor`). The `notify_*` email tasks implement the Human-in-the-Loop mechanism by warning the analyst shortly before CAPTCHA or login-protected crawls, while additional parameters (jitter, lock groups, overlap prevention, misfire handling) improve execution robustness.

```
1 timezone: Europe/Rome
2
3 servers:
4   clean: ${SCRAPYD_CLEAN_URL}
5   tor:   ${SCRAPYD_TOR_URL}
6
7 defaults:
8   prevent_overlap: true
9   misfire_grace_seconds: 900
10
```

```
11 tasks:
12 # === CAPTCHA / LOGIN NOTIFICATIONS ===
13 - name: notify_dls_captcha
14   type: email
15   cron: "50 09 */2 * *"
16   to: "analyst@argus-cti.local"
17   subject: "[Reminder] DLS crawler starts in 10 minutes"
18   body: |
19       The DLS (Ransomware Leak Sites) crawler will start in 10 minutes.
20       Open the web VNC at http://localhost:7900 and be ready to solve
21       any CAPTCHAs.
22
23 - name: notify_wilders_captcha
24   type: email
25   cron: "50 09 */4 * *"
26   to: "analyst@argus-cti.local"
27   subject: "[Reminder] Wilders Security crawler starts in 10 minutes"
28   body: |
29       The Wilders Security Forum crawler will start in 10 minutes.
30       Open the web VNC at http://localhost:7900 and be ready to solve
31       any CAPTCHAs.
32
33 - name: notify_bhw_weekly_captcha
34   type: email
35   cron: "50 09 * * 0"
36   to: "analyst@argus-cti.local"
37   subject: "[Reminder] BlackHatWorld starts in 10 minutes"
38   body: "BlackHatWorld Social Media starts in 10 minutes. Open VNC for
39         login/captcha."
40
41 - name: notify_garage4hackers_weekly_captcha
42   type: email
43   cron: "50 10 * * 0"
44   to: "analyst@argus-cti.local"
45   subject: "[Reminder] Garage4Hackers starts in 10 minutes"
46   body: "Garage4Hackers starts in 10 minutes. Open VNC for login/
47         captcha."
48
49 - name: notify_ogusers_weekly_captcha
50   type: email
51   cron: "50 11 * * 0"
52   to: "analyst@argus-cti.local"
53   subject: "[Reminder] OGUsers starts in 10 minutes"
```

```
50     body: "OGUsers Market starts in 10 minutes. Open VNC for login/
51     captcha."
52 - name: notify_piratesforum_weekly_captcha
53   type: email
54   cron: "50 12 * * 0"
55   to: "analyst@argus-cti.local"
56   subject: "[Reminder] PiratesForum starts in 10 minutes"
57   body: "PiratesForum starts in 10 minutes. Open VNC for login/captcha
58   ."
59 - name: notify_unknowncheats_weekly_captcha
60   type: email
61   cron: "50 13 * * 0"
62   to: "analyst@argus-cti.local"
63   subject: "[Reminder] UnknownCheats starts in 10 minutes"
64   body: "UnknownCheats starts in 10 minutes. Open VNC for login/
65   captcha."
66 # === DARK WEB ===
67 - name: dark_crawler_periodic
68   type: spider
69   project: Sentinels_Dark
70   spider: Dread
71   server: tor
72   cron: "0 04 */2 * *"
73
74 # === CLEAN WEB: Telegram Crawler ===
75 - name: clean_telegram_crawler_generic
76   project: Sentinels_Light
77   spider: Telegram_Crawler
78   server: clean
79   cron: "30 06 * * *"
80   jitter: 900
81
82 # === CLEAN WEB: Forum Weekly ===
83 - name: clean_blackhatworld_weekly
84   project: Sentinels_Light
85   spider: BlackHatWorldSocialMedia
86   server: clean
87   cron: "00 10 * * 0"
88
89 - name: clean_garage4hackers_weekly
90   project: Sentinels_Light
```

```
91     spider: Garage4HackersCyberSecurity
92     server: clean
93     cron: "00 11 * * 0"
94
95 - name: clean_ogusers_weekly
96   project: Sentinels_Light
97   spider: OGUsers_OG_Twitter_Usernames
98   server: clean
99   cron: "00 12 * * 0"
100
101 - name: clean_piratesforum_weekly
102   project: Sentinels_Light
103   spider: PiratesForumSelected
104   server: clean
105   cron: "00 13 * * 0"
106
107 - name: clean_unknowncheats_weekly
108   project: Sentinels_Light
109   spider: UnknownCheats_AntiCheat_Software_Programming
110   server: clean
111   cron: "00 14 * * 0"
112
113 - name: clean_wilders_forum_periodic
114   project: Sentinels_Light
115   spider: WildersForum
116   server: clean
117   cron: "00 17 * * 0"
118
119 # === CLEAN WEB: APT Reports ===
120 - name: clean_aptcybercriminalscampaigncollection_apr
121   project: Sentinels_Light
122   spider: APTCybercriminalsCampaignCollection
123   server: clean
124   cron: "00 02 1,15 * * "
125
126 - name: clean_aptnotes_apr
127   project: Sentinels_Light
128   spider: AptNotes
129   server: clean
130   cron: "15 02 1,15 * * "
131
132 - name: clean_exploitdb_papers_apr
133   project: Sentinels_Light
134   spider: ExploitDB_Papers
```

```
135     server: clean
136     cron: "30 02 1,15 * *"
137
138 - name: clean_zeroscience_apr
139   project: Sentinels_Light
140   spider: ZeroScience
141   server: clean
142   cron: "45 02 1,15 * *"
143
144 # === CLEAN WEB: Posts & Bulletins ===
145 - name: clean_broadcom_blogs_posts
146   project: Sentinels_Light
147   spider: Broadcom_Blogs
148   server: clean
149   cron: "00 03 2,6,10,14,18,22,26,30 * *"
150
151 - name: clean_broadcom_bulletin_posts
152   project: Sentinels_Light
153   spider: Broadcom_Bulletin
154   server: clean
155   cron: "05 03 2,6,10,14,18,22,26,30 * *"
156
157 - name: clean_malwarebytes_blog_posts
158   project: Sentinels_Light
159   spider: Malwarebytes_Blog
160   server: clean
161   cron: "15 03 2,6,10,14,18,22,26,30 * *"
162
163 - name: clean_morphisec_blog_posts
164   project: Sentinels_Light
165   spider: Morphisec_blog
166   server: clean
167   cron: "20 03 2,6,10,14,18,22,26,30 * *"
168
169 - name: clean_security_com_posts
170   project: Sentinels_Light
171   spider: Security.Com
172   server: clean
173   cron: "25 03 2,6,10,14,18,22,26,30 * *"
174
175 - name: clean_securityaffairs_posts
176   project: Sentinels_Light
177   spider: SecurityAffairs
178   server: clean
```

```
179     cron: "30 03 2,6,10,14,18,22,26,30 * *"
180
181 - name: clean_talosintelligence_vulnerability_posts
182   project: Sentinels_Light
183   spider: TalosIntelligence_Vulnerability
184   server: clean
185   cron: "35 03 2,6,10,14,18,22,26,30 * *"
186
187 - name: clean_thehackernews_cyberattack_posts
188   project: Sentinels_Light
189   spider: TheHackerNews_CyberAttack
190   server: clean
191   cron: "40 03 2,6,10,14,18,22,26,30 * *"
192
193 - name: clean_thehackernews_databreach_posts
194   project: Sentinels_Light
195   spider: TheHackerNews_DataBreach
196   server: clean
197   cron: "45 03 2,6,10,14,18,22,26,30 * *"
198
199 - name: clean_thehackernews_vulnerability_posts
200   project: Sentinels_Light
201   spider: TheHackerNews_Vulnerability
202   server: clean
203   cron: "50 03 2,6,10,14,18,22,26,30 * *"
204
205 - name: clean_threatpostmalware_posts
206   project: Sentinels_Light
207   spider: ThreatPostMalware
208   server: clean
209   cron: "55 03 2,6,10,14,18,22,26,30 * *"
210
211 - name: clean_threatpostvulnerabilities_posts
212   project: Sentinels_Light
213   spider: ThreatPostVulnerabilities
214   server: clean
215   cron: "00 04 2,6,10,14,18,22,26,30 * *"
216
217 - name: clean_trendmicro_malware_posts
218   project: Sentinels_Light
219   spider: TrendMicro_Malware
220   server: clean
221   cron: "05 04 2,6,10,14,18,22,26,30 * *"
222
```



```
223 - name: clean_welivesecurity_posts
224   project: Sentinels_Light
225   spider: WeLiveSecurity
226   server: clean
227   cron: "10 04 2,6,10,14,18,22,26,30 * *"
228
229 - name: clean_welivesecurity_posts_bis
230   project: Sentinels_Light
231   spider: WeLiveSecurity
232   server: clean
233   cron: "10 04 2,6,10,14,18,22,26,30 * *"
234
235 - name: clean_zerodayinitiative_blog_posts
236   project: Sentinels_Light
237   spider: ZeroDayInitiative_Blog
238   server: clean
239   cron: "15 04 2,6,10,14,18,22,26,30 * *"
240
241 - name: clean_ahnlabasecreportlistmulticategory_posts
242   project: Sentinels_Light
243   spider: AhnlabAsecReportListMultiCategory
244   server: clean
245   cron: "20 04 2,6,10,14,18,22,26,30 * *"
246
247 - name: clean_cloudgoogle_threatintelligenceblog_posts
248   project: Sentinels_Light
249   spider: CloudGoogle_ThreatIntelligenceBlog
250   server: clean
251   cron: "25 04 2,6,10,14,18,22,26,30 * *"
252
253 - name: clean_talosintelligence_blog_posts
254   project: Sentinels_Light
255   spider: TalosIntelligence_Blog
256   server: clean
257   cron: "30 04 2,6,10,14,18,22,26,30 * *"
258
259 # === CLEAN WEB: Discord Crawlers ===
260 - name: clean_discord_cti_blackhills
261   project: Sentinels_Light
262   spider: Discord_CTI_BlackHills_Crawler
263   server: clean
264   lock_group: Discord
265   cron: "45 05 * * *"
266   jitter: 1200
```

```

267
268 - name: clean_discord_grouped_rootkit
269   project: Sentinels_Light
270   spider:
271     Discord_RootKit_OwlSec_BlackHills_ReverseEng_HackDifferent_Crawler
272   server: clean
273   lock_group: Discord
274   cron: "15 07 * * *"
275   jitter: 1200

```

**Listing A.1:** Declarative Scheduler Configuration

## A.2. Dynamic Loading Logic (scheduler.py)

The `scheduler.py` service loads `config.yaml` at startup, expands environment variables (e.g., Scrapyd and SMTP), and registers each task as an APScheduler job backed by a persistent SQL job store. Cron expressions are mapped to execution times via `CronTrigger` using the configured timezone, with reliability controls such as misfire handling, overlap prevention, jitter, and optional lock groups. A small FastAPI layer exposes health and job-inspection endpoints.

```

1
2 # /app/scheduler.py
3
4 import os, time, logging, yaml
5 from typing import Any, Dict, List, Optional
6 from pytz import timezone
7 from datetime import datetime
8 from fastapi import FastAPI
9 from pydantic import BaseModel
10 from apscheduler.schedulers.background import BackgroundScheduler
11 from apscheduler.triggers.cron import CronTrigger
12 from apscheduler.jobstores.sqlalchemy import SQLAlchemyJobStore
13
14 logging.basicConfig(level=os.getenv("LOG_LEVEL", "INFO"), format="%(
15     asctime)s %(levelname)s %(message)s")
16 log = logging.getLogger("scheduler")
17
18 app = FastAPI(title="Crawler Scheduler")
19
20 # --- Config Helpers ---
21 def _env_subst(v: Any) -> Any:
22     return os.path.expandvars(v) if isinstance(v, str) else v

```

```

22
23 def load_config(path="config.yaml") -> Dict[str, Any]:
24     with open(path, "r", encoding="utf-8") as f:
25         cfg = yaml.safe_load(f) or {}
26         for k, v in (cfg.get("servers") or {}).items():
27             cfg["servers"][k] = _env_subst(v)
28         if "smtp" in cfg and isinstance(cfg["smtp"], dict):
29             for k, v in list(cfg["smtp"].items()):
30                 cfg["smtp"][k] = _env_subst(v) if isinstance(v, str) else v
31     return cfg
32
33 cfg: Dict[str, Any] = {}
34 sched: Optional[BackgroundScheduler] = None
35
36 # --- DTOs ---
37 class JobDTO(BaseModel):
38     id: str
39     name: str
40     nextRunAt: Optional[str]
41     trigger: str
42     type: str # "spider" | "email" | other
43     project: Optional[str] = None
44     spider: Optional[str] = None
45     server: Optional[str] = None
46
47 def _cron_string(tr: CronTrigger) -> str:
48     """Extract crontab string from CronTrigger."""
49     try:
50         fields = [f.expression for f in tr.fields]
51         return " ".join(fields[:5])
52     except Exception:
53         return str(tr)
54
55 def _infer_type(job_kwargs: Dict[str, Any], func_ref: str) -> str:
56     if "spider" in job_kwargs and "project" in job_kwargs:
57         return "spider"
58     if "subject" in job_kwargs and "to" in job_kwargs:
59         return "email"
60     return "other"
61
62 def _job_to_dto(job) -> JobDTO:
63     tr = job.trigger
64     kwargs = job.kwargs or {}

```

```

65     jt = _infer_type(kwargs, job.func_ref if hasattr(job, "func_ref")
66     else "")
67     nx = job.next_run_time
68     return JobDTO(
69         id=job.id,
70         name=getattr(job, "name", job.id),
71         nextRunAt=nx.astimezone(timezone(cfg.get("timezone", "UTC"))).
72         isoformat() if nx else None,
73         trigger=_cron_string(tr) if isinstance(tr, CronTrigger) else str
74         (tr),
75         type=jt,
76         project=kwargs.get("project"),
77         spider=kwargs.get("spider"),
78         server=kwargs.get("server_key") or kwargs.get("server"),
79     )
80
81 # --- FastAPI Lifecycle ---
82 @app.on_event("startup")
83 def start():
84     global cfg, sched
85     cfg = load_config("config.yaml")
86     tz = timezone(cfg.get("timezone", "UTC"))
87     prevent_overlap = (cfg.get("defaults", {}) or {}).get("
88     prevent_overlap", True)
89     misfire_grace_seconds = (cfg.get("defaults", {}) or {}).get("
90     misfire_grace_seconds", 300)
91
92     jobstores = {"default": SQLAlchemyJobStore(url=os.getenv("
93     JOBSTORE_URL", "sqlite:///app/data/jobs.sqlite"))}
94     sched = BackgroundScheduler(
95         jobstores=jobstores,
96         timezone=tz,
97         job_defaults={"coalesce": True, "max_instances": 1, "
98     misfire_grace_time": misfire_grace_seconds},
99     )
100     sched.start()
101
102 # Build Spider -> LockGroup map
103 spider_group_map = {}
104 for t in cfg.get("tasks", []):
105     if "spider" in t and "lock_group" in t:
106         spider_group_map[t["spider"]] = t["lock_group"]
107
108 # Register jobs from YAML

```

```

102     for t in cfg.get("tasks", []):
103         trigger = CronTrigger.from_crontab(t["cron"], timezone=tz)
104
105         if "jitter" in t:
106             try:
107                 trigger.jitter = int(t["jitter"])
108             except ValueError:
109                 log.warning(f"Invalid jitter value for {t.get('name')},
ignored.")
110
111         job_id = t.get("id") or t["name"]
112         task_type = (t.get("type") or "spider").lower()
113         if task_type == "spider":
114             func_ref = "jobs:schedule_runner"
115             kwargs = {
116                 "project": t["project"],
117                 "spider": t["spider"],
118                 "server_key": t["server"],
119                 "servers": cfg["servers"],
120                 "prevent_overlap": prevent_overlap,
121                 "spider_group_map": spider_group_map,
122             }
123         elif task_type == "email":
124             func_ref = "jobs:send_email"
125             kwargs = {
126                 "to": t["to"],
127                 "subject": t["subject"],
128                 "body": t["body"],
129                 "smtp": cfg.get("smtp"),
130             }
131         else:
132             log.warning(f"Task {job_id} ignored: unknown type {task_type
!r}")
133             continue
134
135         sched.add_job(func_ref, trigger, id=job_id, replace_existing=
True, kwargs=kwargs, name=t.get("name", job_id))
136
137         log_msg = f"Registered job {job_id} --> {t['cron']} (TZ={tz})"
138         if "jitter" in t:
139             log_msg += f" [Jitter: {t['jitter']}s]"
140         log.info(log_msg)
141
142     log.info("Scheduler started + API ready.")

```

```

143
144 @app.on_event("shutdown")
145 def shutdown():
146     if sched:
147         sched.shutdown()
148
149 # --- Endpoints ---
150 @app.get("/health")
151 def health():
152     return {"status": "ok", "now": datetime.utcnow().isoformat() + "Z"}
153
154 @app.get("/jobs")
155 def list_jobs():
156     jobs = (sched.get_jobs() if sched else [])
157     return {"count": len(jobs), "jobs": [_job_to_dto(j).model_dump() for
158         j in jobs]}
159
160 @app.get("/jobs/next")
161 def next_jobs(limit: int = 8):
162     jobs = (sched.get_jobs() if sched else [])
163     rows = sorted(jobs, key=lambda j: (j.next_run_time or datetime.max))
164     out = [_job_to_dto(j).model_dump() for j in rows if j.next_run_time
165        ][:limit]
166     return out
167
168 @app.post("/jobs/{job_id}/runnow")
169 def run_now(job_id: str):
170     j = sched.get_job(job_id) if sched else None
171     if not j:
172         return {"ok": False, "error": "job not found"}
173     sched.add_job(j.func, trigger="date", run_date=datetime.utcnow(),
174         kwargs=j.kwargs)
175     return {"ok": True}

```

Listing A.2: Dynamic Job Registration Logic

### A.3. Job Execution Handlers (jobs.py)

The `jobs.py` module contains the runtime actions used by the scheduler: launching spiders on Scrapy and sending HITL email alerts. The `schedule_runner` handler prevents duplicate executions by checking Scrapy running/pending jobs and, using a global lock plus optional `lock_group` logic, enforces mutual exclusion when multiple triggers fire simultaneously. The `send_email` handler delivers notification emails via SMTP, reading

parameters from the YAML smtp block or environment variables (also supporting `_FILE` secrets for passwords).

```

1 # /app/jobs.py
2
3 import logging
4 import time
5 import os
6 import smtplib
7 import threading
8 from email.message import EmailMessage
9 import requests
10 from urllib.parse import urljoin
11
12 log = logging.getLogger("scheduler")
13
14 # Global lock to prevent race conditions during concurrent job starts
15 _scheduler_lock = threading.Lock()
16
17 # -----
18 # Spider Scheduling Functions
19 # -----
20
21 def _list_jobs(server_url: str, project: str):
22     r = requests.get(urljoin(server_url, f"/listjobs.json?project={
23 project}"), timeout=20)
24     r.raise_for_status()
25     return r.json()
26
27 def _schedule_job(server_url: str, project: str, spider: str,
28 jobid: str = None, args: dict | None = None, settings:
29 dict | None = None):
30     data = {"project": project, "spider": spider}
31     if jobid:
32         data["jobid"] = jobid
33     if args:
34         data.update(args)
35     if settings:
36         data["setting"] = [f"{k}={v}" for k, v in settings.items()]
37     r = requests.post(urljoin(server_url, "/schedule.json"), data=data,
38 timeout=30)
39     r.raise_for_status()
40     return r.json()
41
42 def schedule_runner(project: str,

```

```

40         spider: str,
41         server_key: str,
42         servers: dict,
43         prevent_overlap: bool = True,
44         spider_group_map: dict | None = None):
45     """
46     APScheduler job to start a spider on Scrapy.
47     Uses a lock to prevent concurrent jobs from starting overlapping
48     instances.
49     """
50     server_url = servers[server_key]
51     with _scheduler_lock:
52         try:
53             if prevent_overlap:
54                 jobs = _list_jobs(server_url, project)
55                 active = [*jobs.get("running", []), *jobs.get("pending",
56
57
58                 # Standard overlap check: same spider name
59                 if any(j.get("spider") == spider for j in active):
60                     log.info(f"[SKIP] {project}:{spider} already running
61 /pending on {server_url}")
62                     return
63
64                 # Lock Group check: mutual exclusion for a group of
65 spiders
66                 if spider_group_map:
67                     my_group = spider_group_map.get(spider)
68                     if my_group:
69                         for j in active:
70                             running_spider_name = j.get("spider")
71                             running_group = spider_group_map.get(
72 running_spider_name)
73
74                             if running_group == my_group:
75                                 log.info(f"[SKIP] {project}:{spider}
76 blocked by group '{my_group}' (active: {running_spider_name})")
77                                 return
78
79                 jobid = f"{project}-{spider}-{int(time.time())}"
80                 resp = _schedule_job(server_url, project, spider, jobid=
81 jobid)
82                 log.info(f"[OK] scheduled {project}:{spider} -> {resp}")
83         except Exception:

```



```

77         log.exception(f"[ERR] scheduling {project}:{spider}")
78
79
80 # -----
81 # Email Notification Functions
82 # -----
83
84 def _env_or_file(var: str) -> str | None:
85     """Reads variable from environment or from a file if VAR_FILE is
86     defined."""
87     path = os.getenv(f"{var}_FILE")
88     if path and os.path.exists(path):
89         try:
90             with open(path, "r", encoding="utf-8") as f:
91                 return f.read().strip()
92         except Exception:
93             pass
94     return os.getenv(var)
95
96 def _smtp_from_env_or_cfg(smtp_cfg: dict | None) -> dict:
97     smtp_cfg = smtp_cfg or {}
98     return {
99         "host": smtp_cfg.get("host") or os.getenv("SMTP_HOST"),
100         "port": int(smtp_cfg.get("port") or os.getenv("SMTP_PORT") or
101 587),
102         "user": smtp_cfg.get("user") or os.getenv("SMTP_USER"),
103         "password": smtp_cfg.get("password") or _env_or_file("SMTP_PASS"
104 ),
105         "use_tls": str(smtp_cfg.get("use_tls") or os.getenv("
106 SMTP_USE_TLS") or "true").lower() == "true",
107         "from_addr": smtp_cfg.get("from") or os.getenv("SMTP_FROM") or
108 smtp_cfg.get("user") or os.getenv("SMTP_USER"),
109     }
110
111 def send_email(to: str | list[str],
112               subject: str,
113               body: str,
114               smtp: dict | None = None):
115     """
116     APScheduler job to send SMTP email.
117     Requires SMTP_* environment variables or 'smtp' config block.
118     """
119     try:

```

```

116     cfg = _smtp_from_env_or_cfg(smtp)
117     if not cfg["host"]:
118         raise RuntimeError("SMTP host not configured")
119     if not cfg["from_addr"]:
120         raise RuntimeError("SMTP sender not configured")
121
122     recipients = [to] if isinstance(to, str) else to
123
124     msg = EmailMessage()
125     msg["From"] = cfg["from_addr"]
126     msg["To"] = ", ".join(recipients)
127     msg["Subject"] = subject
128     msg.set_content(body)
129
130     with smtplib.SMTP(cfg["host"], cfg["port"], timeout=30) as s:
131         s.ehlo()
132         if cfg["use_tls"]:
133             s.starttls()
134             s.ehlo()
135         if cfg["user"] and cfg["password"]:
136             s.login(cfg["user"], cfg["password"])
137         s.send_message(msg)
138
139     log.info(f"[MAIL OK] to={recipients} subject={subject!r}")
140 except Exception:
141     log.exception(f"[MAIL ERR] to={to} subject={subject!r}")

```

**Listing A.3:** Job Handlers for Spiders and HITL Notifications

# B | Data Warehouse Backbone

This appendix details the structure of the final DB. This is the core part of Argus collection system: querying this DB allows us to extract all relevant inferences about collected data.

## B.1. Sql\_init - Data Warehouse initialization script

```

1  -- =====
2  -- 1. MASTER TABLE (Hybrid SQL/NoSQL)
3  -- =====
4  CREATE TABLE IF NOT EXISTS intel_items (
5      id SERIAL PRIMARY KEY,
6
7      -- Taxonomy
8      source_type VARCHAR(50) NOT NULL, -- e.g., 'Report', 'Forum', 'DLS'
9      origin_name VARCHAR(255),         -- e.g., 'LockBit Blog', 'Mandiant
10
11      -- Content Payload
12      title TEXT NOT NULL,
13      summary TEXT,                    -- Abstractive summary generated
14      original_url TEXT,               by LLM
15
16      -- Temporal Dimensions
17      reference_date TIMESTAMP WITH TIME ZONE, -- Event occurrence
18      created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(), -- Ingestion time
19
20      -- FLEXIBLE METADATA (JSONB)
21      -- Stores source-specific fields (e.g., TLP color, author reputation
22      -- Indexable via GIN (Generalized Inverted Index)
23      source_metadata JSONB,
24
25      -- NATIVE FULL-TEXT SEARCH

```

```

26     -- Pre-computed TSVECTOR for high-performance text mining
27     full_text_search TSVECTOR
28 );
29
30 -- =====
31 -- 2. NORMALIZED ENTITY EXTRACTION
32 -- =====
33 -- Unique registry of threat entities (IoCs, Actors, Tools)
34 CREATE TABLE IF NOT EXISTS extracted_entities (
35     id SERIAL PRIMARY KEY,
36     entity_type VARCHAR(50) NOT NULL, -- e.g., 'IP', 'SHA256', 'APT'
37     entity_value TEXT NOT NULL,       -- Normalized value
38     UNIQUE (entity_type, entity_value)
39 );
40
41 -- Many-to-Many Bridge Table
42 -- Enables pivoting: "Find all items related to Entity X"
43 CREATE TABLE IF NOT EXISTS item_entities (
44     item_id INTEGER REFERENCES intel_items(id) ON DELETE CASCADE,
45     entity_id INTEGER REFERENCES extracted_entities(id) ON DELETE
46     CASCADE,
47     PRIMARY KEY (item_id, entity_id)
48 );
49 -- =====
50 -- 3. KNOWLEDGE BASE (MITRE ATT&CK MAPPING)
51 -- =====
52 -- Static reference data for enrichment
53
54 CREATE TABLE IF NOT EXISTS kb_actors (
55     id SERIAL PRIMARY KEY,
56     mitre_id VARCHAR(50) UNIQUE,
57     name VARCHAR(255),
58     description TEXT,
59     aliases TEXT[],
60     last_update TIMESTAMP WITH TIME ZONE
61 );
62
63 CREATE TABLE IF NOT EXISTS kb_malware (
64     id SERIAL PRIMARY KEY,
65     mitre_id VARCHAR(50) UNIQUE,
66     name VARCHAR(255),
67     type VARCHAR(50),
68     description TEXT,

```

```

69     platforms TEXT[],
70     last_update TIMESTAMP WITH TIME ZONE
71 );
72
73 CREATE TABLE IF NOT EXISTS kb_techniques (
74     id SERIAL PRIMARY KEY,
75     mitre_id VARCHAR(50) UNIQUE,
76     name VARCHAR(255),
77     description TEXT,
78     last_update TIMESTAMP WITH TIME ZONE
79 );
80
81 -- =====
82 -- 4. PERFORMANCE OPTIMIZATION (INDEXES)
83 -- =====
84
85 -- GIN Index for JSONB querying (e.g., source_metadata->>'tlp' = 'RED')
86 CREATE INDEX IF NOT EXISTS idx_metadata ON intel_items USING GIN (
87     source_metadata);
88
89 -- GIN Index for Full-Text Search
90 CREATE INDEX IF NOT EXISTS idx_full_text ON intel_items USING GIN (
91     full_text_search);
92
93 -- B-Tree Indexes for standard filtering
94 CREATE INDEX IF NOT EXISTS idx_intel_items_source_type ON intel_items(
95     source_type);
96 CREATE INDEX IF NOT EXISTS idx_intel_items_reference_date ON intel_items
97     (reference_date DESC);
98 CREATE INDEX IF NOT EXISTS idx_entities_value ON extracted_entities(
99     entity_value);

```

**Listing B.1:** PostgreSQL Schema Definition (init.sql)

