



Università degli Studi di Salerno

Dipartimento di Informatica

---

Corso di Laurea Magistrale in Sicurezza Informatica

# Automazione della Threat Intelligence tramite LLM: Progettazione, implementazione e benchmarking comparativo per la generazione di bundle STIX 2.1

## **Relatori**

Prof. Arcangelo Castiglione

Ing. Antonio Formato

## **Candidato**

Giulio Triggiani

Matricola: 0522501328

---

Anno Accademico 2025/2026



# Indice

<b>Abstract</b>	<b>1</b>
<b>1 Introduzione</b>	<b>2</b>
1.1 Contesto: L'ecosistema della Cyber Threat Intelligence . . . . .	2
1.1.1 Definizione e imperativi strategici della CTI nella difesa proattiva . . .	2
1.1.2 Il ciclo di vita della Cyber Threat Intelligence . . . . .	3
1.1.3 La sfida cruciale della condivisione delle informazioni ed il ruolo degli standard . . . . .	4
1.2 Definizione del progetto: Obiettivi ed impatto . . . . .	4
1.2.1 Obiettivo primario: Dal testo grezzo al grafo STIX . . . . .	4
1.2.2 Obiettivi secondari: Sviluppo e valutazione . . . . .	5
1.2.3 L'impatto della ricerca: Contributi attesi . . . . .	6
1.3 Struttura della tesi . . . . .	6
<b>2 Background e stato dell'arte</b>	<b>8</b>
2.1 Fondamenti di Cyber Threat Intelligence . . . . .	8
2.1.1 I livelli della CTI: Strategica, Operativa, Tattica . . . . .	9
2.1.2 Gli Indicatori di Compromissione (IoC): La traccia digitale della minaccia	9
2.1.3 Tattiche, Tecniche e Procedure (TTP): Il <i>modus operandi</i> dell'avversario	10
2.1.4 La gerarchia del valore: Il modello della "Pyramid of Pain" . . . . .	11
2.1.5 Il framework MITRE ATT&CK: Una tassonomia globale del compor- tamento avversario . . . . .	13
2.2 Lo standard STIX 2.1 in dettaglio . . . . .	14
2.2.1 Architettura e approccio: Il modello a grafo connesso . . . . .	14
2.2.2 I componenti del modello dei dati: SDO, SCO e SRO . . . . .	15
2.2.3 Proprietà comuni e struttura . . . . .	17
2.3 Large Language Models (LLM) . . . . .	18
2.3.1 Architettura e addestramento: Dai Transformer al "System 2" Thinking	18
2.3.2 Prompt Engineering: Programmare con il linguaggio naturale . . . . .	19
2.3.3 Parametri di inferenza e sfide intrinseche . . . . .	19
2.3.4 Panoramica dei modelli utilizzati: Evoluzione e differenze . . . . .	21
2.4 Lavori correlati e stato dell'arte . . . . .	21
2.4.1 Approcci tradizionali: Espressioni regolari e NLP classico . . . . .	22

2.4.2	L'era della Generative AI: LLM per la CTI . . . . .	22
2.4.3	Gap di ricerca e posizionamento della tesi . . . . .	22
<b>3</b>	<b>Progettazione e metodologia</b>	<b>23</b>
3.1	Architettura della soluzione: Il Workflow GenAI-STIX . . . . .	23
3.1.1	Data Ingestion & Normalization . . . . .	24
3.1.2	Advanced IOC Extraction . . . . .	24
3.1.3	Programmatic Object Generation . . . . .	25
3.1.4	Comprehensive Entity Extraction . . . . .	25
3.1.5	Final Bundling & Orchestration . . . . .	25
3.2	Dataset e costruzione del Ground truth . . . . .	25
3.2.1	Strategia di ricerca e difficoltà di reperimento . . . . .	27
3.2.2	Fonte primaria selezionata: NCSC Malware Reports . . . . .	27
3.2.3	Criteri di esclusione: Il caso CISA . . . . .	27
3.3	Prompt Engineering e strategie di istruzione . . . . .	28
3.3.1	Role Prompting: Definizione della "persona" . . . . .	28
3.3.2	Few-Shot Learning: Apprendimento per analogia . . . . .	28
3.3.3	Chain-of-Thought e scomposizione del task . . . . .	29
3.3.4	Vincoli di formato e JSON Mode . . . . .	30
3.4	Metodologia di valutazione e protocollo sperimentale . . . . .	30
3.4.1	Il problema dell'allineamento . . . . .	30
3.4.2	Pipeline 1: Valutazione semantica . . . . .	30
3.4.3	Pipeline 2: Graph Evaluation . . . . .	33
<b>4</b>	<b>Implementazione</b>	<b>37</b>
4.1	Stack tecnologico e ambiente di sviluppo . . . . .	37
4.1.1	Manipolazione dati e standard: La libreria stix2 . . . . .	38
4.1.2	Interazione con gli LLM: openai . . . . .	38
4.1.3	Normalizzazione and Ingestion: MarkItDown e iocextract . . . . .	39
4.2	Dettagli implementativi del generatore . . . . .	39
4.2.1	Setup e configurazione . . . . .	39
4.2.2	Ingestion and Normalization Markdown . . . . .	40
4.2.3	Estrazione degli IoC . . . . .	40
4.2.4	Generazione SCO & Indicator . . . . .	43
4.2.5	Estrazione SDO . . . . .	45
4.2.6	Assemblaggio finale . . . . .	46
4.3	Implementazione delle Pipeline di valutazione . . . . .	47
4.3.1	Pipeline 1: Semantic Evaluation . . . . .	47
4.3.2	Pipeline 2: Graph Evaluation . . . . .	49
4.4	Sfide tecniche e soluzioni adottate . . . . .	50
4.4.1	Interoperabilità e confronto dei Pattern YARA . . . . .	50
4.4.2	La sfida "topologica" delle relazioni . . . . .	51

<b>5</b>	<b>Analisi e discussione dei risultati</b>	<b>52</b>
5.1	Analisi delle performance di rilevamento . . . . .	52
5.1.1	I risultati nell'estrazione dei TTP . . . . .	53
5.1.2	Affidabilità nel rilevamento delle vittime (Identity) . . . . .	54
5.1.3	La gestione degli indicatori tecnici (Indicator) . . . . .	55
5.1.4	Benchmark sintetico dei modelli . . . . .	55
5.1.5	Analisi dell'impatto della temperatura . . . . .	56
5.2	Valutazione olistica del grafo . . . . .	56
5.2.1	Profondità descrittiva nei comportamenti (Attack Pattern) . . . . .	57
5.2.2	L'eccellenza di Llama 4 su Malware e Identity . . . . .	57
5.2.3	Coerenza topologica delle relazioni . . . . .	58
5.2.4	Impatto del Reasoning Effort (GPT-5 Low vs High) . . . . .	58
5.2.5	Analisi della stabilità per Bundle . . . . .	58
5.2.6	Analisi complessiva dei risultati . . . . .	58
5.3	Discussione critica e implicazioni operative . . . . .	60
5.3.1	Il superamento dei test . . . . .	60
5.3.2	Il dilemma "Open vs Closed" . . . . .	60
5.3.3	Il problema della "libertà espressiva" dello standard . . . . .	61
5.3.4	Il caso CISA e la divergenza topologica . . . . .	61
5.3.5	Limiti attuali ed implicazioni per l'adozione Operativa . . . . .	61
<b>6</b>	<b>Conclusioni e sviluppi futuri</b>	<b>63</b>
6.1	Sintesi del percorso di ricerca . . . . .	63
6.2	Contributi della tesi . . . . .	64
6.3	Sviluppi futuri: Verso la "Agentic CTI" . . . . .	64
6.3.1	Specializzazione tramite Fine-Tuning . . . . .	64
6.3.2	Architetture Multi-Agent . . . . .	64
6.4	Conclusioni finali . . . . .	65

# Elenco delle figure

2.1	I tre livelli della CTI . . . . .	9
2.2	Il ciclo di vita degli IoC . . . . .	11
2.3	The Pyramid of Pain . . . . .	13
2.4	MITRE ATT&CK matrix example . . . . .	14
2.5	Esempio di struttura a grafo STIX . . . . .	15
2.6	Esempio di relazioni STIX 2.1 . . . . .	17
3.1	Diagramma di flusso del tool di generazione . . . . .	26
3.2	Diagramma di flusso della valutazione semantica . . . . .	34
3.3	Diagramma di flusso della Graph Evaluation . . . . .	36
5.1	Performance per classe: Gemini 2.5 Pro high reasoning . . . . .	53
5.2	Performance per classe: GPT-5 high reasoning . . . . .	54
5.3	Performance per classe: Llama 3.3 $T = 0.1$ . . . . .	54
5.4	Performance per classe: Llama 4 Maverick $T = 0.1$ . . . . .	55
5.5	Performance media per tipo di oggetto Gemini 2.5 Pro . . . . .	56
5.6	Performance media per tipo di oggetto GPT-5 . . . . .	57
5.7	Performance media per tipo di oggetto Llama 4 Maverick . . . . .	58
5.8	Performance per bundle Gemini 2.5 Pro . . . . .	59
5.9	Performance per bundle GPT-5 . . . . .	59
5.10	Performance per bundle Llama 4 Maverick . . . . .	59

# Abstract

L'efficacia della Cyber Threat Intelligence (CTI) nella difesa proattiva si confronta oggi con la sfida di gestire efficientemente ingenti volumi di informazioni non strutturate. La maggior parte delle informazioni sulle minacce viene diffusa attraverso report testuali non strutturati, rendendo la conversione nello standard STIX 2.1 un processo manuale, lento e non scalabile, perciò questa tesi propone un framework innovativo progettato per automatizzarla. L'architettura sviluppata adotta un approccio ibrido che orchestra componenti deterministiche (basate su Regex), per l'estrazione rigorosa degli indicatori tecnici (IoC), e componenti basate su Large Language Models, per l'inferenza semantica di entità e relazioni complesse. Per superare la scarsità di dati di training pubblici, è stato costruito un dataset (*Ground Truth*) basato su report reali del National Cyber Security Centre (NCSC). Su tale dataset è stato condotto un benchmark comparativo rigoroso che ha coinvolto cinque famiglie di modelli (inclusi GPT-4o, GPT-5, Gemini 2.5 Pro e Llama 4) valutati attraverso una doppia pipeline di analisi: una focalizzata sulle metriche di rilevamento e l'altra sulla fedeltà descrittiva del grafo. I risultati sperimentali dimostrano che i modelli *High-Reasoning* (GPT-5 e Gemini 2.5) hanno superato la barriera della comprensione semantica raggiungendo una precisione superiore al 94% nell'estrazione delle Tattiche, Tecniche e Procedure (TTP), risolvendo di fatto il problema della mappatura automatica sul framework MITRE ATT&CK. Lo studio conclude che l'integrazione di LLM nella pipeline CTI non sia più una prospettiva teorica, ma una realtà tecnica matura capace di trasformare l'analista da "*data entry*" a supervisore strategico.

# Capitolo 1

## Introduzione

In questo capitolo introduttivo verranno analizzati il contesto in cui si inserisce il lavoro di tesi e le motivazioni che ne sono alla base.

In particolare, si tratteranno i seguenti punti:

- **Contesto della Cyber Threat Intelligence (CTI):** Si definirà l'ecosistema della CTI, il suo ciclo di vita e le sfide attuali legate alla condivisione delle informazioni.
- **Standardizzazione e complessità operativa:** Si analizzerà il "collo di bottiglia" umano causato dalla traduzione manuale dei report testuali nello standard STIX 2.1 e l'impatto di questa latenza.
- **Obiettivo della tesi:** Verranno definiti gli scopi principali e secondari di questo studio, evidenziando il contributo pratico e scientifico del sistema proposto.
- **Struttura della tesi:** verrà fornita una panoramica della suddivisione dei capitoli successivi.

### 1.1 Contesto: L'ecosistema della Cyber Threat Intelligence

#### 1.1.1 Definizione e imperativi strategici della CTI nella difesa proattiva

La Cyber Threat Intelligence (CTI) rappresenta un pilastro fondamentale delle moderne strategie di sicurezza informatica. In un panorama di minacce digitali in continua e rapida evoluzione, la CTI è la disciplina che si occupa della raccolta, dell'elaborazione, dell'analisi e dell'organizzazione delle informazioni relative a minacce informatiche, attori malevoli e loro metodologie. Essa trascende la mera raccolta di dati sulle minacce, configurandosi come un processo analitico che trasforma dati grezzi (come indirizzi IP, hash di file o log) in informazioni contestualizzate, meccanismi di attacco, indicatori, implicazioni e, soprattutto, in consigli attuabili riguardo minacce esistenti o emergenti. [1][2][3]

L'essenza della CTI risiede nella sua capacità di fornire alle organizzazioni la conoscenza necessaria per prendere decisioni informate e tempestive, permettendo di anticipare le mosse degli avversari anziché limitarsi a reagire ai loro attacchi. Questo paradigma segna una transizione cruciale da una postura di sicurezza puramente reattiva, focalizzata sulla risposta

agli incidenti (*Incident Response*) dopo che si sono verificati, ad una postura proattiva e predittiva, guidata dall'intelligence. L'obiettivo strategico è comprendere profondamente le motivazioni, le capacità e le infrastrutture degli attori delle minacce per rafforzare le difese, ridurre la superficie di attacco ed ottimizzare l'allocazione delle risorse di sicurezza. [2][3]

### 1.1.2 Il ciclo di vita della Cyber Threat Intelligence

L'efficacia della CTI non deriva da un'attività singola, ma da un processo ciclico e iterativo noto come "ciclo di vita della Cyber Threat Intelligence". Questo framework strutturato garantisce che l'intelligence prodotta sia pertinente, accurata e utile per l'organizzazione. Sebbene le nomenclature possano variare leggermente, il processo si articola in fasi fondamentali che si alimentano a vicenda in un ciclo di miglioramento continuo:

1. **Pianificazione e direzione (Planning/Direction):** Questa fase iniziale è cruciale e strategica. L'organizzazione definisce gli obiettivi del proprio piano di intelligence, allineandoli alle esigenze degli stakeholder (ad es. management, SOC, team di *Incident Response*) e ai requisiti di business. Vengono poste domande fondamentali: Quali sono gli asset critici da proteggere? Quali sono le minacce più probabili per il nostro settore? Quali informazioni sono necessarie per migliorare le nostre difese?
2. **Raccolta (Collection):** Una volta definiti i requisiti inizia la fase di raccolta dei dati grezzi le cui informazioni vengono attinte da una vasta gamma di fonti, sia interne che esterne. Le fonti interne includono log di rete, dati da dispositivi di sicurezza (SIEM, EDR) e report sugli incidenti, invece le fonti esterne comprendono feed di intelligence open-source (OSINT) e commerciali, blog di sicurezza, forum sul deep e dark web e piattaforme di condivisione specifiche del settore.
3. **Elaborazione (Processing):** I dati grezzi raccolti sono spesso eterogenei, non strutturati e voluminosi, ma la fase di elaborazione li trasforma in un formato adatto all'analisi. Questo può includere l'organizzazione, la formattazione, la decrittazione, la traduzione e l'estrazione di indicatori specifici di log.
4. **Analisi (Analysis):** I dati elaborati diventano *intelligence*. Gli analisti umani, supportati da strumenti avanzati, esaminano le informazioni per identificare pattern, correlazioni, Tattiche, Tecniche e Procedure (TTP) e per contestualizzare le minacce: si riesce così a rispondere alle domande poste durante la pianificazione. È proprio in questa fase, ad alta intensità cognitiva e prevalentemente manuale, che si concentra lo sforzo maggiore degli specialisti della CTI.
5. **Disseminazione (Dissemination):** L'intelligence prodotta ha valore solo se raggiunge le persone giuste al momento giusto e in un formato comprensibile e attuabile. I risultati vengono distribuiti agli stakeholder pertinenti, in diversi formati: report strategici per il management, avvisi tattici per i team operativi o feed di dati leggibili da macchine per gli strumenti di sicurezza automatizzati.

6. **Feedback:** Il ciclo si chiude con la raccolta di feedback dagli stakeholder. Questo input è essenziale per valutare l'efficacia dell'intelligence prodotta e per affinare l'intero processo, ricalibrando i requisiti e migliorando le tecniche di analisi.

[2][3][4]

### 1.1.3 La sfida cruciale della condivisione delle informazioni ed il ruolo degli standard

In un panorama di minacce sempre più interconnesse, nessuna organizzazione può difendersi efficacemente in isolamento, la condivisione di CTI (*information sharing*) tra aziende, settori e agenzie governative è un moltiplicatore di forza che consente una difesa collettiva. Tuttavia, la condivisione efficace è storicamente ostacolata da diverse sfide: mancanza di fiducia, preoccupazioni per la riservatezza e barriere legali.

Dal punto di vista tecnico, la sfida più significativa è la **mancanza di interoperabilità**. La stragrande maggioranza della CTI di alta qualità viene prodotta e disseminata sotto forma di report non strutturati: blog post, avvisi governativi e articoli di ricerca scritti in linguaggio naturale. Se ogni organizzazione utilizza formati di dati proprietari o descrizioni testuali non standardizzate, la capacità di aggregare, correlare e agire rapidamente sull'intelligence condivisa è drasticamente ridotta.

Questa problematica evidenzia la necessità di un **linguaggio comune**, uno standard *de facto* che permetta una comunicazione delle informazioni sulle minacce che sia fluida, coerente e, soprattutto, automatizzata. Avere uno standard robusto è il prerequisito fondamentale per trasformare l'intelligence da un prodotto di analisi manuale a un flusso di dati utilizzabile in contesti operativi e machine-readable, gettando così le basi per la discussione sul ruolo dello standard STIX. [5][6][7]

## 1.2 Definizione del progetto: Obiettivi ed impatto

Per affrontare la lenta e dispendiosa traduzione manuale di intelligence testuale in formati strutturati, che determina il "collo di bottiglia", questa tesi si pone una serie di obiettivi chiari e misurabili. L'intento è fornire un contributo sia scientifico che pratico al campo della cybersecurity, indagando come le più recenti innovazioni nel campo dell'Intelligenza Artificiale possano risolvere questa sfida critica per la comunità CTI.

### 1.2.1 Obiettivo primario: Dal testo grezzo al grafo STIX

L'obiettivo primario di questo lavoro è la **progettazione, l'implementazione** e la **valutazione** di un sistema end-to-end che utilizzi un Large Language Model (LLM) per automatizzare il processo di analisi dei report di CTI non strutturati, ponendo particolare attenzione alle capacità degli LLM di inferire relazioni implicite che, come vedremo, rappresentano un punto cruciale per l'Intelligence.

Il fine ultimo del sistema è duplice:

1. **Estrazione di entità e relazioni:** Analizzare i report di CTI redatti in linguaggio naturale (come blog post, analisi di vendor o advisory governativi) per identificare ed estrarre non solo le entità chiave della minaccia (es **Threat-Actor**, **Malware**, **Indicator**, **Attack-Pattern**), ma anche, in modo cruciale, le complesse **relazioni** semantiche che intercorrono tra di esse.
2. **Generazione strutturata:** Convertire le informazioni estratte in bundle STIX 2.1 che siano, non solo **sintatticamente validi** (ovvero conformi allo schema JSON ufficiale dello standard), ma anche e soprattutto **semanticamente corretti**. Con questo si intende che l'output del modello deve catturare il significato e il contesto effettivo della minaccia descritta nel report, rappresentando semanticamente e in modo corretto e veritiero l'intelligence presente nello stesso e non una semplice corrispondenza letterale.

Il sistema proposto mira a funzionare come un "assistente analista" capace di processare un report testuale, produrre un file STIX 2.1 *machine-readable* che possa essere direttamente usato in altre piattaforme di sicurezza o come base di partenza per la creazione di un file STIX 2.1 più ricco e complesso, ma che richieda agli esperti CTI meno tempo e risorse per essere prodotto e che possa poi essere usato per arricchire la difesa automatizzata.

### 1.2.2 Obiettivi secondari: Sviluppo e valutazione

Per raggiungere l'obiettivo primario, sono stati definiti i seguenti obiettivi secondari:

- **Indagare l'efficacia del Prompt Engineering:** Si intende analizzare e testare sistematicamente diverse strategie di *Prompt Engineering* per istruire l'LLM sui compiti specifici di estrazione e mappatura STIX, che è un dominio altamente specializzato.
- **Sviluppare una metodologia di valutazione rigorosa:** Si prevede la creazione di un dataset *Ground Truth*, si misureranno le performance del sistema e si analizzerà l'adeguatezza delle metriche tradizionali di *information extraction* (Precision, Recall e F1-Score) validando prima sintatticamente i report prodotti, per assicurarsi che aderiscano allo standard STIX 2.1, e poi confrontando due approcci per la **valutazione semantica**. Il primo valuta il grafo STIX sui singoli oggetti che lo compongono e le relazioni che intercorrono tra di loro. Il secondo, invece, valuta l'intero grafo nel suo complessivo (*graph-based similarity*). Lo scopo è quello di ottenere una valutazione il più onesta possibile delle potenzialità degli LLM applicati a questo ambito.
- **Identificazione limiti e potenzialità:** Si analizzeranno criticamente i risultati per identificare i limiti attuali e le potenzialità future dell'uso di LLM *general-purpose* (come GPT-5 e Gemini 2.5) per questo compito. Questo include sia l'analisi delle tipologie di errore più comune (es. "allucinazioni") che delle aree di maggiore difficoltà (es. estrazione di relazioni complesse implicite nel testo del report).

### 1.2.3 L'impatto della ricerca: Contributi attesi

Questa tesi si propone di contribuire al progresso della conoscenza in un'area di ricerca critica e complessa come può essere quella della CTI e degli LLM applicati alla CTI e di fornire un risultato pratico e tangibile.

Il **contributo scientifico** risiede nell'applicazione e nella valutazione empirica di LLM di ultima generazione a un problema persistente e rilevante nella CTI. A differenza di molti lavori esistenti che si concentrano sulla generazione di grafi parziali o fanno uso di LLM obsoleti, questo studio affronta la generazione olistica dei principali oggetti e relazioni STIX. Si esplora come le capacità emergenti di comprensione contestuale di questi modelli possano essere sfruttate per un compito che va oltre la *Named Entity Recognition* (NER), ma affronta contesti complessi e l'uso di terminologia specifica, richiedendo la comprensione e la generazione di un grafo di relazioni complesso.

Il **contributo pratico** è altrettanto significativo: il sistema sviluppato rappresenta un *proof-of-concept* per uno strumento in grado di accelerare drasticamente il ciclo di vita della CTI. Automatizzando o semi-automatizzando la conversione da testo a STIX, si può ridurre significativamente il carico di lavoro degli analisti di sicurezza e questo permetterebbe loro di concentrarsi su compiti analitici di più alto livello, come l'analisi di ipotesi e l'attribuzione e la valutazione strategica delle minacce, anziché sulla tediosa estrazione manuale. In ultima analisi, ciò si traduce in una maggiore tempestività e qualità dell'intelligence condivisa, attuando una risposta alle minacce più rapida ed efficace a livello di ecosistema.

## 1.3 Struttura della tesi

Il presente lavoro di tesi è organizzato in sei capitoli, strutturati per guidare il lettore in modo logico attraverso il processo di ricerca, dalla definizione del problema alla presentazione dei risultati.

- **Capitolo 1: Introduzione.** Stabilisce il contesto della ricerca, delinea il problema operativo della conversione da testo a STIX, definisce gli obiettivi e i contributi attesi del lavoro.
- **Capitolo 2: Background e stato dell'arte.** Fornisce le fondamenta teoriche necessarie, approfondendo i concetti di Cyber Threat Intelligence, le specifiche dello standard STIX 2.1, i principi di funzionamento dei Large Language Models e le pratiche di *Prompt Engineering* ad essi collegati. Analizza inoltre i lavori correlati per posizionare questa tesi nel panorama scientifico attuale.
- **Capitolo 3: Progettazione e metodologia.** Descrive in dettaglio l'architettura del sistema software implementato, il processo di creazione del corpus di dati e del *Ground Truth*, e il protocollo di valutazione utilizzato per misurare le performance.
- **Capitolo 4: Implementazione.** Presenta i dettagli tecnici dello sviluppo, includendo lo stack tecnologico, le librerie software utilizzate e le sfide pratiche affrontate durante la codifica del sistema.

- **Capitolo 5: Analisi e discussione dei risultati.** Costituisce il nucleo sperimentale della tesi. Presenta e analizza i risultati quantitativi e qualitativi ottenuti, comparando i diversi modelli LLM e le strategie di confronto semantico.
- **Capitolo 6: Conclusioni e sviluppi futuri.** Riassume i risultati raggiunti, discute i contributi finali del lavoro e riconosce i limiti dello studio, delineando infine le possibili direzioni per future ricerche ed estensioni del progetto.

## Capitolo 2

# Background e stato dell'arte

Per apprezzare appieno la complessità e l'innovazione del progetto, è necessario analizzare i tre domini concettuali che si intersecano in questo lavoro: la Cyber Threat Intelligence, lo standard STIX 2.1 e i Large Language Models.

Inoltre, si esaminerà lo stato dell'arte della ricerca.

In particolare verranno trattati i seguenti punti:

- **Fondamenti di Cyber Threat Intelligence (CTI):** Si definiranno i concetti chiave della CTI, dai suoi livelli (strategico, operativo, tattico) ai framework essenziali per la sua comprensione, come la "*Pyramid of Pain*" e la matrice MITRE ATT&CK.
- **Lo standard STIX 2.1 in dettaglio:** Verrà analizzata l'architettura a grafo, l'approccio e i componenti principali (SDO, SCO, SRO) dello standard STIX, evidenziando inoltre la complessità della sua creazione manuale.
- **Large Language Models (LLM):** Si introdurranno i principi di funzionamento dei modelli linguistici, con un focus sui modelli scelti e sulle loro differenze, e sulle tecniche di *Prompt Engineering*.
- **Stato dell'arte e lavori correlati:** Si esaminerà la letteratura scientifica esistente riguardo l'automazione dell'estrazione di CTI, identificando i limiti degli approcci attuali e il "gap" di ricerca che questo progetto intende colmare.

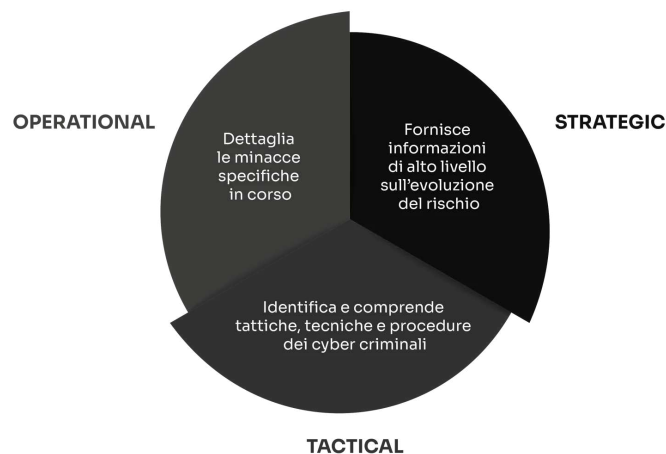
### 2.1 Fondamenti di Cyber Threat Intelligence

La Cyber Threat Intelligence è una disciplina fondamentale nel dominio della sicurezza informatica moderna. Essa rappresenta il processo di raccolta, analisi, distribuzione e scambio di informazioni relative a cyberattacchi, attori malevoli e loro infrastrutture. Come evidenziato nell'introduzione di questo lavoro, la CTI trascende la mera raccolta di dati grezzi: è un processo analitico che trasforma tali dati in informazioni contestualizzate e, soprattutto, in "consigli attuabili" (*actionable advice*). L'obiettivo finale è permettere alle organizzazioni di passare da una postura di sicurezza puramente reattiva a una **proattiva**, anticipando e mitigando le minacce prima che possano causare danni significativi. [2]

### 2.1.1 I livelli della CTI: Strategica, Operativa, Tattica

La CTI non è un concetto monolitico, ma si articola su tre livelli distinti, ognuno destinato ad un pubblico diverso e con finalità specifiche. Come illustrato nel libro *Visual Threat Intelligence*:

- **Intelligenza Strategica (Strategic CTI):** È il livello più alto. Si tratta di un'analisi "forward-looking" focalizzata su trend a lungo termine, rischi emergenti e fattori geopolitici. Non è tecnica, ma mira a informare le decisioni del top management (come CISO, CIO, e dirigenti) rispondendo alla domanda: "Chi ha i mezzi, il movente e l'opportunità per attaccarci?". [2][8]
- **Intelligenza Operativa (Operation CTI):** Questo livello fornisce una visione più completa sulle capacità, le motivazioni e le intenzioni degli attori delle minacce specifici. Risponde alle domande "Chi, Come e Perché" di un attacco. È destinata a team specializzati come i *Threat Hunter* e gli analisti di *Incident Response* per aiutarli a comprendere la natura degli avversari che potrebbero dover affrontare. [2][8]
- **Intelligenza Tattica (Tactical CTI):** È il livello più tecnico e immediato. Fornisce informazioni in tempo reale sulle minacce che stanno attivamente prendendo di mira un'organizzazione. Si concentra su indicatori tecnici per supportare le operazioni di sicurezza quotidiane e migliorare il rilevamento. È utilizzata principalmente dai *SOC Analyst* e risponde alla domanda: "Quali sono gli indicatori, il malware e le infrastrutture da bloccare ora?". [2][8]



Fonte: cyberoo.com

Figura 2.1: I tre livelli della CTI

### 2.1.2 Gli Indicatori di Compromissione (IoC): La traccia digitale della minaccia

Gli Indicatori di Compromissione (IoC) costituiscono l'elemento fondamentale dell'intelligence tattica. Sono definiti come **artefatti** o **caratteristiche osservabili** all'interno di una rete

o su un sistema e indicano, con un alto grado di confidenza, un'intrusione di sicurezza o un'attività malevola in corso. Essi rappresentano le "tracce" forensi lasciate dagli attaccanti durante le loro operazioni. [2][9]

Nel contesto delle analisi delle minacce, gli IoC non sono tutti uguali, ma come descritto da Lockheed Martin e ripreso dalla letteratura di settore, possono essere classificati in tre tipologie principali:

1. **Indicatori atomici (Atomic):** Sono indicatori indivisibili che mantengono il loro significato specifico nel contesto di un'intrusione. Esempi classici includono indirizzi IP, indirizzi email e identificatori di vulnerabilità (CVE). Questi sono i dati più semplici da estrarre automaticamente tramite pattern matching (es. Regex) nei report testuali.
2. **Indicatori calcolati (Computed):** Derivano dai dati relativi all'incidente tramite elaborazione. L'esempio più comune sono gli hash dei file (MD5, SHA256) che identificano univocamente un campione di malware specifico o un file malevolo.
3. **Indicatori comportamentali (Behavioral):** Combinano indicatori atomici e calcolati con logiche più complesse, descrivendo le azioni dell'intruso (es. "un documento macro che scarica un eseguibile da un IP specifico").

Un aspetto critico per l'automazione della CTI è comprendere che gli IoC non sono statici, ma essi seguono un **ciclo di vita** preciso che ne determina l'utilità operativa: [2][9]

- **Rivelazione (Revelation):** L'indicatore viene scoperto tramite analisi, collaborazione o intelligence esterna.
- **Maturazione (Maturation):** L'indicatore viene contestualizzato e utilizzato per aggiornare i sensori di sicurezza (es. regole firewall, firme IDS).
- **Utilità (Utility):** Il valore massimo si raggiunge quando l'indicatore permette di rilevare attivamente attività ostili o prevenire attacchi futuri.

Nello standard **STIX 2.1** un IoC è formalmente rappresentato dall'oggetto **SDO Indicator**. Questo oggetto contiene una proprietà fondamentale, **pattern**, che permette di descrivere la logica di rilevamento in un linguaggio strutturato (es. STIX Patterning o YARA), rendendo l'informazione immediatamente utilizzabile dai sistemi di difesa automatizzati. [10][7]

### 2.1.3 Tattiche, Tecniche e Procedure (TTP): Il *modus operandi* dell'avversario

Mentre gli IoC rispondono alla domanda "cosa" (quale file, quale IP), le Tattiche, Tecniche e Procedure (TTP) rispondono alla domanda "**come**" o "**cosa**" l'avversario stia cercando di ottenere. L'analisi dei TTP è essenziale per profilare gli attori delle minacce (*Threat Actors*) e comprendere le loro capacità operative. [2]

Questo concetto derivato dalla dottrina militare, si articola in tre livelli gerarchici:

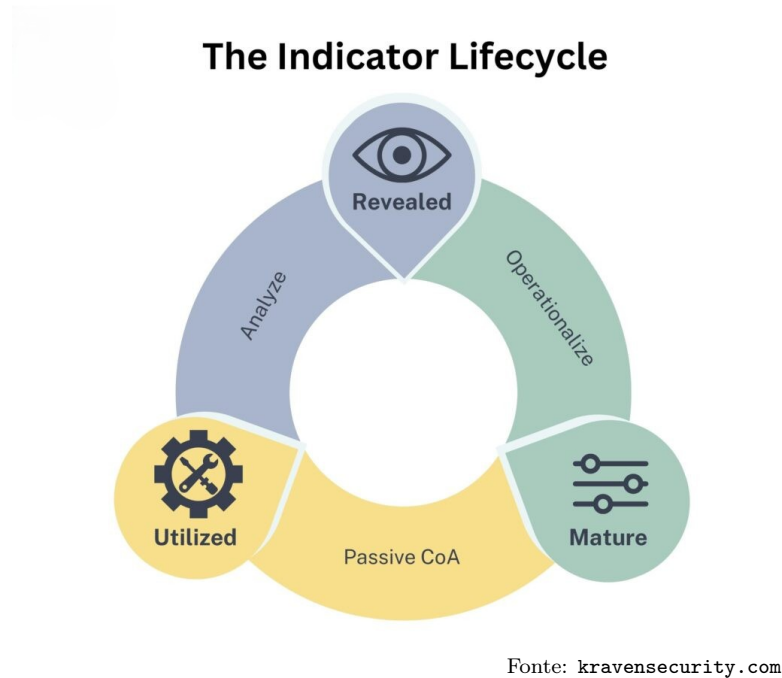


Figura 2.2: Il ciclo di vita degli IoC

- **Tattiche (Tactics):** Rappresentano gli obiettivi di alto livello dell'attaccante o la strategia generale (il "perché"). Esempi includono l'Ottenimento dell'Accesso Iniziale (*Initial Access*) o l'Esfiltrazione di Dati (*Exfiltration*).
- **Tecniche (Techniques):** Descrivono i metodi specifici utilizzati per realizzare una tattica (il "come"). Ad esempio, per ottenere l'*Initial Access*, un attaccante potrebbe usare la tecnica del *Phishing*.
- **Procedure (Procedures):** Forniscono i dettagli granulari, passo dopo passo, di come una tecnica viene eseguita da uno specifico attore. Ad esempio, l'uso di uno specifico script offuscato per eseguire il *Phishing*.

A differenza degli IoC, che possono essere modificati dall'attaccante (es. cambiando un indirizzo IP), i TTP sono legati al comportamento e alle abitudini dell'avversario. Modificare i TTP richiede agli avversari di riapprendere nuove competenze o sviluppare nuovi strumenti e questo rende l'intelligence basata sui TTP molto più duratura e preziosa per i difensori. [2]

Nello standard **STIX 2.1**, le TTP sono mappate principalmente attraverso l'oggetto **Attack-Pattern**, che è spesso collegato direttamente alle matrici del framework **MITRE ATT&CK**, il quale funge da knowledge base comune per la classificazione di questi comportamenti. [10][7]

#### 2.1.4 La gerarchia del valore: Il modello della "Pyramid of Pain"

Questo modello classifica gli indicatori in sei livelli, non basandosi sulla loro importanza tecnica, ma sulla quantità di "dolore" (in termini di costi, tempo e risorse) che infliggono all'avversario quando i difensori sono in grado di rilevarli e bloccarli. L'obiettivo della CTI

è spostare il focus difensivo verso la cima della piramide dove l'impatto sull'attaccante è massimo. [2]

Di seguito vengono dettagliati i sei livelli della piramide, dalla base al vertice:

1. **Valori hash (Hash Values) - *Trivial*:** Alla base della piramide si trovano gli hash dei file (es. MD5, SHA-1, SHA-256) che sono gli indicatori più comuni e facili da ottenere. Sebbene identifichino univocamente uno specifico file malevolo, sono estremamente fragili: basta che l'attaccante modifichi un singolo bit (ad esempio ricompilando o aggiungendo un byte nullo) per cambiare completamente l'hash, rendendo inutile il rilevamento precedente. Per l'attaccante aggirare un blocco basato su hash è un'operazione banale (*trivial*). [2][11]
2. **Indirizzi IP (IP Addresses) - *Easy*:** Il secondo livello comprende gli indirizzi IP, spesso utilizzati per identificare strutture di *Command and Control* (C2) o macchine compromesse. Anche questi sono considerati indicatori "facili" (*easy*) da gestire per un avversario. Gli attaccanti possono cambiare indirizzo IP rapidamente utilizzando proxy, VPN, reti anonime (come Tor) o semplicemente spostando la loro infrastruttura su nuovi server compromessi o cloud provider. [2]
3. **Nomi di dominio (Domain Names) - *Simple*:** Al terzo livello troviamo i nomi di dominio. Sebbene modificarli richieda un minimo sforzo in più rispetto a un IP (come la registrazione di un nuovo dominio o l'aggiornamento dei record DNS), l'operazione rimane "semplice" (*simple*). Molti attaccanti utilizzano algoritmi di generazione automatica dei domini (DGA) o provider di DNS dinamici per cambiare costantemente i domini che utilizzano, rendendo i blocchi statici poco efficienti nel lungo termine. [2]
4. **Artefatti di rete e host (Network & Host Artifacts) - *Annoying*:** Salendo nella piramide, entriamo nella zona in cui l'azione difensiva inizia a diventare "fastidiosa" (*annoying*) per l'attaccante. [2]
  - **Network Artifacts:** Includono pattern osservabili nel traffico di rete, come stringhe specifiche negli URI o valori distintivi nello User-Agent HTTP.
  - **Host Artifacts:** Includono tracce lasciate sui sistemi, come specifiche chiavi di registro create dal malware, nomi di file o directory distintivi. Costringere l'attaccante a modificare questi elementi richiede una riscrittura parziale del codice o la riconfigurazione dei tool, aumentando il costo dell'attacco.
5. **Strumenti (Tools) - *Challenging*:** Questo livello riguarda il software utilizzato dall'avversario per raggiungere il suo obiettivo. Include utility per creare documenti malevoli, backdoor per il C2, o strumenti per il movimento laterale e il furto di credenziali (es. *Mimikatz*, *PowerShell Empire*). Se i difensori sono in grado di rilevare lo strumento stesso e non solo un suo artefatto, l'attaccante deve trovarne uno nuovo o svilupparne uno da zero. Questo rappresenta una sfida significativa (*challenging*), poiché richiede tempo per lo sviluppo, per il test e per l'apprendimento del nuovo tool. [2]

6. **Tattiche, Tecniche e Procedure (TTPs) - Tough:** Al vertice della piramide si trovano le TTP le quali descrivono il comportamento dell'avversario: come opera, le sue strategie ed i suoi metodi, dalla ricognizione all'esfiltrazione dei dati. Rilevare e bloccare un attacco a questo livello è "duro" (*tough*) per l'avversario. Se i difensori riescono a neutralizzare una specifica tecnica, costringono l'attaccante a reinventare il proprio intero *modus operandi*: è a questo livello che l'intelligence diventa veramente strategica e operativa. [2]

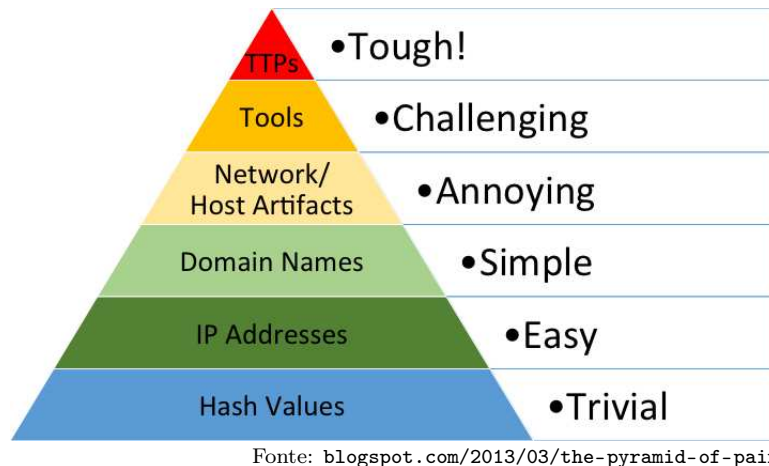


Figura 2.3: The Pyramid of Pain

Questa gerarchia evidenzia una dicotomia fondamentale per il sistema proposto in questa tesi: l'estrazione automatica degli indicatori alla base della piramide (Hash, IP, Domini) è un compito che può essere gestito con tecniche deterministiche e pattern matching, mentre l'identificazione e la corretta classificazione degli elementi al vertice (TTP) richiedono capacità di comprensione semantica avanzata, rendendo l'uso dei Large Language Models (LLM) indispensabile per un'analisi automatica, completa ed efficace.

### 2.1.5 Il framework MITRE ATT&CK: Una tassonomia globale del comportamento avversario

Per descrivere e categorizzare in modo standardizzato le TTP (il vertice della *Pyramid of Pain*), il framework **MITRE ATT&CK** (*Adversarial Tactics, Techniques, and Common Knowledge*) è emerso come lo standard *de facto* a livello globale. ATT&CK è una *knowledge base* curata ed accessibile pubblicamente, basata su osservazioni del mondo reale, che cataloga le azioni degli avversari durante l'intero ciclo di vita di un attacco informatico. [2][12]

Il framework fornisce un lessico comune per la comunità della cybersecurity ed è organizzato in matrici strutturate su due livelli principali:

1. **Tattiche (Tactics):** Rappresentano l'obiettivo tattico di alto livello, il "perché" di un'azione (es. *Initial Access*, *Persistence*). Nelle matrici le tattiche sono posizionate nelle colonne.

2. **Tecniche (Techniques):** Descrivono il "come" un avversario raggiunge un obiettivo tattico (es. *Phishing* come tecnica per ottenere *Initial Access*). Nelle matrici queste sono rappresentate dalle celle all'interno di ogni colonna.

La relazione tra STIX e ATT&CK è simbiotica: ATT&CK fornisce la tassonomia standardizzata per le TTP, mentre STIX, attraverso l'oggetto SDO *Attack-Pattern*, fornisce il linguaggio strutturato per rappresentare queste TTP e collegarle, attraverso le *Relationship*, ad altri elementi di intelligence (come *Threat-Actor*, *Malware* e *Indicator*) all'interno di un contesto più ampio. [5][10]

The image shows a screenshot of the MITRE ATT&CK matrix interface. It displays a grid of attack techniques organized into columns. The columns are: Initial Access (9 techniques), Execution (10 techniques), Persistence (18 techniques), Privilege Escalation (12 techniques), Defense Evasion (37 techniques), Credential Access (14 techniques), Discovery (25 techniques), Lateral Movement (9 techniques), and Collection (17 techniques). Each column contains a list of specific attack techniques, such as 'Replication Through Removable Media', 'Windows Management Instrumentation', 'BITS Jobs', 'Process Injection', 'Obfuscated Files or Information', 'Credentials from Password Stores', 'System Information Discovery', 'Replication Through Removable Media', 'Screen Capture', etc. The interface also includes search and filter controls at the top.

Fonte: [attack.mitre.org](https://attack.mitre.org)

Figura 2.4: MITRE ATT&CK matrix example

## 2.2 Lo standard STIX 2.1 in dettaglio

Per superare la frammentazione e l'ambiguità intrinseca dei report non strutturati, la comunità internazionale della cybersecurity è conversta verso l'adozione dello standard **STIX** (*Structured Threat Information Expression*). Originariamente sviluppato da MITRE e ora mantenuto dal comitato tecnico **OASIS Cyber Threat Intelligence (CTI) TC**, STIX è un linguaggio formale e un formato di serializzazione progettato per l'acquisizione, la caratterizzazione e lo scambio di Cyber Threat Intelligence. [7][10]

La versione **2.1**, che costituisce il riferimento tecnico per questo lavoro di tesi, introduce miglioramenti sostanziali rispetto alle iterazioni precedenti, consolidando l'uso di **JSON** come formato di serializzazione obbligatorio. Questa scelta garantisce un equilibrio ottimale tra leggibilità umana ed efficienza di parsing delle moderne applicazioni web e delle pipeline di analisi dati. [7][10]

### 2.2.1 Architettura e approccio: Il modello a grafo connesso

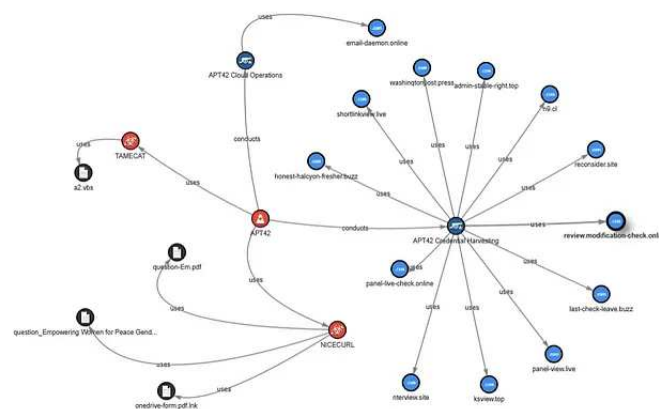
L'approccio architetturale di STIX 2.1 si distacca dai modelli gerarchici o tabulari tradizionali per abbracciare un **modello a grafo connesso**. In questa rappresentazione, l'intelligence

non è vista come una serie di eventi isolati, ma come una rete di entità interdipendenti. [7][10]

- **I nodi:** Sono costituiti dagli oggetti che rappresentano le entità del dominio cyber (es. attori, malware, vulnerabilità, o osservabili tecnici come indirizzi IP).
- **Gli archi:** Sono costituiti dalle relazioni che collegano questi nodi definendo la semantica dell'interazione (es. "attribuzione", "bersaglio", "utilizzo").

Questo approccio grafo-centrico è fondamentale per l'analisi moderna delle minacce perché permette di:

1. **Catturare il contesto:** Un indicatore (es. hash di file) non è solo un dato tecnico, ma diventa intelligence quando è collegato a un malware che a sua volta può essere collegato a un *Threat Actor*.
2. **Supportare l'analisi *pivoting*:** Gli analisti possono navigare il grafo spostandosi da un nodo all'altro per scoprire connessioni non evidenti (es. partendo da una vittima per risalire alla campagna e poi all'attaccante).
3. **Modularità:** Le informazioni possono essere condivise in frammenti (oggetti singoli) o come interi grafi (*Bundle*), facilitando l'aggiornamento incrementale della concorrenza.



Fonte: [medium.com/@antonio.formato](https://medium.com/@antonio.formato)

Figura 2.5: Esempio di struttura a grafo STIX

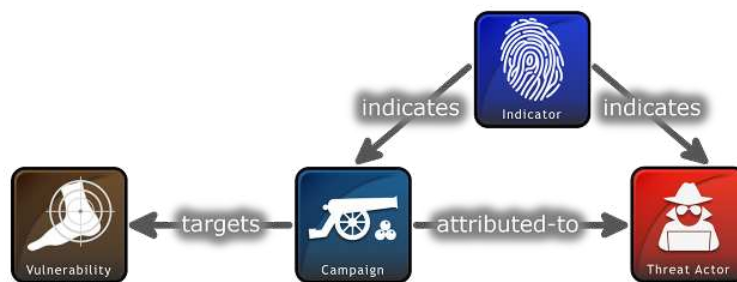
## 2.2.2 I componenti del modello dei dati: SDO, SCO e SRO

Il modello dei dati di STIX 2.1 è rigorosamente tipizzato e si articola in tre categorie principali di oggetti ("**STIX Objects**") ciascuna con un ruolo semantico distinto. [7][10]

1. **STIX Domain Objects (SDO) - L'intelligence di alto livello:** Gli SDO rappresentano i concetti astratti e di "alto livello" dell'intelligence, descrivono i comportamenti, gli attori, gli intenti e le strategie. Tra i 18 SDO definiti, quelli critici nella modellazione delle minacce includono:

- **Attack Pattern:** Descrive una Tattica, Tecnica o Procedura (TTP) utilizzata dagli avversari per compromettere i bersagli ed è l'oggetto primario per l'interoperabilità con il framework MITRE ATT&CK. [7]
    - Proprietà Chiave: `name`, `external_references` (per il linking a CAPEC/ATT&CK), `kill_chain_pases` (per posizionare l'attacco nella Cyber Kill Chain).
  - **Campaign:** Raggruppa una serie di attività malevole condotte in un periodo di tempo specifico con obiettivi comuni, spesso dirette contro uno specifico settore o area geografica. [7]
    - Proprietà Chiave: `objective` (l'intento strategico), `aliases`, `first_seen`, `last_seen`.
  - **Threat Actor:** Caratterizza gli individui, i gruppi o le organizzazioni che operano con intenti malevoli e permette di modellare non solo l'identità, ma anche le risorse, le motivazioni e il livello di sofisticazione dell'avversario. [7]
    - Proprietà Chiave: `threat_actor_types` (es. "nation-state", "crime-syndicate"), `goals`, `sophistication`.
  - **Malware:** Dettaglia un software malevolo inserito in un sistema per comprometterne la riservatezza, integrità o disponibilità. STIX distingue tra "famiglie" di malware e singole "istanze". [7]
    - Proprietà Chiave: `is_family` (booleano discriminante), `malware_types` (es. "ransomware", "rootkit"), `architecture_execution_envs`.
  - **Indicator:** Contiene un pattern tecnico utilizzato per rilevare attività sospette. A differenza degli altri SDO, l'indicator è un oggetto "attuabile" che contiene regole logiche (es. in linguaggio STIX Patterning, YARA o Snort) per il rilevamento automatico. [7]
    - Proprietà Chiave: `description`, `indicator_types`, `pattern`.
2. **STIX Cyber-observable Objects (SCO) - Gli elementi tecnici:** Gli SCO rappresentano gli elementi tecnici osservati a livello di host o di rete. Sono le rappresentazioni digitali concrete di entità cibernetiche e costituiscono i "dati grezzi" su cui si basa l'intelligence. A differenza degli SDO, che rispondono al "Chi" e al "Perché", gli SCO rispondono al "Cosa". Esempi includono: [7] [10]
- **File:** Caratterizzato da hash (MD5, SHA256), nome, dimensione e directory padre.
  - **Network Traffic:** Rappresenta il traffico di rete, includendo protocolli, porte e riferimenti al payload.
  - **IPv4-Addr / IPv6-Addr:** Rappresentano specifici indirizzi IP o blocchi CIDR.
  - **Domain-Name:** Rappresentano i nomi di dominio DNS.
3. **STIX Relationship Objects (SRO) - Il tessuto connettivo:** Gli SRO sono gli archi del grafo che collegano SDO e SCO trasformando una lista di oggetti isolati in una narrazione coerente. Esistono due tipi di SRO: [7][10]

- **Relationship:** Il connettore generico utilizzato per collegare due oggetti qualsiasi. È definito dalla tripla: *Source* (Sorgente), *Target* (Destinazione) e *Relationship Type* (Tipo di relazione). [7][10]
  - *Esempi:* **Threat-Actor** (source) *-uses->* **Malware** (target); **Campaign** (source) *-targets->* **Identity** (target).
- **Sighting:** Un tipo speciale di relazione che denota l'evidenza empirica ("ho visto questo"), collega un oggetto di intelligence (es. un **Indicator** o un **Threat Actor**) a un'osservazione concreta nel mondo reale (spesso rappresentata da **Observed Data**) fornendo la conferma temporale e contestuale della presenza di una minaccia. [7][10]



Fonte: [oasis-open.github.io](https://oasis-open.github.io)

Figura 2.6: Esempio di relazioni STIX 2.1

### 2.2.3 Proprietà comuni e struttura

Per garantire la consistenza e la gestione del ciclo di vita dei dati all'interno di sistemi distribuiti, tutti gli oggetti STIX (SDO e SRO) ereditano un set di **Proprietà Comuni** (*Common Properties*). Le più rilevanti includono: [7][10]

- **type:** Definisce la categoria dell'oggetto (es. "attack-pattern"), essenziale per il parsing e la validazione dello schema
- **id:** Un identificatore univoco globale che funge da chiave primaria per l'oggetto all'interno del grafo. La struttura è **type--UUID** (es. **malware--31b940d4-6f7f...**).
- **created / modified:** Timestamp che gestiscono il versionamento e permettono ai sistemi di gestire aggiornamenti dell'intelligence (es. aggiungere un alias a un Threat Actor) mantenendo la coerenza storica.
- **created\_by\_ref:** Un riferimento all'identità (**Identity**) che ha generato l'oggetto, fondamentale per stabilire la provenienza e l'attendibilità del dato.
- **spec\_version:** Indica la versione dello standard utilizzata, garantendo la compatibilità tra sistemi che potrebbero supportare versioni differenti.
- **labels:** Un array di stringhe per etichettare l'oggetto con termini a vocabolario aperto, utile per categorizzazioni rapide non coperte dalle proprietà strutturate.

L'insieme di questi oggetti, quando deve essere trasmesso o condiviso, viene incapsulato in un oggetto contenitore chiamato **Bundle**. Il bundle non ha significato semantico proprio, gli oggetti al suo interno non sono necessariamente correlati solo perché sono nello stesso bundle, ma funge esclusivamente da meccanismo di incapsulamento e trasporto. [7][10]

## 2.3 Large Language Models (LLM)

L'evoluzione del *Natural Language Processing* ha subito un'accelerazione senza precedenti con l'avvento dei Large Language Models. Questi modelli hanno segnato un cambio di paradigma radicale: si è passato da sistemi specifici, addestrati per singoli compiti (come la classificazione o la traduzione), a **Modelli di Fondazione** (*Foundational Models*) generalisti. Un LLM è definibile come un sistema probabilistico complesso che, addestrato su una quantità massiva di dati, apprende la distribuzione statistica del linguaggio e sviluppa capacità emergenti di ragionamento, comprensione del contesto e generazione creativa. [13]

### 2.3.1 Architettura e addestramento: Dai Transformer al "System 2" Thinking

La spina dorsale degli LLM moderni è l'**architettura Transformer**, introdotta nel celebre lavoro "*Attention Is All You Need*" di **Vaswani et al.** La chiave di questa architettura è il meccanismo di **Self-Attention** che permette al modello di processare l'intera sequenza di input simultaneamente e di pesare la rilevanza di ogni parola rispetto alle altre, indipendentemente dalla loro distanza nel testo. Questo ha superato i limiti di "memoria" delle precedenti Reti Neurali Ricorrenti (RNN). [14]

Mentre il Transformer originale prevedeva un Encoder e un Decoder (per processare l'input e l'output), la maggior parte dei modelli generativi attuali utilizza un'architettura **Decoder-Only** autoregressiva ed ottimizzata per prevedere il token successivo in una sequenza, come formalizzato da **Radford et al.** [15]

Il ciclo di vita di un LLM si articola in due fasi critiche:

1. **Pre-training:** Il modello viene esposto a petabyte di dati non etichettati. In questa fase, come dimostrato da Brown et al. [16], emergono le capacità di *In-Context Learning*: il modello impara a svolgere nuovi compiti semplicemente osservando esempi nel prompt, senza modificare i propri pesi.
2. **Post-training e allineamento:** Per trasformare un previsore di testo in un assistente utile, si applicano tecniche come il *Supervised Fine-Tuning* (SFT) e il *Reinforcement Learning from Human Feedback* (RLHF) introdotte da **Ouyang et al.** [17]. Nelle iterazioni più recenti, questo processo si è evoluto per includere il *Reasoning Training*, che insegna ai modelli a "pensare" prima di rispondere (il cosiddetto "*System 2 thinking*"), migliorando drasticamente le prestazioni in compiti logici complessi come la generazione di bundle STIX.

### 2.3.2 Prompt Engineering: Programmare con il linguaggio naturale

Nel contesto di questa tesi, il **Prompt Engineering** non si riduce ad "impartire ordini" all'LLM, ma rappresenta la metodologia di programmazione del modello. Dato che gli LLM sono sensibili al contesto, la struttura del prompt determina la qualità dell'output, specialmente in compiti rigidi come la generazione di JSON STIX 2.1. [18]

Le tecniche di Prompt Engineering includono:

- **Zero-Shot vs. Few-Shot Prompting:**

- Lo *Zero-Shot* affida il compito al modello senza esempi. Sebbene sia utile per testare la conoscenza di base, spesso fallisce nel rispettare schemi complessi. [18]
- Il *Few-Shot Prompting* inserisce nel prompt coppie di esempi (Input, Output). Questa tecnica è fondamentale poiché fornisce al modello un "template implicito" della struttura JSON desiderata, riducendo drasticamente gli errori di sintassi. [16][18]

- **Chain-of-Thought (CoT):** Introdotta da **Wei et al.** [19], questa tecnica istruisce il modello a generare una catena di ragionamento intermedia prima della risposta finale (es. "Analizza prima le TTP, poi estrai gli IoC, infine genera il JSON"). Questa tecnica permette al modello di disambiguare relazioni complesse prima di impegnarsi nella generazione dell'output. [18]

- **Role Prompting:** Consiste nell'assegnare al modello un ruolo specifico (es. "Agisci come un analista CTI esperto"). Questo condiziona l'LLM verso una terminologia più tecnica e precisa, riducendo le allucinazioni generiche. [18]

- **Retrieval-Augmented Generation (RAG) come Prompting:** Sebbene RAG sia un'architettura, a livello di prompt implica l'iniezione dinamica di contesto (es. definizioni specifiche dello standard STIX) all'interno dell'input, permettendo al modello di generare output più precisi e strutturati. [20][18]

### 2.3.3 Parametri di inferenza e sfide intrinseche

Sebbene il *Prompt Engineering* definisca il "Cosa" il modello deve fare, i **parametri di inferenza** (*Hyperparameters*) definiscono il "Come" il modello seleziona le risposte nello spazio delle probabilità. Per un compito rigido come la generazione di bundle STIX 2.1, la comprensione e il controllo di questi parametri sono critici per bilanciare la creatività sintattica con la coerenza strutturale.

#### Gestione del determinismo: Temperature e Sampling

Gli LLM generano testo calcolando la probabilità di ogni possibile token successivo nel vocabolario, tuttavia il modo in cui questo viene effettivamente scelto dipende dalla strategia di campionamento (*decoding strategy*), regolata principalmente dal parametro **Temperature** ( $T$ ).

Come formalizzato nello studio di **Holtzman et al.** The Curious Case of Neural Text Degeneration: [21]

- **Alta temperatura** ( $T \rightarrow 1$ ): La distribuzione di probabilità viene "appiattita". Il modello ha maggiori possibilità di scegliere token meno probabili, e questo aumenta la "creatività" e la varietà lessicale, ma incrementa esponenzialmente il rischio di errori di sintassi (JSON malformati) o di deviazioni logiche.
- **Bassa temperatura** ( $T \rightarrow 0$ ): La distribuzione viene "appuntita" rendendo i token ad alta probabilità quasi certi di essere scelti. Questo approccio è indispensabile per compiti di estrazione strutturata dove la precisione dello schema è prioritaria rispetto alla varietà della risposta.

Spesso, la temperatura viene utilizzata in congiunzione con il **Nucleus Sampling (top-P)**, anch'esso introdotto da **Holtzman et al.** [21], che taglia la "coda" della distribuzione di probabilità, costringendo il modello a scegliere solo tra i token che cumulativamente raggiungono una soglia di probabilità  $P$ , eliminando opzioni assurde senza sacrificare completamente la varietà.

### Il problema delle allucinazioni

La sfida più significativa nell'applicazione degli LLM alla Cyber Threat Intelligence è il fenomeno delle **allucinazioni**. In ambito NLP, un'allucinazione è definita come la generazione di contenuto che è "*nonsensical or unfaithful to the provided source content*" (assurda o non fedele al contenuto della fonte fornita) (**Ji et al.**). [22]

Nel contesto della generazione di bundle STIX, le allucinazioni si manifestano in due forme critiche:

1. **Allucinazioni fattuali:** Il modello inventa oggetti non presenti nel report originale: questo è un rischio critico in un contesto simile poiché porterebbe alla creazione di Intelligence errata.
2. **Allucinazioni istruttive:** Il modello ignora le regole dello schema JSON inventando campi inesistenti o violando i vincoli di tipo.

Studi specifici sulla CTI, come quello di **Perri et al.** [23], hanno evidenziato come anche modelli avanzati possano avere allucinazioni in report complessi se non adeguatamente vincolati da una temperatura bassa e da prompt robusti.

### Variabilità e incoerenza

A causa della natura stocastica degli LLM, lo stesso prompt inviato due volte può generare due bundle STIX diversi. Sebbene abbassare la temperatura riduca questo fenomeno, non lo elimina del tutto. Questa variabilità richiede meccanismi di validazione *post-processing* (come **stix2-validator**) per garantire che l'output sia conforme allo standard indipendentemente dalle fluttuazioni interne del modello.

### 2.3.4 Panoramica dei modelli utilizzati: Evoluzione e differenze

Per testare il sistema di generazione STIX, sono state esaminate tre famiglie di modelli. La sperimentazione ha messo a confronto non solo famiglie diverse (OpenAI, Meta, Google), ma anche diverse versioni all'interno della stessa, per evidenziare come l'evoluzione architetturale impatti sulla qualità dell'intelligence estratta.

1. **Famiglia GPT (OpenAI) - GPT-4.1 e GPT-5:** Questa famiglia rappresenta lo standard di riferimento per i modelli proprietari (*closed-source*). [24]
  - **GPT-4.1:** Una versione più ottimizzata ed efficiente della famiglia GPT-4 con eccellente velocità di inferenza e costi ridotti. È un modello "veloce", ideale per compiti di estrazione massiva dove la latenza è critica, tuttavia mostra limiti nel ragionamento a lungo raggio su report molto complessi. [25]
  - **GPT-5:** Rappresenta il salto generazionale, introduce capacità di **ragionamento profondo**. A differenza di GPT-4.1, GPT-5 è in grado di pianificare la struttura del bundle STIX prima di generarlo (ragionando su di essa prima di produrre un output), riducendo quasi a zero gli errori di *referencing* (es. ID mancati). La sua *context window* è significativamente più ampia e la tendenza alle allucinazioni è drasticamente inferiore. [26]
2. **Famiglia Llama (Meta) - L'evoluzione Open Weights:** L'uso di modelli Open Weights potrebbe essere di fondamentale importanza in alcuni contesti per garantire la privacy dei dati. [27]
  - **Llama 3.3 70B:** Rappresenta il punto di riferimento per l'efficienza, è affidabile per l'estrazione di entità (NER), ma soffre nella gestione di contesti molto grandi e nella generazione di output complessi. [28]
  - **Llama 4 Maverick:** Rappresenta la nuova frontiera dei modelli aperti, utilizza un'architettura *Mixture-of-Experts*. Rispetto al modello 3.3 possiede capacità multimodali native (es. può leggere grafici all'interno di un report), inoltre la sua finestra di contesto è più estesa e la sua capacità di seguire istruzioni complesse lo rende paragonabile a GPT-5. [29]
3. **Famiglia Gemini (Google) - Gemini 2.5 Pro:** Rappresenta l'approccio "*Long Context*" migliore. Si distingue per la sua *Infinite Context Window*, questo gli permette di lavorare al meglio anche con report molto lunghi. Rispetto agli altri modelli mantiene meglio la coerenza su documenti estesi (permettendogli ad esempio di collegare due dati anche molto distanti tra di loro nel testo). [30]

## 2.4 Lavori correlati e stato dell'arte

L'automazione dell'estrazione di Cyber Threat Intelligence non è un problema nuovo, ma le metodologie per affrontarlo hanno subito un'evoluzione radicale negli ultimi anni. La lettera-

tura scientifica può essere, infatti, suddivisa in due ere distinte: l'era pre-LLM, dominata da espressioni regolari e NLP supervisionato, e l'era attuale guidata dalla *Generative AI*.

#### 2.4.1 Approcci tradizionali: Espressioni regolari e NLP classico

I primi tentativi di automatizzare la generazione di STIX si basavano su approcci deterministici. Strumenti come **TRAM (Threat Report ATT&CK Mapper)** del MITRE hanno aperto la strada, utilizzando il *pattern matching* per mappare il testo alle tecniche ATT&CK. Tuttavia, questi sistemi soffrivano di rigidità: un cambiamento nella formulazione di una frase spesso portava al fallimento dell'estrazione.

Un'evoluzione significativa è rappresentata da **STIXnet**, proposto da **Marchiori et al.** [5], che supera i limiti delle regole statiche integrando tecniche di NLP con una *Knowledge Base* interattiva. Pur ottenendo buoni risultati nell'estrazione di entità, l'approccio si basa ancora su componenti di estrazione specifiche e manca della flessibilità necessaria per interpretare relazioni semantiche implicite o nuove forme di minacce non presenti nella base di conoscenza.

#### 2.4.2 L'era della Generative AI: LLM per la CTI

Con l'avvento dei Large Language Models, l'attenzione si è spostata sulla generazione *end-to-end*. Uno dei primi lavori ad esplorare l'uso degli LLM è stato **AGIR** di **Perrina et al.** [3] dimostrando che gli LLM possono manipolare efficacemente il linguaggio tecnico della CTI. Sebbene AGIR si concentri sulla generazione del report testuale a partire da grafi, conferma la capacità dei modelli di comprendere la semantica del dominio.

#### 2.4.3 Gap di ricerca e posizionamento della tesi

Nonostante i progressi, la letteratura attuale presenta delle lacune che questa tesi intende colmare:

1. **Mancanza di analisi comparativa granulare:** Molti studi si focalizzano su un singolo modello e manca un confronto sistematico tra famiglie di modelli, specificamente sul task di generazione di bundle STIX 2.1.
2. **Il problema dell'affidabilità:** Come evidenziato da Perrina et al. [3], gli LLM rimangono "inaffidabili" se non adeguatamente vincolati. Esiste un bisogno critico di definire strategie di *Prompt Engineering* e parametri di inferenza (temperatura) che massimizzino il determinismo strutturale.
3. **Valutazione semantica vs. sintattica:** La maggior parte dei lavori valuta la correttezza del JSON (sintassi). Questo, invece, si propone di valutare anche la qualità semantica (l'intelligence estratta) confrontando l'output di diverse famiglie di modelli.

## Capitolo 3

# Progettazione e metodologia

Dopo aver analizzato il contesto e lo stato dell'arte, qui verrà definita la metodologia di ricerca adottata e l'architettura logica del sistema proposto per l'automazione di Cyber Threat Intelligence strutturata.

L'obiettivo di questo capitolo è illustrare come i requisiti teorici già discussi siano stati tradotti in una soluzione tecnica concreta e come quest'ultima sia stata valutata scientificamente, partendo dalla costruzione dei dati di test fino alla definizione delle metriche di valutazione.

In particolare verranno trattati i seguenti punti:

- **Architettura della soluzione:** Verrà descritta la pipeline modulare progettata che orchestra l'intero flusso di lavoro dall'ingestione del report grezzo alla generazione del bundle STIX 2.1.
- **Dataset e costruzione del Ground Truth:** Si discuterà il processo di selezione dei report di threat intelligence reali che abbiano un corrispettivo bundle STIX 2.1 associato e pubblicamente accessibile.
- **Approccio ibrido e Prompt Engineering:** Si giustificherà la scelta strategica di combinare metodologie deterministiche e probabilistiche, analizzando nel dettaglio le tecniche di istruzione del modello progettate per massimizzare la precisione.
- **Metodologia di valutazione:** Infine si definirà il protocollo sperimentale, descrivendo le metriche quantitative (sintattiche e semantiche) selezionate per misurare oggettivamente le performance del sistema rispetto al Ground Truth.

### 3.1 Architettura della soluzione: Il Workflow GenAI-STIX

L'architettura sviluppata, denominata GenAI-STIX-2.1-Generator, si configura come una pipeline sequenziale ibrida la soluzione non si affida esclusivamente alla generazione probabilistica dell'LLM per l'intero processo, ma orchestra moduli di codice deterministico (Python) con moduli di ragionamento semantico (LLM). Questa scelta architetturale mira a massimizzare l'accuratezza dei dati tecnici (IoC) pur mantenendo la flessibilità necessaria per interpretare le relazioni complesse.

Il flusso di lavoro è strutturato in cinque fasi:

1. **Data Ingestion & Normalization**
2. **Advanced IOC Extraction**
3. **Programmatic Object Generation**
4. **Comprehensive Entity Extraction**
5. **Final Bundling & Orchestration**

### 3.1.1 Data Ingestion & Normalization

La prima fase della pipeline gestisce l'acquisizione del report di Cyber Threat Intelligence. Poiché le fonti possono essere eterogenee (URL web, file PDF, file di testo), invece di passare testo grezzo, il sistema integra la libreria open-source **MarkItDown** [31].

In questo modo, indipendentemente dal file sorgente, il contenuto viene convertito in **Markdown** strutturato. A differenza del testo puro o dell'HTML "sporco", il Markdown preserva la struttura semantica del documento (titoli, elenchi puntati, tabelle, blocchi di codice). Questa strutturazione è fondamentale per l'LLM nelle fasi successive poiché permette al modello di distinguere le sezioni narrative (dove si trovano le TTP) dalle sezioni tecniche (dove si trovano gli IoC) e di ignorare il rumore (es. footer, banner pubblicitari). Inoltre i moderni LLM, come quelli usati per questo progetto, hanno dimostrato prestazioni superiori quando processano testo formattato in Markdown.

### 3.1.2 Advanced IOC Extraction

L'estrazione degli Indicatori di Compromissione (IoC) rappresenta la sfida critica per evitare falsi positivi. Il sistema implementa un approccio ibrido in due step che combina la velocità delle Regular Expressions con la comprensione contestuale dell'LLM:

1. **Pattern Matching deterministico:** Utilizzando la libreria **iocextract**, il sistema scansiona l'intero corpo del report per individuare pattern sintattici corrispondenti a indirizzi IPv4, IPv6, URL, Domini ed Hash (MD5, SHA-1, SHA-256): questa fase garantisce di catturare tutto ciò che sembra un IoC.
2. **Validazione contestuale via LLM:** I candidati estratti vengono passati all'LLM con un prompt specifico. Il modello analizza il contesto in cui l'IoC appare nel report per determinare se è una minaccia reale o un falso positivo contestuale (ad esempio un indirizzo IP di Google DNS 8.8.8.8 citato come esempio oppure un numero di versione software 1.2.3.4 scambiato per un IP). Solo gli IoC validati semanticamente passano alla fase successiva.

### 3.1.3 Programmatic Object Generation

La creazione degli oggetti **STIX Cyber-observable Objects (SCO)** non è generativa ma **programmatica**. Per gli IoC validati, il sistema istanzia direttamente le classi Python della libreria `stix2` [32]. Per questi oggetti il sistema genera degli **UUIDv5** basati sul valore stesso dell'indicatore.

Per ogni SCO, il sistema genera automaticamente un corrispondente oggetto SDO **Indicator**. Viene inoltre creata una relazione di tipo **derived-from** che collega l'indicatore (la regola di rilevamento) allo SCO (il fatto osservato) creando una struttura dati coerente.

### 3.1.4 Comprehensive Entity Extraction

In questa fase il report Markdown viene inviato all'LLM per l'estrazione della "logica della minaccia" e l'LLM opera come un analista CTI esperto per identificare gli **STIX Domain Objects (SDO)** di alto livello.

Il prompt è ingegnerizzato per estrarre e strutturare in JSON le seguenti entità:

- **Threat Actor:** Gruppi o individui responsabili.
- **Malware / Tool:** Software malevolo o strumenti utilizzati.
- **Attak Pattern:** Tattiche e Tecniche mappate sul framework MITRE ATT&CK.
- **Identity:** Vittime o bersagli dell'attacco.
- **Vulnerability:** Riferimenti a CVE sfruttati.

In questa fase l'LLM definisce anche le relazioni semantiche tra queste entità (es. **Threat Actor** – *uses* –> **Malware**), costruendo lo scheletro narrativo del grafo.

### 3.1.5 Final Bundling & Orchestration

Nell'ultima fase avviene l'assemblaggio del bundle finale. Il modulo di orchestrazione si occupa di unire tutti i dati:

- Aggrega gli SCO generati da codice.
- Aggrega gli SDO (oggetti semantici) creati dall'LLM.
- Tutti gli oggetti vengono incapsulati in un oggetto bundle STIX 2.1.
- Il bundle viene serializzato in un file JSON finale pronto per essere visualizzato.

## 3.2 Dataset e costruzione del Ground truth

Uno degli ostacoli più significativi riscontrati nella ricerca sull'automazione della Cyber Threat Intelligence è la scarsità di dataset pubblici di alta qualità che forniscano una mappatura tra report testuali non strutturati e i corrispondenti file strutturati in standard STIX 2.1.

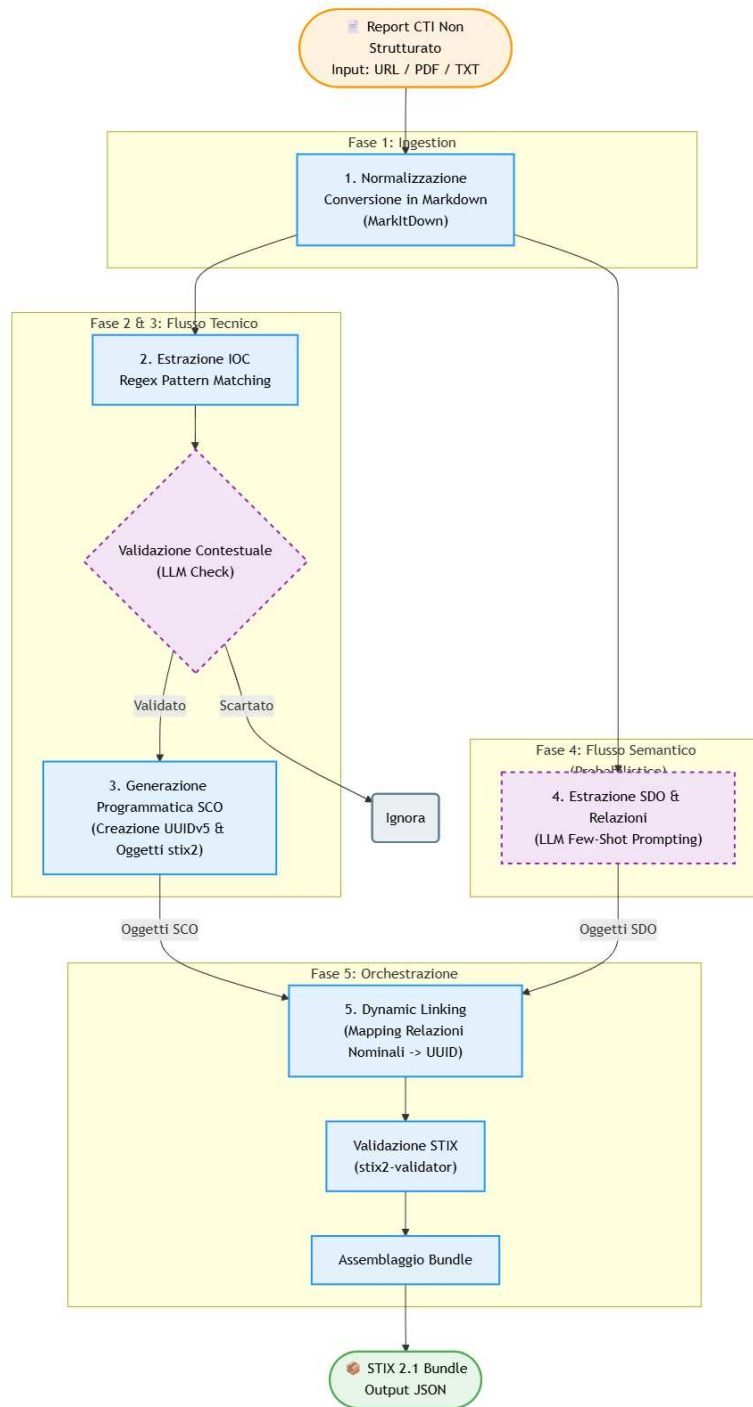


Figura 3.1: Diagramma di flusso del tool di generazione

Sebbene i report in formato PDF siano di uso comune, la quasi totalità dei vendor di sicurezza distribuisce gli indicatori tecnici (IoC) in formati tabulari disgiunti dalla narrazione, senza fornire il grafo STIX completo delle relazioni o usando altri standard come YARA. Questa mancanza ha reso ardua la ricerca di un dataset, per la creazione di un *Ground Truth*, che sia curato e affidabile.

### 3.2.1 Strategia di ricerca e difficoltà di reperimento

L'obiettivo era reperire fonti "ufficiali" che pubblicassero nativamente sia l'analisi testuale (Report) che il file strutturato (Bundle STIX 2.1), per garantire che la "verità" con cui valutare l'LLM fosse stata generata da analisti umani esperti.

La ricerca ha evidenziato una frammentazione estrema nell'adozione dello standard: molte fonti, ad esempio, pubblicano file incompleti, sintatticamente obsoleti (STIX 1.x) o con strutture anomale.

### 3.2.2 Fonte primaria selezionata: NCSC Malware Reports

Dopo un'analisi approfondita, l'unica fonte che ha soddisfatto i requisiti di qualità e completezza strutturale è stata la sezione **Malware Analysis Reports** del **National Cyber Security Center (NCSC)** del Regno Unito. [33]

L'NCSC pubblica analisi tecniche dettagliate di campagne malware accompagnate da file STIX che mappano fedelmente le entità descritte, e tra cui sono stati selezionati 20 report rappresentativi di diverse tipologie di minaccia e che potessero rappresentare una buona varietà in termini di lunghezza e struttura dei report.

Il dataset è stato strutturato in un *repository* organizzato in 20 sottocartelle, ognuna della quale contiene la coppia rigorosa: report testuale (PDF) e bundle STIX 2.1 corrispondente fornito dall'NCSC (JSON).

### 3.2.3 Criteri di esclusione: Il caso CISA

Durante la fase di selezione, è stata analizzata anche la pagina dei **Cybersecurity Alerts & Advisories** della **CISA (Cybersecurity and Infrastructure Security Agency)** degli Stati Uniti [34]. Nonostante l'autorevolezza della fonte, i report STIX forniti dalla CISA sono stati esclusi dal dataset finale per motivi di incompatibilità strutturale.

L'analisi preliminare dei bundle CISA ha rivelato un'anomalia nella topologia del grafo: gli oggetti sembravano strutturati attorno a nodi di tipo **marking-definition** (di norma utilizzati per la gestione dei permessi TLP e del copyright) che agivano come "hub" centrali a cui erano collegate tutte le entità. Questa struttura differisce dall'idea standard di STIX 2.1 (orientata alle relazioni semantiche tra Attore, Malware e Vittima) e da quella generata dal sistema proposto in questa tesi. Includere questi report avrebbe introdotto un *bias* negativo nella valutazione: il sistema avrebbe generato un grafo delle minacce corretto, ma le metriche di similarità lo avrebbero penalizzato pesantemente poiché non replicava la connettività verso i nodi di metadati presenti nei file CISA.

La decisione di escludere questa fonte evidenzia l'importanza di avere un *Ground Truth* che sia non solo valido sintatticamente, ma anche rappresentativo della logica di Threat Intelligence che si intende automatizzare.

### 3.3 Prompt Engineering e strategie di istruzione

L'architettura ibrida descritta in precedenza affida al Large Language Model il compito più complesso cioè l'interpretazione semantica del report e la strutturazione della logica della minaccia, tuttavia, i modelli generativi *general-purpose* non nascono con la capacità nativa di generare JSON STIX 2.1 validi.

Per colmare questo divario tra la natura probabilistica del modello e la rigidità dello schema STIX, è stata implementata una strategia di **Prompt Engineering composita**. Non ci si è limitati a fornire istruzioni in linguaggio naturale (*Zero-Shot*), ma si è strutturato il prompt su tre livelli: **Persona (Role)**, **Context (Few-Shot)** e **Constraint (Format)**.

#### 3.3.1 Role Prompting: Definizione della "persona"

Il primo livello di istruzione definisce il contesto operativo del modello. Viene assegnata all'LLM una "persona" specifica per indirizzarlo verso una terminologia di sicurezza corretta e inibire la creatività narrativa a favore del rigore tecnico.

Come visibile nel codice, il *System Prompt* istruisce il modello ad agire come un esperto di CTI specializzato nello standard OASIS STIX 2.1:

```
1 prompt = f"""
2     As a senior CTI analyst, your task is to meticulously identify and
3     classify all distinct entities within the provided threat report that
4     correspond to STIX Domain Object types.
5     """
```

Listing 3.1: Role Prompting

In questo modo il modello sarà forzato ad usare una terminologia tecnica ed impostare il suo contesto su quello di uno specialista CTI.

#### 3.3.2 Few-Shot Learning: Apprendimento per analogia

Per garantire che il modello rispetti la complessa sintassi STIX (es. l'uso corretto di `type`, la formattazione dei timestamp, la struttura delle `kill_chain_phases`) è stata adottata la tecnica **Few-Shot Prompting**: nel prompt vengono iniettati esempi concreti di output atteso che mostrano al modello come mappare linguaggio naturale in oggetti strutturati.

Di seguito un estratto esemplificativo del *Few-Shot Prompting* utilizzato nel sistema:

```
1 prompt = f"""
2     Example of a final object in the array (FOR FORMATTING REFERENCE ONLY):
3     {{
4         "name": "Name-Technique-Example",
5         "type": "attack-pattern",
6         "description": "Description of how the sample malware uses this
7         technique...",
8         "external_id": "TXXXX",
9         "kill_chain_phases": [
10            {{
11                "kill_chain_name": "mitre-attack",
```

```

11         "phase_name": "tactic-name-example"
12     }}
13 ]
14 }}
15 Formatting example for YARA rule
16 {{
17     "name": "YARA_RULE_EXAMPLE_NAME",
18     "type": "yara-rule",
19     "pattern": "rule YARA_RULE_EXAMPLE_NAME {{ meta: ... strings: ...
condition: ... }}",
20     "indicates_malware": "nome-malware-esempio",
21     "associated_hashes": ["hash_sha256_del_file_di_esempio"]
22 }}
23 ""

```

Listing 3.2: Few-Shot Prompting

L'inclusione di questi *shot* (esempi) nel contesto ha ridotto drasticamente il tasso di errore sintattico, permettendo al modello di replicare la struttura desiderata per analogia.

### 3.3.3 Chain-of-Thought e scomposizione del task

Per affrontare la complessità dell'estrazione di oggetti e relazioni, il *Prompt* include istruzioni sequenziali che includano una **Chain-of-Thought (CoT)** implicita. Invece di chiedere l'intero grafo in un solo passaggio non strutturato, le istruzioni guidano il modello a processare le informazioni per categorie logiche sequenziali:

```

1 prompt = f"""
2     Instructions:
3     1. Read the entire report to understand the context.
4     2. For the **Malware** object, you MUST extract:
5         - Its `name`.
6         - Its `type` as "malware".
7         - A detailed `description`.
8         - A `malware_types` array, inferring the type from this list: ["
remote-access-trojan", "backdoor", "downloader", "spyware", "ransomware"].
9     3. For any **Identity** or **Tool** objects mentioned (e.g., NCSC, Trend
Micro, PwC, VMware), extract their `name`, `type`, and a concise `
description` of their role in the report.
10    4. **Author Identification**: Identify the primary organization that
authored or published this report (e.g., Cisco Talos, Mandiant, NCSC) and
extract it as an 'identity' object.
11    5. For any **Threat-Actor** (e.g., APT groups, specific threat actors),
you MUST extract:
12        - Its `name` and any known `aliases`.
13        - Its `type` as "threat-actor".
14        - A detailed `description` of its goals, motivations, or relevant
TTPs mentioned in the report.
15    """

```

Listing 3.3: Chain-of-Thought implicita

Questa scomposizione riduce il carico cognitivo sul modello e migliora la coerenza degli oggetti estratti.

### 3.3.4 Vincoli di formato e JSON Mode

Infine, sono stati impostati vincoli rigidi sul formato di output. Sfruttando le potenzialità **JSON Mode** delle API OpenAI, il modello è forzato a generare esclusivamente un oggetto JSON valido:

```
1 response_format={"type": "json_object"}
```

Listing 3.4: Vincoli di Formato

Questa combinazione di tecniche trasforma l'LLM da un semplice generatore di testo creativo a un generatore di JSON STIX 2.1 affidabile e capace di produrre output strutturati.

## 3.4 Metodologia di valutazione e protocollo sperimentale

La valutazione di sistemi generativi che producono output strutturati complessi, come grafi STIX, pone delle sfide uniche che rendono inadeguate le metriche tradizionalmente usate per gli NLP. Un file STIX è valido non se sembra "ben scritto", ma se contiene gli oggetti corretti con le proprietà corrette.

Per questo motivo è stato sviluppato un framework di valutazione implementato tramite due pipeline di analisi distinte che operano sul confronto tra il **Bundle Generato** ( $B_{gen}$ ) e il **Bundle di Ground Truth** ( $B_{gt}$ ) fornito dall'NCSC.

### 3.4.1 Il problema dell'allineamento

La sfida preliminare affrontata è l'impossibilità di confrontare gli oggetti basandosi sul loro ID. **Ground Truth** utilizza UUIDv4 generati dagli analisti NCSC, mentre quello **Generato** utilizza nuovi UUIDv4.

Poiché  $ID(gen) \neq ID(gt)$ , il software di valutazione implementa una logica di **Content-Based Matching**: prima di calcolare qualsiasi metrica, il sistema tenta di allineare gli oggetti dei due bundle confrontando le loro proprietà distintive e per farlo usa delle funzioni `object_similarity` e `object_equivalence` della libreria `stix2`. Solo quando un match viene stabilito è possibile valutare la qualità della generazione.

### 3.4.2 Pipeline 1: Valutazione semantica

Qui la valutazione è progettata per misurare le capacità del sistema di **rilevare ed estrarre** le entità presenti nel report.

Non viene effettuato un confronto testuale ma **Semantico**: cerca di capire se il **significato** degli oggetti è lo stesso.

Il confronto avviene in 4 fasi:

1. **Fase di preparazione e pulizia**

2. **Logica di confronto**
3. **Classificazione dei risultati**
4. **Calcolo delle metriche**

Vediamole nel dettaglio.

#### **Fase di preparazione e pulizia**

Prima del confronto i dati vengono puliti e organizzati, cioè validati in accordo allo standard STIX 2.1. Gli oggetti sono divisi in due gruppi SDO/SCO e SRO e vengono rimossi oggetti di metadati che andrebbero a sporcare inutilmente il confronto come **report** e **marking-definition**, con le relative relazioni che li legano agli altri oggetti.

#### **Logica di confronto**

Questa è la parte più complessa nella quale il confronto avviene in due modi diversi a seconda del tipo di oggetto.

1. **Confronto SDO e SCO:** Il sistema prende un oggetto generato dal tool e cerca il suo corrispondente nel Ground Truth:
  - Per gli **Indicatori** usa una logica personalizzata molto specifica:
    - Se la proprietà **pattern** è di tipo **STIX patterning** il confronto avviene come tutti gli altri oggetti.
    - Se la proprietà **pattern** presenta una regola YARA invece, viene eseguita una pulizia della stringa. Rimuove i commenti, normalizza gli spazi e, tramite una regex, standardizza il contenuto dentro le parentesi graffe, le due stringhe (quella generata e quella del Ground Truth) vengono poi semplicemente confrontate.
  - Per tutti gli **Altri Oggetti**: Usa la funzione **object\_equivalence** della libreria **stix2** che confronta le proprietà chiave di un oggetto. Se la similarità supera la soglia impostata (**SIMILARITY\_THRESHOLD**), i due oggetti sono considerati uguali.
2. **Confronto SRO:** Le relazioni sono più difficili da confrontare e non sono supportate dalla libreria **stix2**. Per il confronto, in questo caso, viene usato un sistema a **punteggio ponderato**.

Per dire che una relazione generata dal tool sia corretta, il sistema controlla tre variabili, assegnando un punteggio a ciascuna:

  - (a) **Source Ref (40 punti):** Controlla che l'oggetto di partenza generato dall'LLM sia lo stesso del *Ground Truth* e per farlo usa una mappa creata durante la fase di confronto degli SDO/SCO.
  - (b) **Target Ref (40 punti):** Controlla che l'oggetto di destinazione generato dall'LLM sia lo stesso del *Ground Truth*.

- (c) **Relationship Type (20 punti):** Se i due oggetti corrispondono controlla, infine, che la relazione tra i due sia dello stesso tipo.

Se la somma dei punti è maggiore o uguale alla soglia ( RELATIOSHIP\_SIMILARITY\_THRESHOLD di default impostata su 80), la relazione è considerata corretta.

### Classificazione dei risultati (TP, FP, FN)

Alla fine dei cicli di confronto ogni oggetto viene etichettato:

- **True Positive (TP):** L'oggetto generato dal tool ha una corrispondenza con un oggetto nel *Ground Truth*.
- **False Positive (FP):** L'oggetto generato dal tool **non** ha una corrispondenza nel *Ground Truth*: questa è di fatto un'**allucinazione**.
- **False Negative (FN):** L'oggetto del *Ground Truth* che il tool **non** ha generato.

### Calcolo delle metriche

In fine vengono calcolate delle metriche statistiche basandosi sui TP, FP e FN:

- **Precision:** Questo parametro risponde alla domanda: “Di tutti gli elementi STIX generati dal mio strumento, quale percentuale era effettivamente corretta?” Un punteggio di accuratezza elevato indica che lo strumento è affidabile e non genera molto “rumore” o informazioni errate. È una misura della qualità dell'output.

$$- Precision = \frac{TP}{TP+FP}$$

- **Recall:** Questa metrica risponde alla domanda: “Di tutti gli elementi STIX corretti presenti nel report, quale percentuale è riuscito a trovare il mio strumento?” Un punteggio di recall elevato indica che lo strumento è completo e non omette molte informazioni rilevanti. È una misura della copertura.

$$- Recall = \frac{TP}{TP+FN}$$

- **F1-Score:** Il punteggio F1 è la media armonica di precisione e richiamo, fornisce un unico punteggio bilanciato che tiene conto di entrambi gli aspetti. Questo è particolarmente utile quando si desidera ottenere prestazioni equilibrate, ovvero quando è altrettanto importante ridurre al minimo il rumore (alta precisione) e massimizzare la copertura (alto richiamo). Il punteggio F1 penalizza fortemente i sistemi che eccellono in una metrica a discapito dell'altra.

$$- F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Queste metriche vengono calcolate in tre modi:

- **Micro Averaging:** La Micro Averaging attribuisce lo stesso peso a ciascuna decisione di classificazione individuale su un'istanza di oggetto. Risponde alla domanda: “Considerando tutti gli oggetti STIX estratti da tutti i report, quale percentuale di istanze

individuali è stata gestita correttamente?”. Questo punteggio sarà dominato dalle prestazioni sulle classi più numerose: se lo strumento è molto efficace nell'estrazione di indicatori e indirizzi IPv4 (che sono molto comuni), il punteggio Micro-F1 sarà elevato anche se le prestazioni sulle classi rare ma importanti, come Campagna o Attore della minaccia, sono scarse. In un contesto multi-classe, la precisione *Micro Averaging* è uguale al richiamo *Micro Averaging* e quindi al punteggio F1 micro-medio e all'accuratezza complessiva.

– Micro Averaging:

$$\begin{aligned} * Precision_{micro} &= \frac{\sum_{i=1}^c TP_i}{\sum_{i=1}^c (TP_i + FP_i)} \\ * Recall_{micro} &= \frac{\sum_{i=1}^c TP_i}{\sum_{i=1}^c (TP_i + FN_i)} \\ * F1_{micro} &= 2 \cdot \frac{Precision_{micro} \cdot Recall_{micro}}{Precision_{micro} + Recall_{micro}} \end{aligned}$$

- **Macro Averaging:** La *Macro Averaging* attribuisce lo stesso peso a ciascuna classe indipendentemente dalla sua frequenza. Risponde alla domanda: “In media, come si comporta il mio strumento sui diversi tipi di concetti CTI?”. Questo punteggio considera il punteggio F1 per la classe Threat-Actor (che può basarsi solo su pochi casi) altrettanto importante quanto il punteggio F1 per la classe Indicator (basato su centinaia di casi). Si tratta di una misura molto più indicativa della versatilità dello strumento e della sua capacità di gestire anche i tipi di entità più rari e difficili, infatti se lo strumento ha prestazioni scadenti su una classe minoritaria, il punteggio *Macro-F1* ne risentirà in modo significativo.

– Macro Averaging:

$$\begin{aligned} * Precision_{macro} &= \frac{1}{c} \sum_{i=1}^c P_i \\ * Recall_{macro} &= \frac{1}{c} \sum_{i=1}^c R_i \\ * F1_{macro} &= \frac{1}{c} \sum_{i=1}^c F_i \end{aligned}$$

- **Weighted Averaging:** Questo metodo rappresenta un compromesso tra Micro e Macro: cerca di tenere conto dello squilibrio delle classi (come Micro) mentre calcola le metriche per ogni tipo (come Macro). In molti scenari il risultato della media ponderata sarà molto simile a quello della media Micro poiché entrambi finiscono per dare maggiore importanza alle classi con più istanze.

– Weighted Averaging:

$$\begin{aligned} * Precision_{weighted} &= \sum_{i=1}^c \omega_i \times P_i \\ * Recall_{weighted} &= \sum_{i=1}^c \omega_i \times R_i \\ * F1_{weighted} &= \sum_{i=1}^c \omega_i \times F1_i \end{aligned}$$

### 3.4.3 Pipeline 2: Graph Evaluation

L'obiettivo di questa pipeline è ottenere un **punteggio percentuale** che rappresenti quanto il report generato è simile a quello del Ground Truth, considerando non solo i singoli oggetti ma il grafo nella sua interezza.

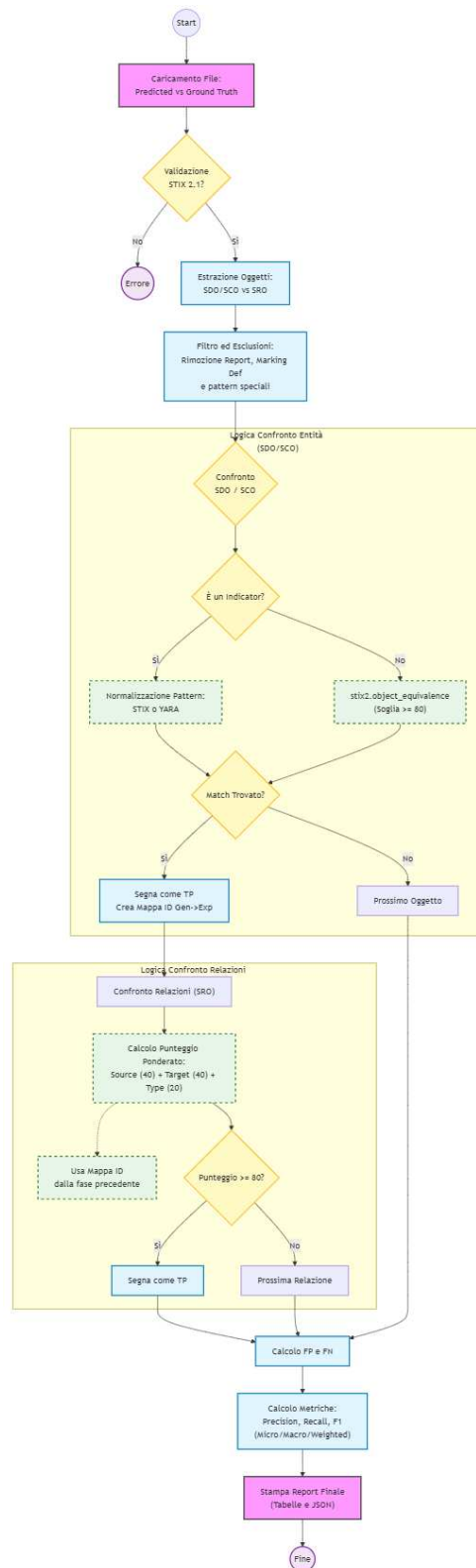


Figura 3.2: Diagramma di flusso della valutazione semantica

Il flusso di lavoro avviene in 5 fasi:

- **Caricamento e validazione:** I file JSON vengono caricati e si controlla che siano

conformi allo standard STIX 2.1.

- **Pre-Processing:** vengono estratti gli oggetti e vengono rimossi dal confronto gli `indicator` che hanno la proprietà `pattern_type` di `pattern` diversa da `stix patterning`. Questo viene fatto perché il parser semantico della libreria `stix2` non supporta altri formati e dunque andrebbe a invalidare il confronto.
- **Creazione `MemoryStore`:** Per confrontare i grafi, la libreria `stix2` richiede che gli oggetti siano caricati in un `MemoryStore` e non in una semplice lista.
- **Confronto grafo:** Viene eseguita la funzione `graph_similarity`. Questa funzione, fornita dalla libreria `stix2`, analizza la topologia del grafo e il contenuto semantico. Restituisce un punteggio globale e popola un dizionario (`prop_scores`) con i dettagli per ogni nodo.
- **Analisi con `Pandas`:** I risultati grezzi vengono trasformati in un `DataFrame Pandas`. Vengono calcolate statistiche descrittive (come media e percentili) e fatti raggruppamenti per tipo di oggetto per valutare dove il tool performa meglio.

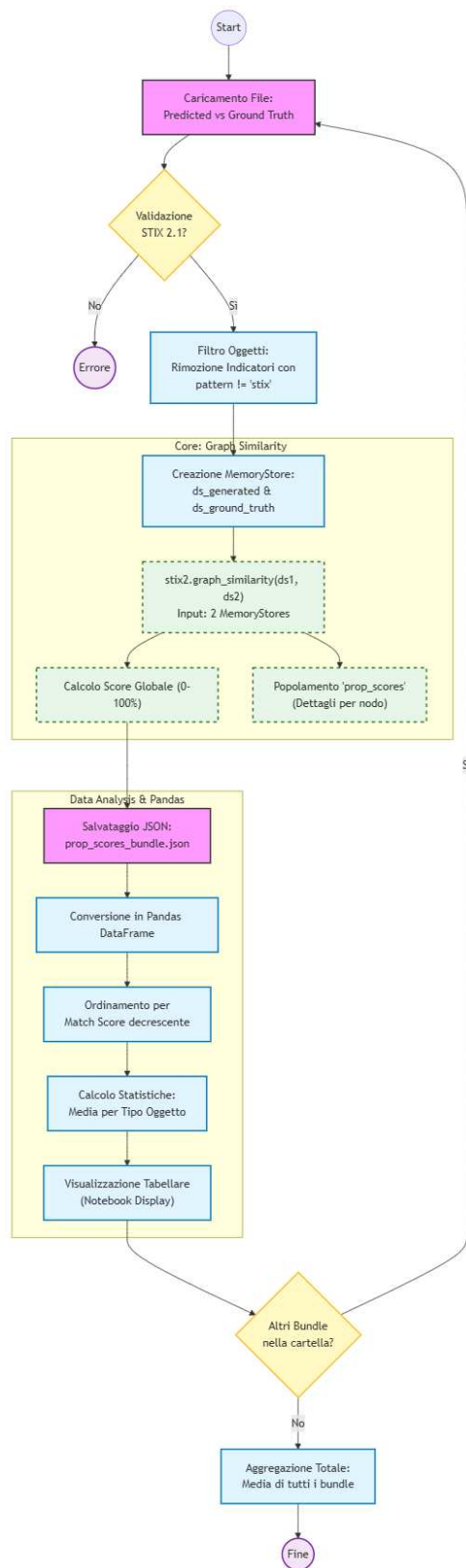


Figura 3.3: Diagramma di flusso della Graph Evaluation

## Capitolo 4

# Implementazione

Dopo aver definito l'architettura logica e la metodologia di ricerca nel capitolo precedente, questo capitolo scende nel dettaglio tecnico della realizzazione del sistema *GenAI-STIX* 2.1 *Generator*. Verranno descritti l'ambiente di sviluppo, lo stack tecnologico e le soluzioni implementative adottate per tradurre i requisiti teorici in codice funzionale.

L'obiettivo è fornire una visione complessiva del lavoro svolto, analizzando come sono state usate le varie librerie.

In particolare verranno trattati i seguenti punti:

- **Stack tecnologico e ambiente di sviluppo:** Si presenteranno le tecnologie utilizzate, giustificando la scelta del linguaggio Python, delle librerie utilizzate per la manipolazione dei dati STIX (`stix2`), l'interazione con gli LLM (`openai`) e l'elaborazione dei report (`Markdown`, `iocextract`).
- **Dettagli implementativi del generatore:** Verrà analizzato il codice del generatore, mostrando attraverso snippet significativi come è stata realizzata l'orchestrazione dei vari elementi che lo compongono.
- **Implementazione della Pipeline di valutazione:** Si descriverà la logica dei due strumenti di valutazione sviluppati, focalizzandosi su come è stato strutturato il *matching* tra gli oggetti ed il confronto degli stessi.
- **Sfide tecniche e implementative:** Infine, si descriveranno le principali sfide incontrate durante lo sviluppo e le soluzioni adottate per mitigarle.

### 4.1 Stack tecnologico e ambiente di sviluppo

La realizzazione del sistema ha richiesto l'integrazione di tecnologie eterogenee spaziando dall'elaborazione del linguaggio naturale alla gestione rigorosa di strutture dati. Per supportare l'architettura descritta nel capitolo precedente, è stato selezionato uno stack tecnologico basato interamente sull'ecosistema **Python** (versione  $\geq 3.8$ ) [35].

La scelta di quest'ultimo come linguaggio di riferimento è motivata dalla sua posizione dominante sia nel campo dell'Intelligenza Artificiale Generativa, sia in quello della Cybersecu-

ality. Python offre, infatti, le uniche implementazioni ufficiali e mantenute della libreria OASIS per lo standard STIX, garantendo una conformità nativa alle specifiche.

L'ambiente di sviluppo primario è stato **Jupyter Notebook** [36]: questa scelta ha garantito un approccio allo sviluppo iterativo e sperimentale. La natura a celle dei notebook ha permesso di visualizzare immediatamente i grafi generati e di calibrare i parametri di inferenza in tempo reale, combinando codice eseguibile, documentazione Markdown e visualizzazioni grafiche in un unico documento interattivo.

Di seguito verranno analizzate le librerie core che compongono l'ossatura del sistema.

#### 4.1.1 Manipolazione dati e standard: La libreria `stix2`

Il cuore funzionale del progetto è la libreria `stix2` [32] che implementa l'intera logica di business dello standard STIX 2.1. Nel progetto il suo utilizzo è stato critico per tre scopi distinti:

1. **Generazione e validazione:** La libreria garantisce che ogni oggetto stanziato (es. `stix2.Malware`, `stix2.Indicator`) rispetti i vincoli obbligatori (proprietà *required*, formati di timestamp).
2. **Gestione del versionamento:** Gestisce automaticamente la generazione di **UUIDv4** casuali per le entità.
3. **Algoritmi di confronto (Similarity & Equivalence):** Per la fase di valutazione, sono state sfruttate le funzioni di comparazione semantica offerte dalla libreria:
  - **object\_equivalence:** Funzione booleana che determina se due oggetti rappresentano la stessa entità, ma ignora le differenze superficiali (come diversi ID o timestamp di creazione).
  - **object\_similarity:** Funzione che restituisce un punteggio (0 - 100) basato sulla sovrapposizione ponderata delle proprietà.
  - **Graph Similarity:** Estende il concetto di similarità all'intero grafo permettendo di confrontare il bundle generato con il *Ground Truth* valutando non solo i nodi, ma anche la topologia delle connessioni.

#### 4.1.2 Interazione con gli LLM: `openai`

L'interfaccia con i Large Language Model è gestita tramite la libreria ufficiale `openai` [37]. L'implementazione sfrutta le funzionalità più recenti dell'API *ChatCompletion*:

- **System & User Roles:** Serve ad implementare la strategia di *Role Prompting* definita nel prompt engineering.
- **JSON Mode:** L'utilizzo del parametro `response_format` è stato fondamentale per costringere modelli probabilistici a produrre output sintatticamente validi.
- **Error Handling:** La libreria gestisce i *retry* automatici in caso di saturazione del *Rate Limit* o errori temporanei del server.

### 4.1.3 Normalization and Ingestion: MarkItDown e iocextract

Per alimentare il generatore con dati puliti sono state impiegate due librerie specializzate:

- **MarkItDown**: Utilizzata nella fase di ingestion per convertire file eterogenei in formato **Markdown** [31]. La scelta del Markdown rispetto al testo semplice è strategica: preservando la struttura semantica del documento (titoli, elenchi puntati, tabelle), si fornisce all'LLM un contesto strutturale che migliora significativamente la capacità di estrarre oggetti validi.
- **iocextract**: Una libreria ottimizzata per l'estrazione di Indicatori di Compromissione tramite *Regular Expressions* avanzate. A differenza di regex "classiche", **iocextract** gestisce casi limite complessi (es. "defanged" IP come 1.2.3[.]4, URL offuscati).

## 4.2 Dettagli implementativi del generatore

Questo rappresenta il cuore del sistema. Di seguito vengono analizzati i blocchi di codice più significativi che relaizzano la pipeline del generatore.

### 4.2.1 Setup e configurazione

In questa fase iniziale si prepara l'ambiente di lavoro:

```

1 from openai import AzureOpenAI
2
3 Configuration: Set up the Azure OpenAI client
4 try:
5     client = AzureOpenAI(
6         azure_endpoint = userdata.get('AZURE_OPENAI_ENDPOINT'),
7         api_key=userdata.get('AZURE_OPENAI_KEY'),
8         api_version="2024-12-01-preview"
9     )
10    DEPLOYMENT_NAME = userdata.get('DEPLOYMENT_NAME')
11    print("Azure OpenAI client configured successfully.")
12
13    # --- GLOBAL CONFIGURATION OF MODELS ---
14    # The temperature can be a value between 0.1 and 1, lower temperature for
15    # more predictable, structured output.
16    # Reasoning can be set to: minimal, low, medium, or high values.
17
18    # IOC extraction
19    TEMP_IOC_EXTRACTION = 1
20    REASONING_IOC_EXTRACTION = "high"
21
22    # SDO extraction
23    TEMP_SDO_EXTRACTION = 1
24    REASONING_SDO_EXTRACTION = "high"
25
26    print("Global configuration variables loaded.")

```

```
26 # --- END OF CONFIGURATION ---
```

Listing 4.1: Setup e Configurazione

Vengono installate le librerie necessarie e configurato l'ambiente **Azure OpenAI**.

Di particolare importanza sono i parametri di configurazione del modello LLM:

- **TEMP\_IOC\_EXTRACTION** e **TEMP\_SDO\_EXTRACTION**: Impostano la temperature per le due chiamate all'LLM presenti nel generatore, una per l'estrazione e la convalida degli IoC e l'altra per l'estrazione degli SDO.
- **REASONING\_IOC\_EXTRACTION** e **REASONING\_SDO\_EXTRACTION**: Impostano il livello di ragionamento del modello (solo se supportato dal modello stesso).

### 4.2.2 Ingestion and Normalization Markdown

La prima sfida tecnica è la normalizzazione dell'input. Il codice utilizza la libreria **MarkItDown** per convertire qualsiasi formato di input in una stringa Markdown pulita.

```
1 def convert_pdf_to_markdown(filename, folder_path):
2     """
3     Converts a file to Markdown.
4     """
5     # Proceed with the conversion
6     print(f"File '{filename}' found. I'm starting the conversion to Markdown ...")
7     try:
8         # 1. Create an instance of the MarkItDown class
9         md_converter = MarkItDown(enable_plugins=False)
10
11        # 2. Call the .convert() method on the created object
12        result = md_converter.convert(full_pdf_path)
13
14        print("Conversion successfully completed!")
15
16        # 3. Return the textual content of the result
17        return result.text_content
```

Listing 4.2: Conversione in Markdown

Questo è un estratto della funzione `convert_pdf_to_markdown` che converte un PDF in Markdown. Essa ha come input: `filename`, il nome del file PDF da convertire, e `folder_path`, il percorso della cartella che lo contiene. Restituisce `result.tect_content`, il contenuto del PDF convertito in Markdown.

### 4.2.3 Estrazione degli IoC

In questa parte per estrarre gli IoC si usa un approccio ibrido a 3 fasi:

1. **Stage 1 (Regex)**: Viene usata la libreria `iocextract` per estrarre velocemente gli IoC.

2. **Stage 2 (LLM Analysis):** Viene chiesto all'LLM di estrarre solo gli indicatori riconducibili a delle minacce reali strutturandoli in un JSON preciso.
3. **Stage 3 (Consolidation):** Vengono uniti i risultati, e se un indicatore è stato trovato sia dalla Regex che dall'LLM, il suo punteggio di confidenza diventa "Alto".

Estratto del primo Stage che estrae gli IoC facendo uso della libreria `iocextract`.

```

1 def stage1_regex_triage(text: str) -> Dict[str, str]:
2     print("--- Stage 1: Starting regex triage ---")
3     candidate_dict: Dict[str, str] = {}
4     try:
5         # Extract IPs
6         for ip in iocextract.extract_ips(text, refang=True):
7             # Check to avoid common local IPs if not desired
8             if ip not in ['127.0.0.1']:
9                 candidate_dict[ip] = 'ipv4'
10
11        # Extract hashes
12        for h in iocextract.extract_hashes(text):
13            h_lower = h.lower()
14            if len(h_lower) == 32: candidate_dict[h_lower] = 'md5'
15            elif len(h_lower) == 40: candidate_dict[h_lower] = 'sha1'
16            elif len(h_lower) == 64: candidate_dict[h_lower] = 'sha256'
17
18        # Regex for URLs
19        for url in iocextract.extract_urls(text, refang=True):
20            candidate_dict[url] = 'url'
21
22        # Regular expression for specific directories and paths
23        path_re = re.compile(r'(/[\\w\\.\\/-\\|]+|[%[a-zA-Z]+[%[\\w\\.\\/-]+)')
24        for path in set(path_re.findall(text)):
25            if path not in candidate_dict:
26                if '.' in path.split('/')[1].split('\\')[1] and not path.
27                endswith(('/', '\\')):
28                    candidate_dict[path] = 'file-path'
29                else:
30                    candidate_dict[path] = 'directory'
31        return candidate_dict

```

Listing 4.3: Regex Triage

Estratto del secondo Stage che usa LLM per analizzare il contesto ed estrarre gli IoC che effettivamente rappresentano una minaccia.

```

1 def stage2_llm_analysis(text: str, openai_client: OpenAI, deployment_name:
2     str) -> List[Dict]:
3     print("\n--- Stage 2: Starting deep contextual analysis with LLM ---")
4     if not openai_client: return []
5
6     prompt = f"""As a senior CTI analyst, your task is to meticulously
7     analyze the following threat report. Your goal is to identify ALL

```

```

Indicators of Compromise (IOCs). Scrutinize the text to confirm that IOCs
are presented in a malicious context.
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
try:
    response = openai_client.chat.completions.create(
        model=deployment_name,
        messages=[{"role": "user", "content": prompt}],
        functions=[IOC_FUNCTION_SCHEMA],
        function_call={"name": "extract_and_validate_iocs"},
        temperature=TEMP_IOC_EXTRACTION,
        max_tokens=8192,
        reasoning_effort=REASONING_IOC_EXTRACTION
    )

    result = response.choices[0].message
    if result.function_call:
        function_args = json.loads(result.function_call.arguments)
        llm_iocs = function_args.get("iocs", [])
        print("Stage 2: LLM extracted and validated {len(llm_iocs)} IOCs.
    ")
    return llm_iocs
    else:
        print("Stage 2: LLM did not call the function. No IOCs extracted
        by LLM.")
        return []

```

Listing 4.4: LLM Analysis

Estratto del terzo Stage che consolida i risultati estratti tramite Regex e quelli estratti tramite LLM.

```

1 def stage3_consolidate_and_score(regex_iocs: Dict[str, str], llm_iocs: List[
2     Dict]) -> List[Dict]:
3
4     print("\n--- Stage 3: Consolidating and scoring IOCs ---")
5
6     final_iocs: Dict[str, Dict] = {}
7
8     # First process the IOCs from the LLM (most reliable by type and context)
9     for ioc in llm_iocs:
10         ioc_type = ioc.get("type")
11         value = ioc.get("value", "")
12         if ioc_type in ['md5', 'sha1', 'sha256']:
13             value = value.lower()
14
15         # Prepare the base object
16         final_ioc_data = {
17             "value": value,
18             "type": ioc_type,
19             "name": ioc.get('name', 'N/A'),
20             "description": ioc.get('description', ''),
21             "confidence": 'medium'

```

```

20     }
21     # Add specific fields for file paths if present and valid
22     if ioc_type == 'file-path':
23         if ioc.get("filename") and ioc.get("directory_path"):
24             final_ioc_data["filename"] = ioc.get("filename")
25             final_ioc_data["directory_path"] = ioc.get("directory_path")
26         else:
27             print("LLM extracted file-path without required filename/
directory_path: {value}")
28             continue
29
30     final_iocs[value] = final_ioc_data
31
32     # Compare with regex candidates to increase confidence
33     for regex_value, regex_type in regex_iocs.items():
34         if regex_type in ['md5', 'sha1', 'sha256']:
35             regex_value = regex_value.lower()
36         if regex_type == 'ipv4-addr': regex_type = 'ipv4'
37
38         if regex_value in final_iocs:
39             # If the LLM found the same value, it increases confidence.
40             final_iocs[regex_value]['confidence'] = 'high'
41
42     consolidated_list = list(final_iocs.values())
43     return consolidated_list

```

Listing 4.5: Consolidazione

#### 4.2.4 Generazione SCO & Indicator

In questa fase, in modo programmatico, gli IoC estratti vengono trasformati in SCO (l'oggetto reale) e Indicatori (la regola per rilevare quell'oggetto), inoltre vengono create le relazioni contestuali agli oggetti estratti.

La generazione segue vari step:

Questo è un estratto del primo step in cui vengono creati gli SCO e gli Indicatori per gli IoC che non rappresentano un has:

```

1 def generate_sco_and_indicators_with_relations(ioc_list: List[Dict]) -> Tuple
[List, List, List]:
2     # Step 1: Create SCOs and Indicators for non-hash types
3     for ioc in ioc_list:
4         ioc_type = ioc.get("type")
5         ioc_value = ioc.get("value", "")
6         ioc_name = ioc.get("name")
7         ioc_desc = ioc.get("description")
8
9         sco = None
10        indicator_pattern = None
11
12        try:
13            # Clean and validate IP

```

```

14         if ioc_type == "ipv4":
15             ip_cleaned = ioc_value.replace('[.]', '.').replace('[:]', ':')
16         )
17         ip_only = ip_cleaned.split(':')[0]
18         sco = IPv4Address(value=ip_only)
19         indicator_pattern = f"[ipv4-addr:value = '{sco.value}']"
20     [...]
21
22     # Indicator Creation
23     if indicator_pattern:
24         if sco:
25             stix_scos.append(sco)
26
27             indicator = Indicator(
28                 allow_custom=True,
29                 name=ioc_name,
30                 description=ioc_desc,
31                 pattern_type="stix",
32                 pattern=indicator_pattern,
33                 valid_from=dt.datetime.now(dt.timezone.utc)
34             )
35             stix_indicators.append(indicator)
36
37     elif ioc_type not in ["md5", "sha1", "sha256"]:
38         print("Internal Logic Error: Pattern not generated for IOC:
        {ioc}")

```

Listing 4.6: Generazione SCO e Indicator: Step 1

Questo è un estratto del secondo step in cui vengono creati gli SCO e gli Indicator per i file hash:

```

1 def generate_sco_and_indicators_with_relations(ioc_list: List[Dict]) -> Tuple
  [List, List, List]:
2     # Step 2: Create an Indicator for EACH hash file
3     for ioc in ioc_list:
4         ioc_type = ioc.get("type")
5         if ioc_type in ["md5", "sha1", "sha256"]:
6             ioc_value = ioc.get("value", "").lower()
7             file_name_desc = ioc.get("name")
8
9             if not ioc_value or not file_name_desc:
10                 print("Skipping HASH IOC due to missing value/name: {ioc}")
11                 continue
12
13             try:
14                 indicator = Indicator(
15                     allow_custom=True,
16                     name=f"Indicator for {file_name_desc} ({ioc_type.upper()}
17                 ),
18                 description=ioc.get("description"),
19                 pattern_type="stix",

```

```

19         pattern=generate_stix_pattern_single_hash(ioc_type,
ioc_value),
20         valid_from=dt.datetime.now(dt.timezone.utc)
21     )
22     stix_indicators.append(indicator)

```

Listing 4.7: Generazione SCO e Indicator: Step 2

Questo è un estratto dell'ultimo step in cui vengono create le relazioni **derived-from** tra SCO e Indicator:

```

1 def generate_sco_and_indicators_with_relations(ioc_list: List[Dict]) -> Tuple
[ List, List, List]:
2     # Step 3: Create 'derived-from' relationships
3     for file_name, indicators in file_indicator_map.items():
4         if len(indicators) > 1:
5             for ind1, ind2 in combinations(indicators, 2):
6                 try:
7                     stix_relationships.append(Relationship(ind1.id, 'derived
-from', ind2.id))
8
9     return stix_scos, stix_indicators, stix_relationships

```

Listing 4.8: Generazione SCO e Indicator: Step 3

#### 4.2.5 Estrazione SDO

Mentre la parte precedente gestiva i dati tecnici (IP, Hash), questa so occuperà di narrativa e contesto. Si farà uso di un prompt molto dettagliato (che abbiamo discusso nei capitoli precedenti) per chiedere all'LLM di identificare le entità di alto livello, come:

- **Malware:** Nome, tipo, descrizione.
- **Threat Actor:** Chi è l'attaccante.
- **Attack Pattern:** Le tecniche MITRE ATT&CK usate.
- **Vulnerability:** Le CVE sfruttate.

Il risultato è un grande oggetto JSON contenente tutti gli SDO estratti.

Questo è un estratto del codice per la generazione degli SDO:

```

1 def extract_all_entities_revised(report_text: str):
2     try:
3         response = client.chat.completions.create(
4             model=DEPLOYMENT_NAME,
5             messages=[{"role": "user", "content": prompt}],
6             temperature=TEMP_SDO_EXTRACTION,
7             response_format={"type": "json_object"},
8             reasoning_effort=REASONING_SDO_EXTRACTION
9         )
10
11     entity_list = result_json.get("entities", [])

```

```

12         return result_json
13
14 # --- Extraction execution ---
15 sdo_entities = []
16 main_malware_name = None
17 if text and client:
18     extraction_result = extract_all_entities_revised(text)
19     if extraction_result:
20         sdo_entities = extraction_result.get("entities", [])
21
22     print(json.dumps(sdo_entities, indent=2))

```

Listing 4.9: Generazione SDO

#### 4.2.6 Assemblaggio finale

Questa è l'ultima fase, in cui viene unito il tutto creando il grafo STIX finale. In particolare, vengono creati gli SDO (sulla base di quelli estratti in precedenza), le relazioni contestuali a questi, si impacchetta il tutto in un oggetto `Bundle` come file JSON.

Questo è un estratto del codice per la creazione degli SDO:

```

1     created_sdos = {}
2     malware_main_obj = None
3
4     for entity in sdo_entities:
5         entity_type = entity.get("type")
6         entity_name = entity.get("name")
7         sdo = None
8
9         if entity_type == "malware":
10             sdo = Malware(
11                 name=entity_name.lower(),
12                 is_family=True,
13                 description=entity.get("description"),
14                 malware_types=entity.get("malware_types", ["remote-access-
trojan"]),
15                 created_by_ref=identity_author.id
16             )
17             created_sdos[entity_name] = sdo
18
19         elif entity_type == "attack-pattern":
20             kill_chain_phases = entity.get("kill_chain_phases", [])
21             for phase in kill_chain_phases:
22                 if 'phase_name' in phase:
23                     phase['phase_name'] = phase['phase_name'].lower().replace
(' ', '-')
24
25             sdo = AttackPattern(
26                 name=entity_name,
27                 description=entity.get("description"),
28                 created_by_ref=identity_author.id,

```

```

29         kill_chain_phases=kill_chain_phases,
30         external_references=[{
31             "source_name": "mitre-attack",
32             "external_id": entity.get("external_id"),
33             "url": f"https://attack.mitre.org/techniques/{entity.get(
('external_id').replace('.', '/'))}"
34         }]
35     )
36     created_sdos[entity.get("external_id")] = sdo
37
38 [...]
```

```

39
40 # --- Create the final bundle ---
41 final_bundle = Bundle(*all_stix_objects)
```

Listing 4.10: Creazione SDO

### 4.3 Implementazione delle Pipeline di valutazione

La validazione scientifica del sistema ha richiesto lo sviluppo di due strumenti di analisi dedicati. La sfida implementativa principale non risiede nel calcolo delle formule statistiche, ma nella fase preliminare di **allineamento degli oggetti**, in particolare nel modo in cui decidere se due oggetti "raccontano" la stessa minaccia.

#### 4.3.1 Pipeline 1: Semantic Evaluation

Come discusso nei capitoli precedenti, questa pipeline di valutazione si occupa di valutare i singoli oggetti in maniera semantica considerando se i due oggetti rappresentano la stessa minaccia sulla base delle proprietà significative di questi ultimi e non su timestamp o ID.

Vediamo degli snippet di codice significativi:

##### Similarity Threshold

Questa prima parte, oltre ad importare le librerie necessarie, si occupa di definire le soglie di similarità per la valutazione, una per gli SDO e SCO e uno per le SRO:

```

1 # Set the similarity threshold for considering two objects equivalent (from 0
  to 100)
2 SIMILARITY_THRESHOLD = 80
3
4 # Set the similarity threshold for relationships (da 0 a 100)
5 RELATIONSHIP_SIMILARITY_THRESHOLD = 80
```

Listing 4.11: Soglie di similarità

Una soglia bassa rende il confronto meno stringente.

##### Funzioni di supporto

Qui abbiamo due funzioni di supporto al tool di valutazione:

`validate_bundle` controlla se il bundle JSON rispetta lo standard STIX 2.1:

```

1 def validate_bundle(file_path):
2     results = validate_file(file_path)
3
4     if results.is_valid:
5         print("Validation of {os.path.basename(file_path)} success.")
6         return True
7     else:
8         print("Validation ERROR for {os.path.basename(file_path)}:")
9         print_results(results)
10        return False

```

Listing 4.12: Validatore bundle

`extract_and_categorize_objects` separa SDO/SCO dagli SRO, questo perché successivamente saranno confrontati in due modi diversi:

```

1 def extract_and_categorize_objects(bundle):
2     sdo_sco_list = []
3     sro_list = []
4     objects = bundle.get("objects", [])
5
6     for obj in objects:
7         obj_type = obj.get("type", "")
8         if obj_type in ["relationship", "sighting", "sighting-of"]:
9             sro_list.append(obj)
10        else:
11            sdo_sco_list.append(obj)
12
13    return sdo_sco_list, sro_list

```

Listing 4.13: Separa SDO/SCO e SRO

### Logica di confronto principale

Qui, dopo una prima fase di preparazione avviene il confronto vero e proprio tra gli SDO/SCO:

Come prima cosa una funzione normalizza il contenuto delle regole YARA (come già discusso nei capitoli precedenti):

```

1 def normalize_braces_content(text):
2     def replacer(match):
3         content = match.group(1)
4         cleaned_content = re.sub(r'[^a-z0-9\s]', '', content.lower())
5         normalized_content = ' '.join(cleaned_content.split())
6         return '{' + normalized_content + '}'
7     return re.sub(r'\{(.*)\}', replacer, text, flags=re.DOTALL)

```

Listing 4.14: Normalizzazione regole YARA

In seguito gli indicatori con le regole YARA normalizzate vengono confrontati comparando la stringa della regola generata e quella del Ground Truth:

```

1 elif pattern_type_gen == 'yara':
2     norm_gen = pattern_gen.lower()
3     norm_exp = pattern_exp.lower()
4     norm_gen = re.sub(r'//.*', '', norm_gen)

```

```

5     norm_exp = re.sub(r'//.*', '', norm_exp)
6     norm_gen = ' '.join(norm_gen.split())
7     norm_exp = ' '.join(norm_exp.split())
8
9     norm_gen_braces = normalize_braces_content(norm_gen)
10    norm_exp_braces = normalize_braces_content(norm_exp)
11    if norm_gen_braces == norm_exp_braces:
12        is_equivalent = True

```

Listing 4.15: Confronto Indicatori YARA

I restanti oggetti vengono semplicemente confrontati con la funzione `object_equivalence` di `stix2`:

```

1 if object_equivalence(gen_obj, exp_obj, threshold=SIMILARITY_THRESHOLD):
2     is_equivalent = True

```

Listing 4.16: Confronto semantico

Per quanto riguarda le relazioni, la libreria `stix2` non supporta il confronto semantico di queste ultime (come discusso nei capitoli precedenti), perciò in questo caso, si userà una logica a pesi ponderati quindi una relazione sarà corretta se collega gli stessi concetti:

```

1 weights = {'source_ref': 40, 'target_ref': 40, 'relationship_type': 20}
2
3 for exp_rel in search_pool_expert_sro:
4     current_score = 0
5     if id_map.get(gen_rel.get('source_ref')) == exp_rel.get('source_ref'):
6         :
7         current_score += weights['source_ref']
8     if id_map.get(gen_rel.get('target_ref')) == exp_rel.get('target_ref'):
9         :
10        current_score += weights['target_ref']
11    if gen_rel.get('relationship_type') == exp_rel.get('relationship_type'):
12        :
13        current_score += weights['relationship_type']
14
15    if current_score > highest_score:
16        highest_score = current_score
17        best_match_expert_rel = exp_rel
18
19 if highest_score >= RELATIONSHIP_SIMILARITY_THRESHOLD:
20     results['relationship']['TP'] += 1

```

Listing 4.17: Confronto SRO

### 4.3.2 Pipeline 2: Graph Evaluation

Qui il confronto non avviene sui singoli oggetti, ma sull'interezza del grafo usando la funzione `graph_similarity` della libreria `stix2` (di cui abbiamo discusso in precedenza).

Vediamo un estratto delle parti più importanti:

Per confrontare due grafi, la libreria `stix2` non accetta liste semplici, ma ha bisogno di caricare gli oggetti in un database in memoria chiamato `MemoryStore`.

```
1 ds_generated = MemoryStore(stix_data=gen_objects_to_compare)
2 ds_ground_truth = MemoryStore(stix_data=exp_objects_to_compare)
```

Listing 4.18: Caricamento in `MemoryStore`

A questo punto avviene il confronto con la funzione `graph_similarity`:

```
1 score = graph_similarity(ds_generated, ds_ground_truth, prop_scores=
    prop_scores)
```

Listing 4.19: Confronto dei Grafi

Alla fine del ciclo su tutti i bundle, lo script prende tutti i `DataFrame` dei singoli report e li fonde assieme. Questo permette di calcolare le statistiche globali su tutto il dataset di test:

```
1 master_df = pd.concat(all_dataframes, ignore_index=True)
2
3 final_avg_by_type = master_df.groupby('Type Object')['Match Score (%)'].mean
    ().sort_values(ascending=False)
4
5 final_df_avg_type = final_avg_by_type.reset_index()
6 final_df_avg_type.columns = ['Type Object', 'Average Score (%)']
```

Listing 4.20: Aggregazione Finale

## 4.4 Sfide tecniche e soluzioni adottate

Il passaggio dalla progettazione teorica all'implementazione pratica ha fatto emergere diverse criticità, in particolare durante la fase di valutazione dove il confronto automatico tra il Ground Truth e il Bundle generato ha richiesto lo sviluppo di logiche custom per superare i limiti della libreria standard.

### 4.4.1 Interoperabilità e confronto dei Pattern YARA

Una sfida significativa ha riguardato la gestione degli Indicatori basati su regole **YARA**, il confronto semantico di questi oggetti, infatti, si è rivelato problematico.

La funzione `object_similarity` è ottimizzata per il linguaggio *STIX Patterning*. Quando applicata a due oggetti indicator contenenti regole YARA, la funzione tende a restituire punteggi di similarità bassi o nulli se le stringhe differiscono anche solo per spaziature, indentazione o commenti, pur essendo logicamente identiche. Inoltre, in alcuni casi, il parser della libreria sollevava eccezioni tentando di interpretare la sintassi YARA come se fosse STIX Patterning.

Per risolvere il problema è stata implementata una logica di **Confronto Ad-Hoc per YARA** all'interno della pipeline di valutazione (di cui abbiamo discusso prima). Prima del confronto il codice intercetta gli indicatori di tipo YARA:

1. Estrae la stringa grezza dal pattern.

2. Applica una normalizzazione aggressiva.
3. Esegue un confronto diretto sulle stringhe normalizzate.

Questo approccio ha permesso di riconoscere correttamente come "*True Positive*" regole YARA generate dall'LLM che differivano dal Ground Truth solo per la formattazione estetica.

#### 4.4.2 La sfida "topologica" delle relazioni

Il confronto degli oggetti `Relationship` rappresenta la sfida più complessa del sistema di valutazione.

Un oggetto relazione in STIX è definito da tre proprietà: `source_ref`, `relationship_type`, `target_ref`. Poiché la libreria `stix2` non supporta il confronto di questi oggetti, è stato necessario realizzare una logica personalizzata per le relazioni. Per farlo si è verificato che il `source_ref` dell'oggetto generato corrispondesse al `source_ref` dell'oggetto nel ground truth. La stessa cosa viene fatta con il `target_ref` e se entrambi corrispondono si controlla che anche il `relationship_type` sia lo stesso per una corrispondenza completa.

Questo algoritmo ha permesso di valutare la correttezza della struttura del grafo indipendentemente dalla libreria `stix2`.

## Capitolo 5

# Analisi e discussione dei risultati

Dopo aver analizzato l'architettura del sistema e l'implementazione delle pipeline di valutazione, questo capitolo presenta i risultati sperimentali ottenuti. Costituisce il nucleo empirico della tesi in cui l'efficacia del sistema proposto viene misurata quantitativamente rispetto al *Ground Truth* validato, attraverso un protocollo sperimentale che ha coinvolto il test comparativo di diverse famiglie di Large Language Models.

La domanda di ricerca fondamentale che guida questo capitolo si sposta sulla **sostenibilità operativa**: *è effettivamente adottabile un sistema basato su LLM in un contesto critico e a bassa tolleranza d'errore come la Cyber Threat Intelligence? E quali vantaggi tangibili, in termini di efficienza e profondità di analisi, può apportare concretamente al lavoro quotidiano degli analisti?*

Nel dettaglio, l'esposizione dei risultati è strutturata nei seguenti punti chiave:

- **Analisi granulare degli oggetti:** Verranno discussi i risultati della prima pipeline di valutazione, focalizzata su un confronto semantico puntuale dei singoli oggetti. Si misurerà la capacità del sistema di rilevare correttamente le entità e verranno analizzate le performance comparative dei diversi modelli LLM utilizzati, evidenziando come ciascuna architettura gestisca la generazione dei vari oggetti.
- **Valutazione olistica del grafo:** Si esamineranno i dati della seconda pipeline, dedicata alla validazione della struttura complessiva del grafo. Si verificherà se il sistema è in grado di ricostruire correttamente la topologia delle relazioni nel suo insieme, preservando il contesto narrativo del report originale, e si confronterà la capacità dei vari modelli di mantenere la coerenza logica su grafi complessi.
- **Discussione critica e implicazioni:** Infine, i risultati numerici verranno interpretati per delineare i confini operativi della tecnologia, discutendo i rischi residui e il ruolo imprescindibile della supervisione umana nell'adozione di questi strumenti.

### 5.1 Analisi delle performance di rilevamento

La prima fase dell'analisi sperimentale si concentra sulla capacità del sistema di identificare ed estrarre le corrette entità di minaccia dai report non strutturati. Utilizzando la metrica

di *Object Equivalence* implementata nella Pipeline, sono state confrontate le performance di cinque diverse architetture LLM testando per ciascuna due configurazioni di temperatura ( $T = 0.1$  vs  $T = 1.0$ ) o di *reasoning effort* (per i modelli più recenti).

I dati raccolti dipingono un quadro estremamente positivo per l'applicabilità dell'IA generativa alla CTI: i modelli più recenti hanno superato la barriera critica della comprensione semantica dimostrando una precisione quasi umana nell'estrazione dei comportamenti (TTP).

### 5.1.1 I risultati nell'estrazione dei TTP

Il risultato più rilevante e promettente dell'intera ricerca riguarda la classe **attack-pattern**, infatti uno dei compiti più impegnativi è proprio quello di mappare descrizioni testuali complesse su tecniche MITRE ATT&CK precise.

I risultati ottenuti mostrano che il sistema proposto, guidato da prompt ben strutturati e dai modelli LLM più recenti, riesce egregiamente in questo compito:

- **Eccellenza di GPT-5 e Gemini 2.5:** I modelli hanno raggiunto livelli di accuratezza incoraggianti, come si può vedere dai grafici delle figure 5.1 e 5.2. In particolare **GPT-5 (High Reasoning)** registra una **Precision del 93.8%** e un **F1-Score del 62.4%**. **Gemini 2.5 pro**, allo stesso modo, ottiene dei risultati speculari con una **Precision del 94.2%** e un **F1-Score del 62.3%**.

Questo dimostra che, quando il report descrive una tecnica di attacco, nel **94%** dei casi viene correttamente individuata anche dal sistema perciò il rischio di falsi positivi (allucinazioni di attacchi non avvenuti) è praticamente azzerato.

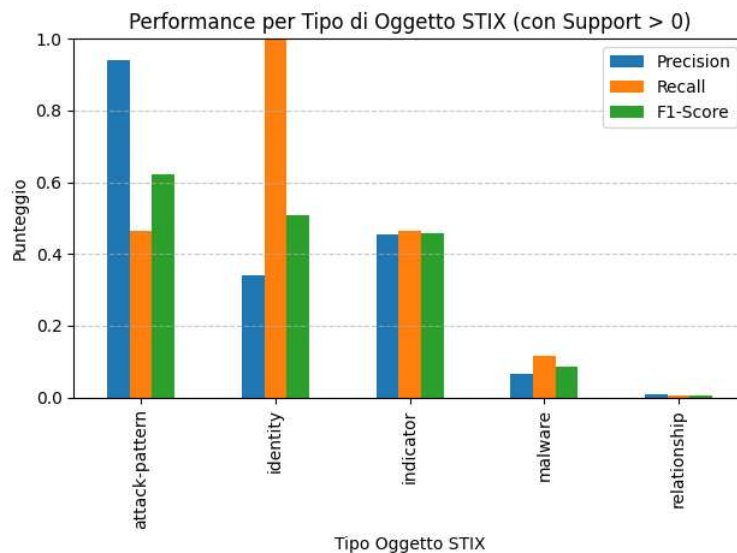


Figura 5.1: Performance per classe: Gemini 2.5 Pro high reasoning

- **I Modelli Open (Llama 4):** Il confronto tra i due modelli di Llama evidenzia un'evoluzione radicale: mentre **Llama 3.3** (figura 5.3) faticava a comprendere il concetto di TTP (F1-Score del 4.4%), il modello **Llama 4 Maverick** (figura 5.4) raggiunge un **F1-Score del 36.6%** e una **Precision del 55.8%**. Sebbene non ancora ai livelli di GPT

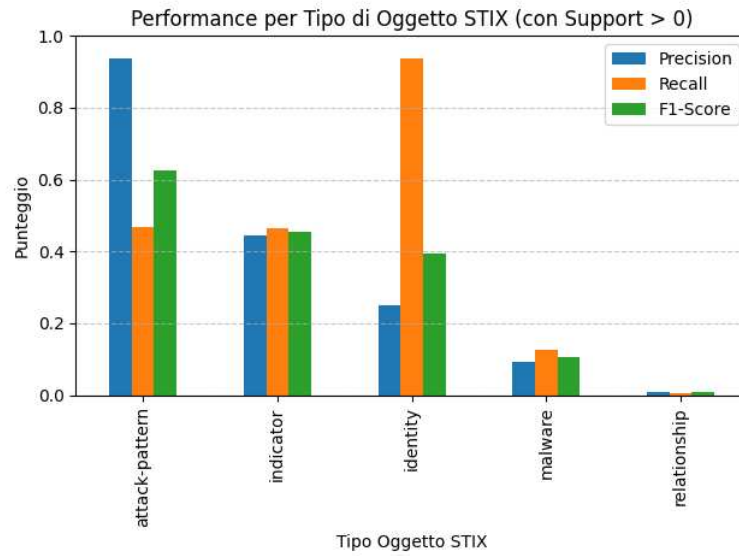
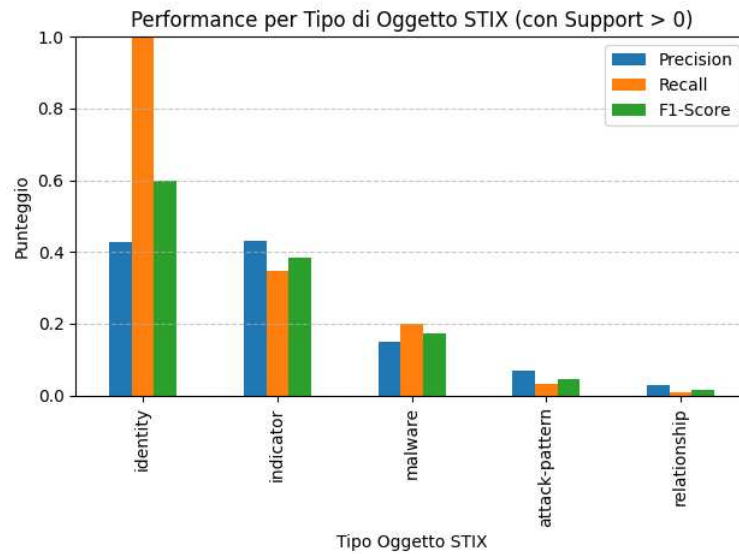


Figura 5.2: Performance per classe: GPT-5 high reasoning

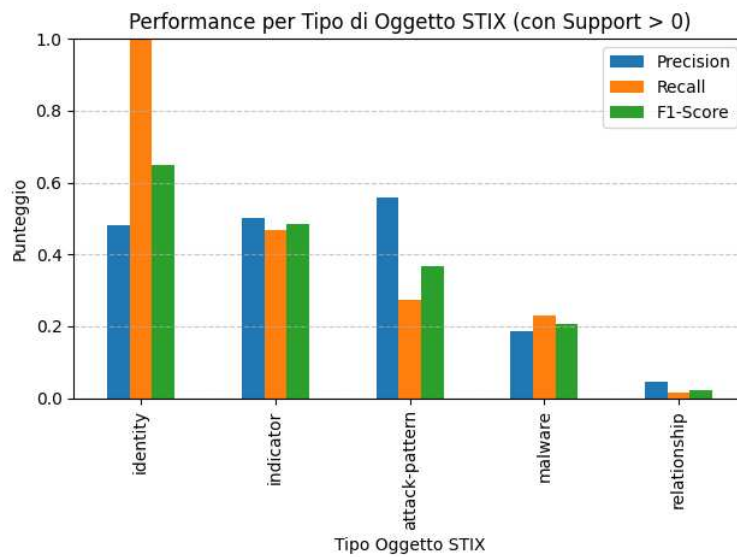
e Gemini, Llama 4 dimostra che è possibile eseguire analisi comportamentali complesse anche con modelli aperti.

Figura 5.3: Performance per classe: Llama 3.3  $T = 0.1$ 

### 5.1.2 Affidabilità nel rilevamento delle vittime (Identity)

Analizzando la classe `identity` (che identifica le vittime o i settori bersaglio), il sistema dimostra una capacità di **Recall** eccezionale.

Tutti i modelli testati hanno raggiunto una **Recall tra l'88% e il 100%**, questo significa che il sistema molto raramente perde qualche informazione. La Precision più bassa ( $\sim 30 - 40\%$ ) è attribuibile a una tendenza "molto cauta" del modello, che preferisce estrarre anche entità marginali (es. aziende citate come contesto) piuttosto che rischiare di ignorare

Figura 5.4: Performance per classe: Llama 4 Maverick  $T = 0.1$ 

Modello	Setup	TTP Precision	TTP F1-Score	Identity Recall	Indicator F1-Score
<b>GPT-5</b>	High Reasoning	<b>93.8%</b>	<b>62.4%</b>	93.8%	45.4%
<b>Gemini 2.5 Pro</b>	High Reasoning / Temp 0.1	<b>94.2%</b>	<b>62.3%</b>	<b>100.0%</b>	46.0%
<b>GPT-4.1</b>	Temp 0.1	68.3%	45.2%	<b>100.0%</b>	44.0%
<b>Llama 4 Maverick</b>	Temp 0.1	55.8%	36.6%	<b>100.0%</b>	<b>48.3%</b>
<b>Llama 3.3</b>	Temp 0.1	7.0%	4.4%	<b>100.0%</b>	38.5%

Tabella 5.1: Confronto delle Performance per Classe (Best Setup per Modello)

una potenziale vittima. Questo comportamento può portare alla creazione di molti falsi positivi che vanno ad abbassare il punteggio sulla Precision, perciò, sebbene il punteggio possa sembrare basso, va contestualizzato nell'ambito del dataset selezionato, infatti, le scelte fatte dagli autori del Ground Truth potrebbero non corrispondere con quelle del modello, ma ciò non denota a priori un errore da parte di quest'ultimo.

### 5.1.3 La gestione degli indicatori tecnici (Indicator)

Per la classe **Indicator** (IoC come Hash e IP), le performance si dimostrano estremamente stabili attraverso tutti i modelli con un **F1-Score medio costante attorno al 45-48%**.

In particolare **Llama 4 Maverick** ( $T = 0.1$ ) ottiene sorprendentemente il punteggio migliore in questa categoria, con un **F1-Score di 48.3%**, superando leggermente anche GPT-5 e Gemini. Questo risultato convalida la scelta dell'architettura adottata, poiché l'estrazione degli IoC non è gestita solo dall'LLM. La qualità del risultato non dipende esclusivamente dall'"intelligenza" del modello, rendendo il tool robusto anche con modelli meno performanti.

### 5.1.4 Benchmark sintetico dei modelli

La tabella 5.1 riassume le metriche chiave per le configurazioni migliori di ciascun modello (a bassa temperatura / high reasoning) evidenziando i punti di forza specifici.

### 5.1.5 Analisi dell’impatto della temperatura

L’esperimento ha anche confermato l’importanza dei parametri di inferenza: confrontando i dati a  $T = 0.1$  vs  $T = 1.0$  i modelli a **bassa temperatura** ( $T = 0.1$ ) mostrano un F1-Score superiore. L’aumento della temperatura, pur aumentando la varietà, introduce rumore che degrada la precisione strutturale, confermando che, per compiti tecnici come l’estrazione di entità STIX, il determinismo debba essere la priorità.

In conclusione, la pipeline certifica che **GPT-5** e **Gemini 2.5** rappresentano lo stato dell’arte per l’analisi, ma **Llama 4** emerge come un valido concorrente offrendo performance solide sugli indicatori tecnici (IoC) e accettabili sui comportamenti (TTP).

## 5.2 Valutazione olistica del grafo

Mentre la pipeline precedente si focalizzava sul rilevamento discreto delle entità, questa sposta l’analisi sul piano della **qualità complessiva** dell’intelligence prodotta. Per la valutazione è stata usata la funzione **graph-similarity**, che, a differenza del confronto nodo per nodo, prende in input due interi grafi e calcola un punteggio di similarità globale ponderato. L’algoritmo non si limita a confrontare le proprietà isolate, ma valuta quanto il grafo generato si sovrapponga semanticamente a quello di riferimento, penalizzando le discrepanze nei dettagli descrittivi e nelle connessioni.

I risultati mostrano che il sistema proposto è in grado di generare grafi con una fedeltà semantica elevata, specialmente nelle componenti più complesse come i TTP. Come si può osservare (figure 5.5 e 5.6), i modelli dotati di capacità di ragionamento avanzato (Gemini 2.5 Pro e GPT-5) mostrano punteggi complessivi superiori, specialmente nell’estrazione di entità complesse.

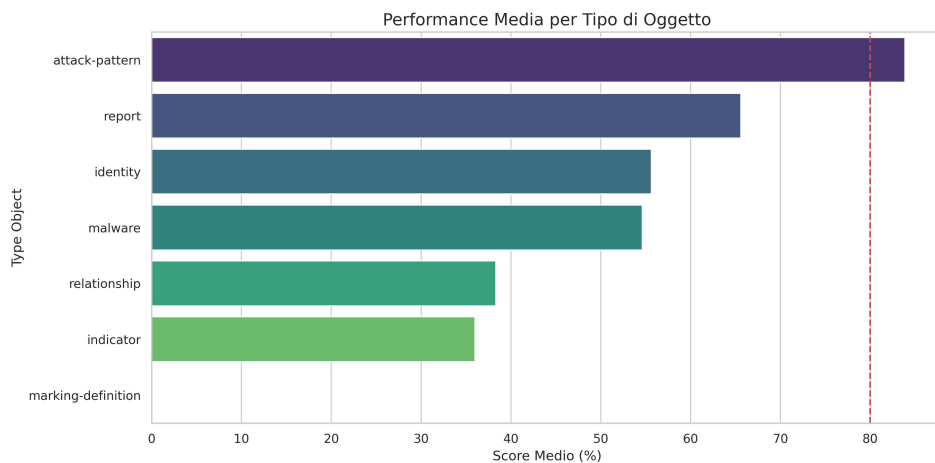


Figura 5.5: Performance media per tipo di oggetto Gemini 2.5 Pro

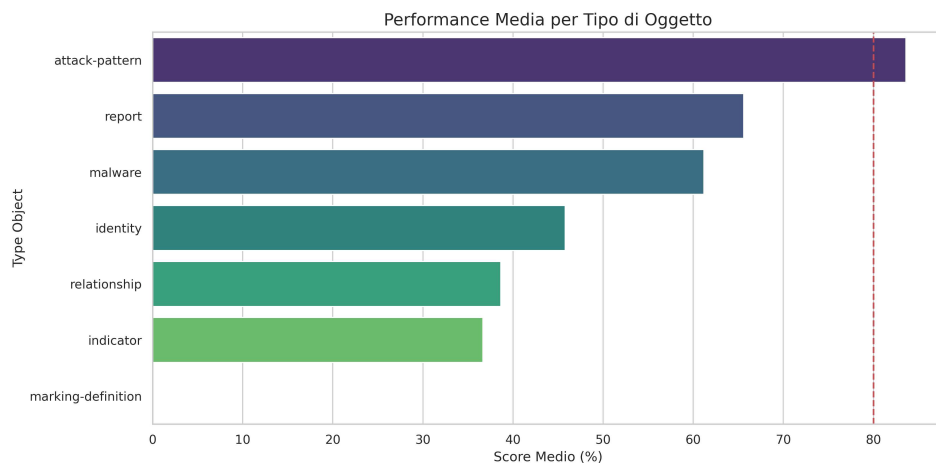


Figura 5.6: Performance media per tipo di oggetto GPT-5

### 5.2.1 Profondità descrittiva nei comportamenti (Attack Pattern)

La classe **attack-pattern** rappresenta il banco di prova per la capacità di "ragionamento" del modello. Qui lo score di similarità indica se il modello ha estratto correttamente i campi contestuali:

- **GPT-5 e Gemini 2.5 pro:** Anche in questo caso GPT-5 e Gemini 2.5 Pro rappresentano i modelli migliori (figure 5.5 e 5.6). In particolare, **Gemini 2.5 Pro** domina questa metrica con uno score medio di similarità del **83.9%**, e **GPT-5** segue a brevissima distanza con **83.6%**.

Uno score superiore all'80% indica che l'oggetto generato è semanticamente quasi identico a quello di riferimento, inoltre il gap con **GPT-4.1** (73.6%) dimostra che i modelli più recenti "comprendono" meglio lo standard MITRE.

- **Llama 4:** Anche **Llama 4 Maverick** (figura 5.7), in termini di qualità descrittiva (66.4%), stacca nettamente **Llama 3.3** (46.4%) confermandosi un'opzione valida per descrivere le minacce, anche se meno preciso di GPT-5 e Gemini 2.5.

### 5.2.2 L'eccellenza di Llama 4 su Malware e Identity

Un dato sorprendente emerge dall'analisi degli oggetti **Malware** e **Identity**.

Per quanto riguarda l'oggetto **Malware**, **Llama 4 Maverick** (figura 5.7) ottiene il punteggio di similarità più alto (**63.6%**) superando **GPT-5** (61.2%) e **Gemini** (54.6%).

Per l'oggetto **Identity**, anche qui, **Llama 4** eccelle con il **69.9%** distaccando nettamente GPT-5 (45.8%).

Questo suggerisce che Llama è più adatto nell'estrarre i metadati standard (es. alias, settori industriali) senza introdurre variazioni creative che il confronto penalizzerebbe, mentre i modelli Gemini e GPT-5 lo sono nel ragionamento astratto (TTP).

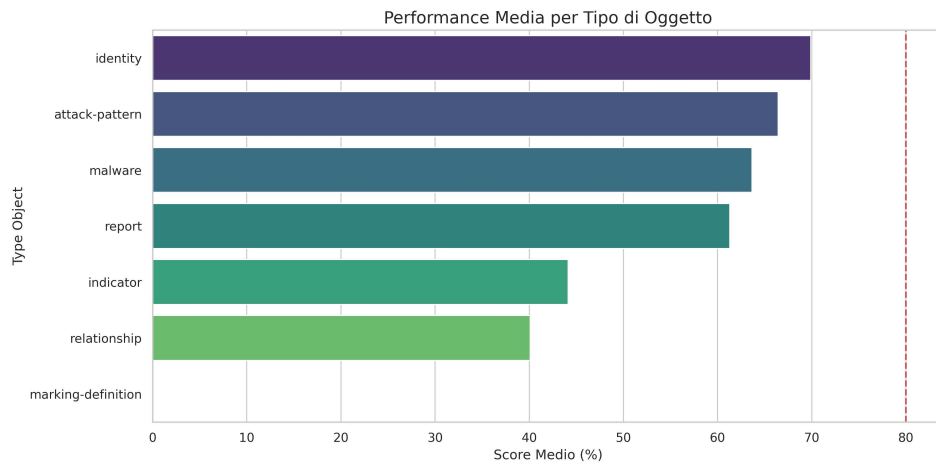


Figura 5.7: Performance media per tipo di oggetto Llama 4 Maverick

### 5.2.3 Coerenza topologica delle relazioni

La metrica per **relationship** misura la capacità di replicare la struttura del grafo. Poiché **graph-similarity** propaga la similarità attraverso i nodi connessi, questo punteggio riflette la correttezza della tripla *Sorgente, Tipo, Destinazione*. I punteggi si attestano per tutti i modelli tra il **38%** e il **40%**, con **Llama 4** leggermente in testa (40.1%). Il valore assoluto basso è fisiologico per questa metrica su grafi sparsi, ma la stabilità indica che l'architettura sviluppata garantisce una struttura coerente indipendentemente dal modello usato.

### 5.2.4 Impatto del Reasoning Effort (GPT-5 Low vs High)

Confrontando le due modalità di GPT-5: la similarità per **Attack Pattern** sale significativamente con *High Reasoning* (da 77.3% a 83.6%), per oggetti come **Malware**, invece, il guadagno è marginale (da 59.4% a 61.2%). Questo comunque giustifica il costo computazionale per un *Reasoning* maggiore per l'analisi comportamentale complessa, mettendo ancora una volta in luce l'efficacia dei modelli più recenti.

### 5.2.5 Analisi della stabilità per Bundle

Analizzando i risultati disaggregati per singolo report (figure 5.8, 5.9 e 5.10) si nota una certa variabilità dipendente dalla complessità del report di input.

Alcuni bundle (es. **bundle\_006**) ottengono costantemente punteggi elevati indicando che report ben strutturati facilitano l'estrazione, altri (es. **bundle\_015**) mostrano un crollo generalizzato suggerendo ambiguità nel testo di partenza che mettono in crisi anche i modelli più avanzati.

### 5.2.6 Analisi complessiva dei risultati

Approfondendo l'analisi tramite i dati raccolti è possibile isolare le ragioni delle variazioni di punteggio (tabella 5.2):

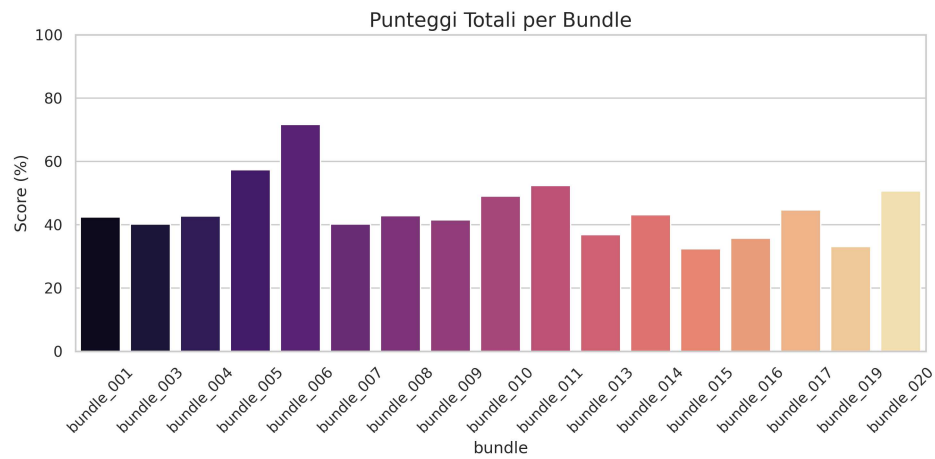


Figura 5.8: Performance per bundle Gemini 2.5 Pro

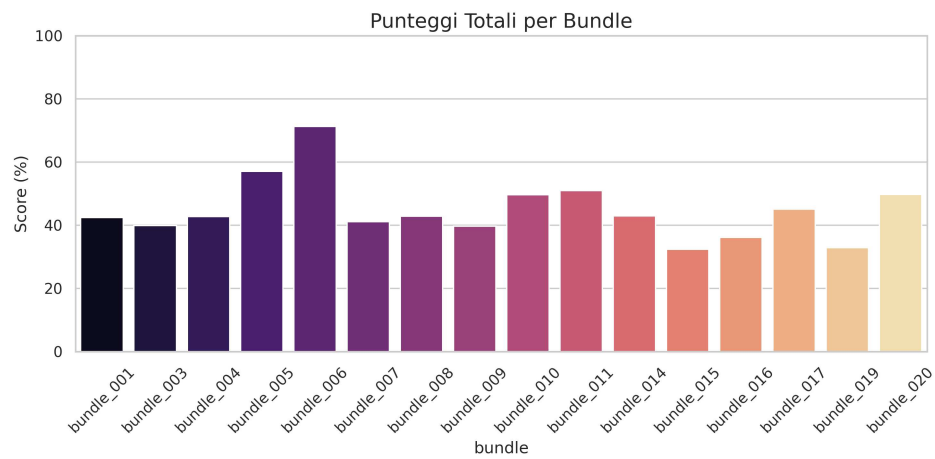


Figura 5.9: Performance per bundle GPT-5

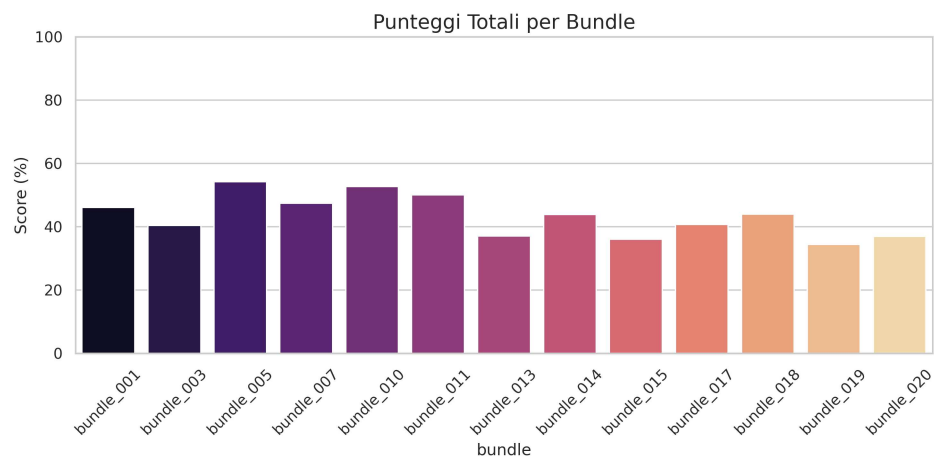


Figura 5.10: Performance per bundle Llama 4 Maverick

- **Match perfetti:** Si riscontrano frequentemente per *identity* e *attack-pattern* ge-

Modello	Attack Pattern	Malware	Identity	Relationship
<b>Gemini 2.5 Pro</b>	<b>83.9%</b>	54.6%	55.6%	38.3%
<b>GPT-5</b>	83.6%	61.2%	45.8%	38.6%
<b>GPT-4.1</b>	73.6%	57.2%	61.8%	38.3%
<b>Llama 4 Maverick</b>	66.4%	<b>63.6%</b>	<b>69.9%</b>	<b>40.01%</b>
<b>Llama 3.3</b>	46.4%	59.2%	64.8%	35.5%

Tabella 5.2: Sintesi dei Risultati

nerati da Gemini e GPT-5. In questi casi, l'LLM ha riportato esattamente le proprietà chiave massimizzando lo score di `object_similarity`.

- **Penalizzazioni lessicali:** Per i `malware` i punteggi medi si assestano raramente sopra il 65% e poichè il confronto per questa classe si basa strettamente sulle proprietà `name` e `malware_types`, la penalizzazione deriva quasi esclusivamente dalla **classificazione tassonomica**. Probabilmente l'LLM spesso fallisce nel replicare i `malware_types`. Il fatto che **Llama 4** qui ottenga punteggi migliori suggerisce che sia più "conservativo" o preciso nell'assegnare queste etichette rispetto agli altri modelli.
- **Match mancati:** I punteggi con 0% indicano casi di allineamento fallace dove l'algoritmo ha accoppiato oggetti che non condividevano alcuna proprietà semantica rilevante.

### 5.3 Discussione critica e implicazioni operative

L'analisi incrociata dei risultati ottenuti permette di tracciare un bilancio definitivo sulla maturità della tecnologia per l'automazione della CTI. Non da meno questo richiede anche una riflessione critica sulla natura stessa dello standard STIX 2.1 e sulle sfide intrinseche della sua valutazione automatica. Sebbene le metriche di similarità forniscano un indicatore oggettivo, esse si scontrano con la realtà operativa della CTI: **l'assenza di un'unica topologia corretta**.

#### 5.3.1 Il superamento dei test

Il dato più impattante della ricerca è la performance dei modelli *High Reasoning* (GPT-5, Gemini 2.5) sull'estrazione degli *Attack Pattern*. Con una precisione >94% e una fedeltà semantica >83%, questi strumenti dimostrano di aver superato la barriera della comprensione del testo tecnico. Per un analista CTI questo significa che l'automazione della mappatura MITRE ATT&CK, tradizionalmente una delle attività più onerose in termini di tempo ed energie, è oggi possibile. L'LLM non estrae più semplicemente le parole chiave, ma è un interprete affidabile del comportamento avversario.

#### 5.3.2 Il dilemma "Open vs Closed"

Il confronto tra i modelli ha fatto emergere un trade-off chiaro:

I **modelli proprietari** (come **GPT-5** e **Gemini**) sono indispensabili per un'analisi approfondita e accurata: se l'obiettivo è comprendere la sfumatura di una tecnica complessa, il loro costo è giustificato dalle performance superiori.

I **modelli open** (come **Llama 4**), pur avendo un *Recall* inferiore sui comportamenti complessi, dimostrano una rigorosità strutturale superiore nella generazione di **Identity** e **Malware**. L'adozione di Llama 4 in ambiti protetti dimostra che è possibile coniugare la massima privacy dei dati con un'efficacia operativa paragonabile ai modelli proprietari e chiusi, trasformando l'uso di modelli open in un vantaggio strategico.

### 5.3.3 Il problema della "libertà espressiva" dello standard

STIX 2.1 è stato progettato per essere estremamente flessibile, per permettere agli analisti di modellare minacce complesse, tuttavia, questa libertà può rappresentare un'arma a doppio taglio. Lo standard definisce cosa sono gli oggetti, ma offre poche linee guida prescrittive su come debbano essere collegati per rappresentare una data narrazione.

Vediamo un esempio: di fronte alla stessa frase due analisti (in questo caso un esperto CTI e un LLM) potrebbero produrre due grafi validi ma diversi:

- *Analista A*: Crea una relazione diretta **Threat-Actor** *-targets->* **Identity**.
- *Analista B*: Introduce un nodo intermedio, **Threat-Actor** *-uses->* **Campaign** *-targets->* **Identity**.

In questo scenario, la pipeline di valutazione penalizzerà l'LLM nonostante la rappresentazione dell'LLM sia corretta e forse addirittura più ricca. Questa osservazione suggerisce che una parte degli "errori" rilevati non siano vere allucinazioni ma legittime **scelte di modellazione alternative**.

### 5.3.4 Il caso CISA e la divergenza topologica

La criticità di questo fenomeno è stata evidenziata durante la fase di selezione del dataset: i report ufficiali della CISA, pur essendo autorevoli, presentavano una struttura STIX "anomala", dove tutti gli oggetti erano collegati a un nodo centrale di metadati **marking-definition**. Se fossero stati usati quei report come *Ground Truth*, il sistema avrebbe ottenuto punteggi di similarità molto bassi: ciò dimostra che il risultato dell'LLM non è "sbagliato" in assoluto, ma semplicemente **non allineato** allo "stile" usato per i report CISA. In questa situazione, paradossalmente, l'LLM potrebbe creare bundle STIX più conformi alle *Best Practices* rispetto a certi bundle.

### 5.3.5 Limiti attuali ed implicazioni per l'adozione Operativa

Nonostante i successi, la ricostruzione perfetta delle relazioni rimane una sfida aperta, poichè l'LLM comprende i collegamenti tra i nodi, ma fatica a definire la direzione o il tipo di arco con la stessa precisione di un umano. Alla luce di ciò, al netto delle variazioni statistiche, l'adozione di LLM nella CTI non deve mirare all'imitazione dell'analista umano, ma alla

**produzione di intelligence azionabile.** Allo stato attuale l'output del sistema deve essere inteso come un ottimo punto di partenza avanzato per la strutturazione di report dettagliati, un grafo quasi completo che l'analista deve rifinire risparmiando però l'80%-90% del lavoro manuale.

## Capitolo 6

# Conclusioni e sviluppi futuri

Il presente lavoro di tesi ha affrontato una delle sfide più significative nell'attuale panorama della cybersecurity: il divario operativo tra la velocità di produzione delle minacce e la capacità umana di analizzarle, nonché il collo di bottiglia causato dalla necessità di processare manualmente tutti i report di Cyber Threat Intelligence prodotti. L'obiettivo primario era valutare se le recenti evoluzioni nell'Intelligenza Artificiale Generativa potessero automatizzare la conversione di report di Cyber Threat Intelligence non strutturati nello standard STIX 2.1 garantendo un livello di accuratezza compatibile con i requisiti rigidi delle operazioni di sicurezza.

### 6.1 Sintesi del percorso di ricerca

Il percorso si è articolato in fasi distinte:

1. **Analisi del problema:** Si è partiti dalla constatazione che lo standard STIX 2.1, pur essendo un linguaggio per lo scambio di intelligence, soffre di una complessità strutturale che ne frena l'adozione.
2. **Progettazione:** Per superare i limiti di affidabilità degli LLM, è stata progettata l'architettura GenAI-STIX 2.1 Generator, un framework che non si affida ciecamente agli LLM ma orchestra moduli deterministici (Regex per gli IoC tecnici) con moduli basati su LLM implementando tecniche avanzate di *Prompt Engineering*.
3. **Data Curation e Benchmarking:** È stato costruito un dataset basato su report reali dell'NCSC affrontando la scarsità di dati pubblici di qualità. Su questo dataset è stato eseguito un benchmark rigoroso che ha messo a confronto cinque architetture di modelli (famiglie GPT, Llama, Gemini).
4. **Valutazione:** È stata definita una metodologia di valutazione che misura la capacità di *Detection* e *Semantic Similarity* permettendo di valutare i punti di forza e debolezza di ogni modello.

I risultati empirici hanno dimostrato che i modelli *High-Reasoning* (GPT-5, Gemini 2.5) se adeguatamente istruiti tramite tecniche di *Prompt Engineering*, sono in grado di generare

bundle JSON secondo lo standard STIX 2.1 sintatticamente validi e complessi. Hanno inoltre mostrato prestazioni eccezionali nell'estrazione di Tattiche e Tecniche (TTP) raggiungendo una precisione superiore al **94%**: questo aiuta notevolmente gli esperti CTI nella mappatura degli oggetti sul framework MITRE ATT&CK.

## 6.2 Contributi della tesi

Il contributo principale di questo lavoro risiede nella dimostrazione che l'ostacolo all'automazione non è più la "comprensione" del report e della minaccia da parte dell'LLM, ma l'ambiguità intrinseca dello standard STIX. La tesi ha evidenziato come la flessibilità dello standard porti a un'eterogeneità strutturale che complica il confronto automatico, suggerendo che gli LLM potrebbero fungere non solo da generatore, ma anche da **normalizzatore**, imponendo uno stile di modellazione coerente che spesso manca nei dati generati manualmente.

## 6.3 Sviluppi futuri: Verso la "Agentic CTI"

Sebbene i risultati siano promettenti, l'analisi ha evidenziato margini di miglioramento in particolare nella ricostruzione della topologia complessa delle relazioni. Le direzioni di ricerca future per colmare questo gap e abilitare l'adozione degli LLM in compiti sempre più critici si concentrano su due aspetti: il **Fine-Tuning** e le architetture **Multi-Agent**.

### 6.3.1 Specializzazione tramite Fine-Tuning

L'uso di modelli generalisti ha mostrato ottimi risultati, un passo logico successivo è l'addestramento di modelli verticali specifici per la CTI: un modello *fine-tuned* apprenderebbe intrinsecamente lo schema STIX, riducendo la necessità di prompt complessi e eliminando quasi totalmente gli errori di sintassi.

### 6.3.2 Architetture Multi-Agent

La limitazione principale emersa in questo studio è la difficoltà di un singolo modello nel gestire contemporaneamente l'estrazione, il collegamento e la validazione. L'evoluzione naturale è il passaggio a un **Sistema Multi-Agent**, in questo modo il compito verrebbe suddiviso tra agenti specializzati che comunicano tra loro:

- **Agente TTP:** Specializzato nell'estrazione di TTP come **Attack Pattern** e **Malware**.
- **Agente IoC:** Specializzato nell'estrazione di oggetti come **Indicator**.
- **Agente Linker:** Specializzato nella logica delle relazioni costruendo il grafo.
- **Agente Revisore:** Verifica che il grafo prodotto rispetti lo standard STIX chiedendo correzioni agli altri agenti se rileva incongruenze o allucinazioni.

Questo potrebbe introdurre un meccanismo di autocorrezione aumentando esponenzialmente l'affidabilità del sistema.

## 6.4 Conclusioni finali

Questa tesi ha fornito evidenza empirica che l'integrazione dei Large Language Models nella pipeline di Cyber Threat Intelligence non è più una speculazione futuristica, ma una realtà tecnica concreta. Il sistema sviluppato dimostra che l'IA può agire efficacemente come "copilota cognitivo", sollevando gli analisti dall'onere della strutturazione dei dati e permettendo loro di concentrarsi sul vero valore aggiunto: l'interpretazione strategica della minaccia. Con l'avvento di modelli sempre più capaci di ragionamento e l'adozione di architetture a agenti autonomi, ci troviamo all'alba di una nuova era per la CTI, dove la barriera tra il linguaggio umano e il dato strutturato è destinata a dissolversi definitivamente.

# Bibliografia

- [1] Matthew Kosinski. *What is threat intelligence?* URL: <https://www.ibm.com/think/topics/threat-intelligence>.
- [2] Thomas Roccia. *Visual Threat Intelligence An Illustrated Guide for Threat Researchers*. SecurityBreak, 2023.
- [3] Filippo Perrina et al. *AGIR: Automating Cyber Threat Intelligence Reporting with Natural Language Generation*. 2023. arXiv: 2310.02655 [cs.CR]. URL: <https://arxiv.org/abs/2310.02655>.
- [4] paloalto. *What is the Threat Intelligence Lifecycle?* URL: <https://www.paloaltonetworks.com/cyberpedia/what-is-the-threat-intelligence-life-cycle>.
- [5] Francesco Marchiori, Mauro Conti e Nino Vincenzo Verde. “STIXnet: A Novel and Modular Solution for Extracting All STIX Objects in CTI Reports”. In: *Proceedings of the 18th International Conference on Availability, Reliability and Security*. ARES 2023. ACM, ago. 2023, pp. 1–11. DOI: 10.1145/3600160.3600182. URL: <http://dx.doi.org/10.1145/3600160.3600182>.
- [6] Antonio Formato. *From Unstructured Threat Intelligence to STIX 2.1 Bundles with Generative AI*. URL: <https://medium.com/@antonio.formato/from-unstructured-threat-intelligence-to-stix-2-1-bundles-with-generative-ai-1065ce399e63>.
- [7] OASIS. *Introduction to STIX*. URL: [https://oasis-open.github.io/cti-documentation/stix/intro?source=post\\_page--1065ce399e63](https://oasis-open.github.io/cti-documentation/stix/intro?source=post_page--1065ce399e63).
- [8] cyberoo-admin. *Cos'è la Cyber Threat Intelligence e perché è la chiave per la sicurezza informatica*. URL: <https://blog.cyberoo.com/guida-completa-alla-cyber-threat-intelligence>.
- [9] Michael J. Hutchins, Eric M. Cloppert e Rohan M. Skinner. “Intelligence-Driven Computer Network Defense Informed by the Cyber Kill Chain”. In: *Lockheed Martin Corporation* (2011). URL: <https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf>.
- [10] OASIS. *STIX Version 2.1*. URL: [https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html?source=post\\_page-----1065ce399e63](https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html?source=post_page-----1065ce399e63).
- [11] David J. Bianco. *The Pyramid of Pain*. Blog Post. Mar. 2013. URL: <http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html>.

- [12] The MITRE Corporation. *MITRE ATT&CK: Adversarial Tactics, Techniques, and Common Knowledge*. Website. 2024. URL: <https://attack.mitre.org>.
- [13] Wayne Xin Zhao et al. *A Survey of Large Language Models*. 2025. arXiv: 2303.18223 [cs.CL]. URL: <https://arxiv.org/abs/2303.18223>.
- [14] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: <https://arxiv.org/abs/1706.03762>.
- [15] Alec Radford et al. *Improving Language Understanding by Generative Pre-Training*. 2018. URL: [https://cdn.openai.com/research\\_covers/language\\_unsupervised/language\\_understanding\\_paper.pdf](https://cdn.openai.com/research_covers/language_unsupervised/language_understanding_paper.pdf).
- [16] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. URL: <https://arxiv.org/abs/2005.14165>.
- [17] Long Ouyang et al. *Training language models to follow instructions with human feedback*. 2022. arXiv: 2203.02155 [cs.CL]. URL: <https://arxiv.org/abs/2203.02155>.
- [18] Jules White et al. *A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT*. 2023. arXiv: 2302.11382 [cs.SE]. URL: <https://arxiv.org/abs/2302.11382>.
- [19] Jason Wei et al. *Emergent Abilities of Large Language Models*. 2022. arXiv: 2206.07682 [cs.CL]. URL: <https://arxiv.org/abs/2206.07682>.
- [20] Patrick Lewis et al. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. 2021. arXiv: 2005.11401 [cs.CL]. URL: <https://arxiv.org/abs/2005.11401>.
- [21] Ari Holtzman et al. *The Curious Case of Neural Text Degeneration*. 2020. arXiv: 1904.09751 [cs.CL]. URL: <https://arxiv.org/abs/1904.09751>.
- [22] Ziwei Ji et al. “Survey of Hallucination in Natural Language Generation”. In: *ACM Computing Surveys* 55.12 (mar. 2023), pp. 1–38. ISSN: 1557-7341. DOI: 10.1145/3571730. URL: <http://dx.doi.org/10.1145/3571730>.
- [23] Emanuele Mezzi, Fabio Massacci e Katja Tuma. “Large Language Models Are Unreliable for Cyber Threat Intelligence”. In: *Availability, Reliability and Security*. Springer Nature Switzerland, 2025, pp. 343–364. ISBN: 9783032006271. DOI: 10.1007/978-3-032-00627-1\_17. URL: [http://dx.doi.org/10.1007/978-3-032-00627-1\\_17](http://dx.doi.org/10.1007/978-3-032-00627-1_17).
- [24] OpenAI. *OpenAI*. URL: <https://openai.com/>.
- [25] OpenAI. *Introducing GPT-4.1 in the API*. URL: <https://openai.com/index/gpt-4-1/>.
- [26] OpenAI. *GPT-5 is here*. URL: <https://openai.com/gpt-5/>.
- [27] Meta. *Llama*. URL: <https://www.llama.com/>.
- [28] Meta. *Llama 3 The open-source AI models you can fine-tune, distill and deploy anywhere. Choose from our collection of models: Llama 3.1, Llama 3.2, Llama 3.3*. URL: <https://www.llama.com/models/llama-3/>.

- [29] Meta. *The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation*. URL: <https://ai.meta.com/blog/llama-4-multimodal-intelligence/>.
- [30] Google. *Gemini 2.5 Pro*. URL: <https://gemini.google/it/about/?hl=it>.
- [31] Microsoft. *Markdown*. URL: <https://github.com/microsoft/markitdown>.
- [32] OASIS. *STIX 2 Python API Documentation*. URL: <https://stix2.readthedocs.io/en/latest/>.
- [33] National Cyber Security Centre. *Malware analysis reports*. URL: <https://www.ncsc.gov.uk/section/keep-up-to-date/malware-analysis-reports>.
- [34] Cybersecurity e Infrastructure Security Agency. *Cybersecurity Alerts & Advisories*. URL: [https://www.cisa.gov/news-events/cybersecurity-advisories?f%5B0%5D=advisory\\_type%3A94&page=0](https://www.cisa.gov/news-events/cybersecurity-advisories?f%5B0%5D=advisory_type%3A94&page=0).
- [35] Python. *Python*. URL: <https://www.python.org/>.
- [36] Jupyter. *Project Jupiter*. URL: <https://jupyter.org/>.
- [37] OpenAI. *OpenAI API*. URL: <https://platform.openai.com/docs/libraries?language=python>.

