



Corso di Laurea in Informatica

TESI DI LAUREA

Integrazione di Large Language Models (LLM) in Sistemi di
Deep Packet Inspection

Relatore:
Prof. Luca Deri

Candidata:
Andrea Nizzi

ANNO ACCADEMICO 2025/2026

Abstract

L'analisi del traffico di rete è un pilastro della sicurezza informatica, ma l'interpretazione dei dati estratti tramite Deep Packet Inspection (DPI) richiede tempo e competenze specialistiche. La tesi presenta e valuta sperimentalmente un agente basato su un modello di linguaggio di grandi dimensioni (LLM) che, dati un host, una finestra temporale e una categoria di indagine, interroga un database di flussi arricchiti da analisi DPI (nDPI) tramite il Model Context Protocol (MCP). L'agente incrocia le evidenze restituite da tool deterministici ed emette un verdetto motivato tra cinque classi: DoS (Denial of Service) volumetrico, scansione/brute force, beaconing C2 (Command and Control), attacco web, traffico benigno.

L'architettura si basa su una pipeline modulare: elaborazione dei dati (CICFlowMeter, nDPI), memorizzazione (MySQL) e un componente di accesso dinamico basato su MCP. Al fianco dell'LLM opera un motore euristico deterministico (`compute_verdict_scores`), che applica soglie statiche alle metriche estratte e propone un verdetto di riferimento. Il compito dell'LLM è incrociare criticamente tale suggerimento con le evidenze grezze, confermandolo, correggendolo, oppure risolvendo i casi in cui l'euristica stessa segnala un conflitto irrisolto tra categorie.

Il sistema è valutato su 242 scenari (finestre host/15 minuti) costruiti da CIC-IDS-2017, per un totale di 529.268 flussi analizzati (conteggiati per scenario), suddivisi in un test set di 193 scenari (283.813 flussi), usato per tarare soglie e prompt, e un insieme di verifica (holdout set) di 49 scenari (245.455 flussi). Il test set raggiunge un'Accuracy del 98,96% e una Precision del 98,90%; l'holdout set, eseguito 3 volte in modalità guidata, riporta un'Accuracy media del 96,60% e una Precision media del 98,29%. Il confronto con l'euristica locale mostra che l'LLM tende a confermare il verdetto euristico nella maggior parte dei casi, ma è in grado di correggere falsi positivi sistematici (es. traffico periodico verso servizi legittimi scambiato per C2). Allo stesso tempo, il meccanismo di post-euristica del sistema corregge alcuni falsi negativi dell'LLM, forzando il verdetto euristico quando la deroga BENIGN non è ammessa. Il sistema integrato risulta quindi

più robusto di ciascun componente preso singolarmente, come discusso quantitativamente nel Capitolo 7.

Si discutono infine i limiti emersi: un margine di approssimazione nella ground truth aggregata, poiché il confronto tra il verdetto dell’LLM (basato sui record nDPI) e le etichette ufficiali (basate sui record CICFlowMeter) avviene a livello di finestra host/15 minuti e non del singolo flusso, per via dei diversi criteri di terminazione del flusso adottati dai due estrattori; soglie euristiche calibrate su un solo dataset; una valutazione condotta su un solo modello LLM; un numero di scenari ridotto per le categorie DOS, WEB e SCAN; e un non-determinismo residuo su alcuni scenari ambigui, che possono produrre verdeti diversi in esecuzioni diverse.

Indice

Abstract	i
1 Introduzione	1
1.1 Contesto e motivazione	1
1.2 Problema	1
1.3 Obiettivo della tesi	2
1.4 Contributi	2
1.5 Struttura della tesi	4
2 Background	5
2.1 Analisi del traffico di rete	5
2.2 Deep Packet Inspection	5
2.3 Feature extraction e dataset	6
2.4 Sistemi di Intrusion Detection (IDS)	6
2.5 Modelli di linguaggio (LLM)	7

2.6	Integrazione tra LLM e dati strutturati	7
3	Stato dell'arte	10
3.1	Approcci tradizionali alla sicurezza di rete	10
3.2	Machine Learning per l'analisi del traffico	10
3.3	Limiti degli approcci esistenti	11
3.4	Utilizzo di modelli LLM	12
3.5	Posizionamento del lavoro	13
4	Architettura del sistema	15
4.1	Visione generale	15
4.2	Modulo di gestione dei dati	17
4.3	Model Context Protocol	18
4.3.1	Architettura e funzionamento	19
4.3.2	Vantaggi del paradigma MCP	20
4.4	Modello LLM	20
4.5	Pipeline operativa e flusso del dato	21
4.5.1	Fase 1: ingestion e normalization	21
4.5.2	Fase 2: user request	22
4.5.3	Fase 3: context enrichment (MCP Server)	22
4.5.4	Fase 4: reasoning & response	23
5	Scelte progettuali	24
5.1	Introduzione	24
5.2	Scelta del database e ingegnerizzazione dello schema	24
5.2.1	Strategia di identificazione e correlazione dei flussi	25
5.3	Scelta del modello e infrastruttura di inferenza	27
5.3.1	Motivazioni della scelta del modello	27
5.3.2	Infrastruttura di inferenza e provider	28
5.3.3	Alternative considerate	28
5.4	Scelta del dataset	29

5.5	Scelta dell'architettura MCP	30
5.6	Riepilogo delle scelte	31
6	Implementazione	32
6.1	Pipeline di elaborazione	32
6.2	Architettura MCP	34
6.2.1	Tassonomia dei tool MCP e scenari di investigazione	34
6.2.2	Il middleware MCP: architettura e implementazione	38
6.3	Preprocessing e feature engineering	40
6.3.1	Disallineamento temporale e aggregazione dei flussi	43
6.4	Prompt engineering e architettura agentica MCP	44
6.4.1	Architettura del system prompt e tassonomia del verdetto	45
6.4.2	Regole di trasparenza investigativa e logica anti-loop	49
7	Metodologia sperimentale e validazione dei risultati	51
7.1	Obiettivo della validazione	51
7.2	Setup sperimentale	52
7.3	Costruzione del benchmark	53
7.4	Metriche di valutazione	55
7.4.1	Metriche quantitative	55
7.4.2	Limiti metodologici del benchmark	56
7.4.3	Coerenza semantica e grounding	58
7.4.4	Analisi delle prestazioni	58
7.5	Scenari di test (categorie A–E)	59
7.5.1	Categoria A — analisi strutturale e applicativa L7	60
7.5.2	Categoria B — monitoraggio volumetrico	60
7.5.3	Categoria C — profiling endpoint	61
7.5.4	Categoria D — analisi comportamentale	61
7.5.5	Categoria E — analisi forense libera	62
7.5.6	Sintesi delle categorie	62
7.6	Risultati sperimentali	63

7.6.1	Test set (193, guidata): metriche quantitative	63
7.6.2	Confronto LLM vs euristica sul test set	66
7.6.3	Holdout set (49, guidata, 3 run)	66
7.6.4	Holdout set (49, generica)	68
7.6.5	Test su traffico reale (18 finestre)	69
7.6.6	Analisi qualitativa dei disaccordi sull'intera campagna di validazione	69
7.6.7	Analisi aggregata del grounding	70
7.7	Analisi dei risultati	72
7.7.1	Contributo dell'LLM rispetto all'euristica	72
7.7.2	Comportamento su scenari con segnale debole	74
7.7.3	Latenza e scalabilità	75
7.7.4	Sintesi	76
8	Risultati e discussione	77
8.1	Interpretazione dei risultati	77
8.2	Vantaggi dell'approccio	77
8.3	Confronto con approcci tradizionali	78
8.4	Limiti e implicazioni	78
9	Conclusioni e sviluppi futuri	81
9.1	Limiti del sistema	81
9.1.1	Un solo modello LLM e validazione esterna limitata	81
9.1.2	Limiti intrinseci dell'architettura	82
9.2	Contributi della tesi	83
9.3	Sviluppi futuri	83
9.4	Conclusioni e considerazioni finali	84
A	Codice sorgente e risorse	91
A.1	Repository ufficiale del progetto	91
A.2	Estratti di codice significativi	91
A.2.1	Community ID bidirezionale	91

A.2.2	Accoppiamento a tolleranza temporale	92
A.2.3	Isolamento strutturale del blind test	93
A.2.4	Tie-breaking e delega all’LLM	94
A.3	Schema del database	95
B	Allineamento tra ndpi_flows e cic_flows	99
B.1	Logica di popolamento	99
B.2	Verifica empirica del drift	100
B.3	Query di verifica dell’allineamento	101
C	Costruzione del benchmark e dettagli metodologici	103
C.1	Correzione della pipeline di ingestion e scarto degli UNMATCHED	103
C.2	Pipeline di generazione e regole di labeling	104
C.3	Configurazioni e composizione del benchmark	106
C.4	Verifica dell’allineamento ndpi/cic	109
C.5	Distribuzione delle finestre per regime e purezza	112
	Dichiarazione sull’uso di strumenti di IA generativa	116

Elenco delle figure

2.1	Flusso del dato: dal traffico grezzo al verdetto motivato.	8
4.1	Architettura logica del sistema.	16
4.2	Schema del database relazionale.	18
4.3	Pipeline operativa del sistema.	23
6.1	Pipeline di elaborazione.	34
6.2	Flusso di comunicazione tra i componenti del sistema.	38

Elenco delle tabelle

5.1	Riepilogo delle scelte progettuali.	31
6.1	Campi delle tabelle cic_flows e ndpi_flows.	42
6.2	Tassonomia delle cinque classi di verdetto.	45
6.3	Soglie numeriche principali utilizzate dal motore euristico.	45
6.4	Regole di priorità per la determinazione del verdetto finale.	48
7.1	Sintesi delle cinque categorie di indagine.	63
7.2	Metriche globali di classificazione (193 scenari, 283.813 flussi).	64
7.3	Matrice di confusione (193 scenari, 5 classi).	64
7.4	Metriche di classificazione per categoria e supporto (193 scenari).	65
7.5	Confronto tra verdetto dell’LLM e verdetto dell’euristica (193 scenari).	66
7.6	Metriche di classificazione sull’holdout set (49 scenari, 245.455 flussi).	67
7.7	Metriche per categoria (holdout, media su 3 run).	67
7.8	Metriche di classificazione sull’holdout set (49 scenari, modalità generica).	68
7.9	Metriche per categoria (holdout, modalità generica).	68
7.10	Falsi positivi su traffico reale.	69
7.11	Classificazione dei disaccordi LLM vs euristica sull’intera campagna di validazione (260 scenari).	70
7.12	Analisi aggregata del grounding (407 run).	71
7.13	Tipi di disaccordo tra LLM ed euristica.	73
7.14	Latenza mediana per set (in secondi).	75
8.1	Confronto con approcci tradizionali.	78
9.1	Sintesi dei limiti del sistema.	82

A.1	Struttura repository GitHub del progetto.	91
C.1	Soglie per generazione del benchmark.	106
C.2	Composizione del benchmark (242 scenari).	108
C.3	Distribuzione label CIC per categoria (242 scenari).	109
C.4	Allineamento ndpi/cic per categoria.	110
C.5	Distribuzione delle finestre per regime (242 scenari).	113
C.6	Distribuzione della purezza sui 242 scenari.	113
C.7	Distribuzione dei regimi per categoria (242 scenari).	114

Listings

A.1	Calcolo del Community ID.	91
A.2	Accoppiamento a tolleranza temporale.	93
A.3	Isolamento strutturale del blind test.	93
A.4	Tie-breaking e delega all’LLM.	94
A.5	Schema del database.	95
B.1	Logica di popolamento flow mapping.	99
B.2	Query di verifica dell’allineamento.	101

1. Introduzione

1.1 Contesto e motivazione

Negli ultimi anni, la crescente complessità delle infrastrutture di rete e l'aumento esponenziale del traffico dati hanno reso sempre più difficile l'analisi manuale dei flussi di comunicazione [1]. Tecniche avanzate come la DPI consentono di estrarre informazioni dettagliate dai pacchetti di rete, ma la loro applicazione genera volumi di dati strutturati la cui interpretazione richiede tempo e competenze specialistiche [2].

Nel dominio della sicurezza informatica, l'identificazione tempestiva di comportamenti anomali, come attacchi di negazione del servizio, scansioni, brute force o comunicazioni di botnet, è diventata critica. Risulta pertanto necessario disporre di strumenti che, oltre a elaborare grandi volumi di dati, ne producano una rappresentazione interpretabile da parte degli operatori di sicurezza.

1.2 Problema

Nonostante i progressi compiuti, gli approcci tradizionali all'analisi della sicurezza di rete mostrano limiti intrinseci che ne ostacolano l'efficacia in scenari operativi complessi. Tali limiti possono essere categorizzati come segue:

1. **Limiti dei sistemi basati su regole (rule-based):** I sistemi di rilevamento basati su firme o regole statiche, pur essendo efficienti nell'identificare minacce note, mostrano una scarsa capacità di generalizzazione verso attacchi zero-day o varianti sconosciute. Come evidenziato da Sommer e Paxson [3], l'assunzione di un ambiente chiuso e prevedibile, tipica di questi sistemi, non regge di fronte all'evoluzione dinamica delle minacce informatiche.
2. **Complessità e opacità dei modelli di Machine Learning (ML):** L'adozione di modelli predittivi basati su ML introduce nuove criticità. Da un lato, richiede un'at-

tenta fase di feature engineering, che è spesso onerosa e richiede competenze specialistiche. Dall'altro, come sottolineato da Arp et al. [4], molti modelli di ML, in particolare quelli basati su Deep Learning, operano come “scatole nere” (black-box), fornendo previsioni accurate ma prive di una spiegazione semantica che possa guidare l'analista nella comprensione della minaccia.

3. **Necessità di correlazione manuale delle evidenze:** La determinazione della natura di un'anomalia richiede tipicamente la correlazione di molteplici fonti di dati (es. volumi di traffico, porte, protocolli applicativi, pattern di periodicità). Questa attività di analisi è intrinsecamente manuale, richiede tempo e si basa su competenze specialistiche, creando un collo di bottiglia nel processo di risposta agli incidenti.

In sintesi, esiste un divario tra la disponibilità del dato grezzo (estratto tramite DPI) e la capacità dell'operatore di trasformarlo rapidamente in informazioni azionabili.

1.3 Obiettivo della tesi

L'obiettivo è progettare e valutare sperimentalmente un agente basato su LLM [5] per la classificazione motivata del traffico di rete a partire da dati strutturati estratti tramite DPI. Il modello non è usato come base di conoscenza, ma come agente che, tramite MCP [6], richiama tool deterministici e ne incrocia i risultati. Nello specifico:

- classificazione a cinque classi con verdetto e motivazione basata sulle evidenze;
- valutazione rispetto alla ground truth di CIC-IDS-2017 e a un classificatore a regole, per isolare il contributo del ragionamento del modello;
- analisi degli errori e dei limiti del metodo.

1.4 Contributi

Questo lavoro si configura come una tesi sperimentale, il cui obiettivo non è proporre un sistema definitivo, ma indagare i limiti e le potenzialità dell'impiego di un LLM in ambito

di sicurezza informatica, a confronto con gli approcci classici basati su regole o modelli di classificazione statici. I principali contributi sono:

- **Progettazione di un'architettura modulare a scopo sperimentale.** L'architettura è concepita per essere model-agnostic a livello strutturale: il modello LLM può in linea di principio essere sostituito senza modificare la logica di analisi del traffico. In questa fase sperimentale, tuttavia, i prompt sono stati progettati e calibrati specificamente per Qwen 3.8 27B. La loro efficacia con altri modelli non è stata verificata: l'unica alternativa configurata (GPT-OSS 120B via Groq) si è rivelata inutilizzabile per i limiti di token del tier gratuito (cfr. §5.3 e §9.1.1).
- **Sperimentazione dell'integrazione dinamica tramite MCP.** Implementazione di un ponte comunicativo tra modelli LLM e database MySQL, per verificare se questo approccio possa mitigare i limiti di aggiornamento tipici dei modelli tradizionali e garantire un accesso dinamico a dati strutturati senza riaddestramento.
- **Valutazione sperimentale su dataset standardizzati.** Test del sistema sul dataset CIC-IDS-2017 [7], per misurarne la precisione rispetto alla ground truth e discuterne criticamente il comportamento rispetto a un classificatore a soglie statiche. La valutazione isola il contributo del ragionamento dell'LLM da quello della sola euristica (§7.6.2), analizzando i casi in cui il modello corregge, conferma o risolve i conflitti del motore deterministico (§7.6.6).
- **Analisi critica dei limiti metodologici.** Discussione esplicita dei limiti della validazione: il confronto tra verdetto LLM e ground truth avviene a livello di finestra host/15 minuti (non di singolo flusso), per via dei diversi criteri di terminazione dei flussi adottati dai due estrattori; le soglie euristiche sono calibrate su un solo dataset; la validazione è condotta su un solo modello LLM; e il numero di scenari per le categorie DOS, WEB e SCAN è molto ridotto. Tali limiti sono discussi in dettaglio nei Capitoli 7 e 8 e costituiscono la base per gli sviluppi futuri delineati nel Capitolo 9.

1.5 Struttura della tesi

Il presente elaborato è organizzato come segue:

- **Capitolo 2 e 3:** forniscono il background teorico sul networking, la DPI e lo stato dell'arte delle tecnologie LLM applicate alla sicurezza.
- **Capitolo 4:** descrive l'architettura logica del sistema proposto.
- **Capitolo 5:** analizza le scelte progettuali, dalle tecnologie di inferenza al dataset selezionato.
- **Capitolo 6 e 7:** dettagliano l'implementazione tecnica e la successiva validazione sperimentale dei risultati.
- **Capitolo 8 e 9:** discutono i limiti del sistema e traggono le conclusioni sulle prospettive future.

2. Background

2.1 Analisi del traffico di rete

L'analisi del traffico di rete consiste nell'osservazione e nell'elaborazione dei pacchetti che transitano all'interno di una rete, con l'obiettivo di comprenderne il comportamento e identificare eventuali anomalie o minacce.

Le informazioni possono essere raccolte a diversi livelli:

- **Livello di pacchetto (packet-level):** analisi dettagliata dei singoli pacchetti e dei relativi header.
- **Livello di flusso (flow-level):** analisi aggregata di sequenze di pacchetti con caratteristiche comuni.
- **Livello applicativo:** analisi dei dati relativi ai servizi specifici (es. HTTP, DNS, TLS).

L'analisi può essere effettuata sia in tempo reale sia offline, a seconda del contesto applicativo.

2.2 Deep Packet Inspection

La DPI è una tecnica evoluta che consente di esaminare il contenuto informativo dei pacchetti (payload) oltre alle informazioni contenute negli header standard.

nDPI è una libreria open-source scritta in C per la classificazione del traffico a livello applicativo (L7), sviluppata come evoluzione di OpenDPI e mantenuta dal progetto ntop [8]. A differenza delle tecniche di DPI tradizionali, che richiedono l'ispezione del payload, nDPI combina pattern matching su header e metadati con analisi statistica, risultando efficace anche su traffico cifrato. Questa tecnica è fondamentale per classificare il traffico in base alla natura del servizio (es. streaming, VoIP) e per rilevare minacce nascoste nei dati applicativi.

Tuttavia, la DPI presenta sfide significative, tra cui l'elevato costo computazionale e la diffusione della cifratura, che rende necessaria l'analisi di metadati statistici.

2.3 Feature extraction e dataset

Per automatizzare l'analisi, i dati grezzi di rete devono essere trasformati in feature rappresentative. In questa fase, i flussi vengono sintetizzati in vettori numerici che includono parametri come la durata della connessione, la dimensione dei pacchetti e l'Inter-Arrival Time (IAT). Quest'ultimo, in particolare, è considerato una delle feature più discriminanti per la classificazione del traffico: la letteratura ne estrae statistiche come minimo, massimo, media e deviazione standard degli intervalli tra pacchetti consecutivi, che si sono dimostrate efficaci nel distinguere applicazioni e sessioni con pattern temporali diversi [9].

CICFlowMeter è uno strumento open-source sviluppato presso il Canadian Institute for Cybersecurity (CIC) per l'estrazione automatica di feature statistiche da flussi di rete (durata, byte rate, Inter-Arrival Time, flag TCP) a partire da file PCAP. Esso costituisce il riferimento metodologico del dataset CIC-IDS-2017 [7], le cui etichette sono state prodotte proprio con questo strumento. Parallelamente, per l'analisi del livello applicativo, si utilizzano tool di Deep Packet Inspection come nDPI, che identificano protocolli e servizi anche su traffico cifrato.

L'impiego di dataset pubblici etichettati, come quelli del Canadian Institute for Cybersecurity, è fondamentale per la fase di validazione, poiché questi dataset forniscono un riferimento oggettivo (ground truth) che permette di misurare l'accuratezza del sistema nel distinguere correttamente categorie di traffico differenti, come flussi benigni, connessioni VPN o tentativi di intrusione.

2.4 Sistemi di Intrusion Detection (IDS)

Gli IDS sono strumenti progettati per identificare attività malevole o violazioni delle policy di sicurezza. In letteratura [10], questi sistemi vengono classificati in base al

posizionamento:

- **Network-based IDS (NIDS):** monitorano l'intero traffico di un segmento di rete.
- **Host-based IDS (HIDS):** monitorano i log e le attività di un singolo dispositivo.

E al metodo di rilevazione:

- **Signature-based:** confronto con pattern di attacchi noti.
- **Anomaly-based:** identificazione di deviazioni rispetto a una baseline di comportamento normale (utile per attacchi zero-day).

Nonostante l'efficacia dei NIDS moderni, uno dei limiti principali rimane la difficoltà per l'operatore umano di interpretare rapidamente i log generati. In questo scenario, l'integrazione di interfacce basate su LLM si propone come soluzione per colmare il divario tra il rilevamento tecnico dell'IDS e l'analisi decisionale.

2.5 Modelli di linguaggio (LLM)

I modelli di linguaggio di grandi dimensioni (LLM) sono basati sull'architettura Transformer [11]. In questa tesi il modello opera come agente: riceve un compito, richiama strumenti esterni (tool calling), legge i risultati e ragiona su di essi fino a emettere un verdetto motivato. Non analizza pacchetti né genera query SQL: le metriche sono calcolate da tool deterministici, e al modello spettano l'ordine dell'indagine, l'incrocio critico delle evidenze e la sintesi testuale. Poiché il verdetto dipende dal suo ragionamento, l'LLM è valutato come classificatore (Cap. 7) e confrontato con un classificatore a regole che usa le stesse evidenze.

2.6 Integrazione tra LLM e dati strutturati

Un limite critico degli LLM è il cosiddetto *knowledge cutoff*, ovvero l'impossibilità di accedere a dati successivi al loro addestramento. Per superare tale limite e ridurre le allucinazioni, è necessario integrare i modelli con fonti di dati esterne (es. database MySQL).

In questo lavoro si adotta il Model Context Protocol [6], un layer di astrazione che permette all'LLM di interagire dinamicamente con strumenti e database esterni. Questa integrazione permette di unire le capacità di ragionamento linguistico con i dati memorizzati, costituendo il fulcro tecnologico del sistema proposto.

L'integrazione di LLM con dati strutturati richiede un meccanismo di accesso sicuro e controllato. Il MCP è un protocollo standardizzato che consente a un LLM di invocare tool predefiniti esposti da un server, senza generare direttamente query SQL. Questo approccio garantisce sia la sicurezza (nessuna SQL injection, accesso controllato alle tabelle) sia la tracciabilità delle operazioni.

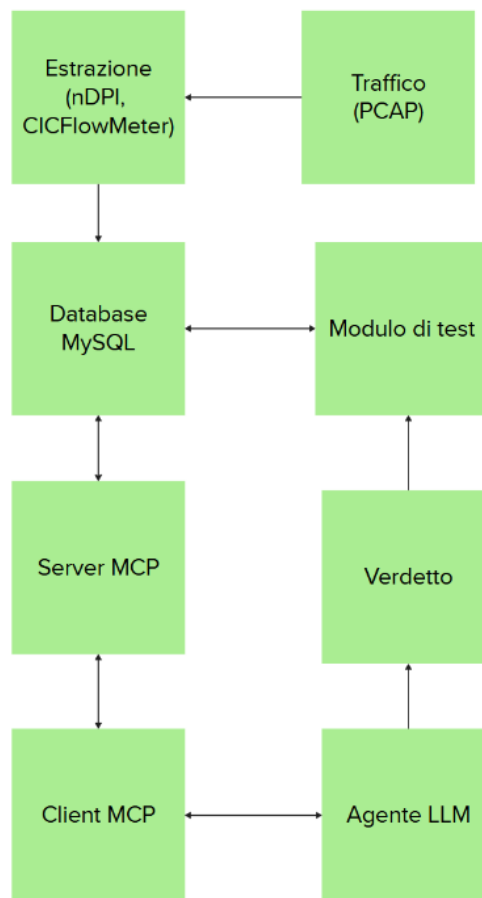


Figura 2.1: Flusso del dato: dal traffico grezzo al verdetto motivato.

Il traffico (PCAP) viene processato in parallelo da nDPI e CICFlowMeter; i risultati sono caricati in MySQL nelle tabelle `ndpi_flows` (privi di etichette) e `cic_flows` (con

etichette ufficiali, ground truth). L'agente LLM interroga il database esclusivamente tramite il server MCP, accedendo alla sola tabella `ndpi_flows`; tra LLM e server MCP opera il client asincrono, che media la comunicazione e applica i controlli. Il modulo di test confronta il verdetto dell'LLM con la ground truth letta da `cic_flows`, producendo l'esito (TP/TN/FP/FN). La tabella `cic_flows` non è accessibile all'LLM.

3. Stato dell'arte

3.1 Approcci tradizionali alla sicurezza di rete

L'analisi della sicurezza di rete è tradizionalmente basata su sistemi di rilevamento delle intrusioni (IDS) e tecniche di monitoraggio del traffico. I principali approcci si dividono in due macro-categorie:

- **Sistemi basati su firme (signature-based):** identificano attacchi noti confrontando il traffico con pattern predefiniti. Tali sistemi risultano molto efficaci per minacce già conosciute, ma non sono in grado di rilevare attacchi nuovi o varianti sconosciute (zero-day).
- **Sistemi basati su anomalie (anomaly-based):** identificano deviazioni dal comportamento normale della rete. Sebbene questo approccio consenta di individuare minacce non note, introduce storicamente un elevato numero di falsi positivi [3].

3.2 Machine Learning per l'analisi del traffico

Negli ultimi anni, l'integrazione del Machine Learning (ML) ha permesso di superare la rigidità dei sistemi basati su regole, esplorando diverse direzioni:

- **Classificazione del traffico:** modelli di apprendimento supervisionato sono stati ampiamente utilizzati per distinguere diverse tipologie di flussi, come nel caso della separazione tra traffico standard e traffico VPN, fondamentale per il monitoraggio delle policy aziendali [12].
- **Rilevamento di minacce complesse:** la letteratura scientifica ha ampiamente dimostrato come l'analisi statistica dei flussi, supportata da feature avanzate estratte tramite DPI, sia essenziale per identificare comportamenti malevoli che sfuggono ai controlli puntuali. Un esempio critico è rappresentato dalle botnet distribuite: Bot-Miner ha dimostrato come le loro attività di comunicazione e coordinamento (canali

di Command & Control) possano essere ricostruite attraverso una correlazione dei dati di traffico su larga scala, indipendente dal protocollo e dalla struttura specifica del malware [13]. Tali capacità analitiche risultano determinanti per il rilevamento di minacce estremamente sofisticate, come le scansioni “stealth” dell’intero spazio di indirizzamento IPv4. In questi scenari, gli attaccanti operano in modo frammentato e con tempistiche dilatate per evitare di innescare i sistemi di rilevamento locali; solo un monitoraggio avanzato, capace di aggregare e analizzare flussi provenienti da diverse porzioni della rete, permette di ricostruire l’attività globale dell’attaccante e neutralizzare l’intrusione [14].

- **Deep Learning (DL):** approcci più recenti basati su reti neurali profonde consentono di apprendere automaticamente rappresentazioni spazio-temporali dei dati di rete, migliorando le prestazioni rispetto ai metodi classici, ma introducendo una maggiore opacità decisionale [15].

3.3 Limiti degli approcci esistenti

Nonostante la maggiore capacità di generalizzazione, gli approcci basati su ML/DL presentano criticità strutturali per i Security Operation Center (SOC) [4]:

1. **Necessità di Feature Engineering:** i modelli richiedono dati pre-processati e trasformati manualmente, un processo costoso e non sempre adattabile in tempo reale.
2. **Difficoltà di interpretazione:** i modelli di Deep Learning spesso agiscono come “black box”, fornendo alert senza una spiegazione semantica che permetta all’operatore di comprendere la natura dell’attacco.
3. **Barriere tecniche:** l’interazione con i dati grezzi o strutturati richiede la conoscenza di linguaggi tecnici (SQL, Python), creando un divario tra la disponibilità del dato e la sua fruibilità da parte di analisti non esperti.

3.4 Utilizzo di modelli LLM

Di recente, i LLM hanno mostrato capacità avanzate nella comprensione di dati strutturati e nella generazione di codice, rendendoli promettenti per la cybersecurity [16].

Un'evoluzione particolarmente rilevante è rappresentata dai sistemi di Retrieval-Augmented Generation (RAG), che combinano le capacità linguistiche del modello con l'accesso a fonti di conoscenza esterne, riducendo il fenomeno delle allucinazioni e garantendo risposte basate su fatti documentati [17].

Nel dominio specifico della sicurezza di rete, diversi lavori recenti hanno esplorato l'impiego di LLM per l'analisi di log e il triage in ambito SOC: HuntGPT integra un classificatore di anomalie con un modello conversazionale per rendere gli alert comprensibili all'analista [18]; IDS-Agent propone, in modo analogo all'approccio qui adottato, un agente LLM che invoca strumenti esterni per condurre un'indagine di intrusion detection spiegabile su reti IoT [19]; un benchmark sistematico ha inoltre valutato le capacità di diversi LLM nell'interpretazione di log di sicurezza [20].

A partire dal 2025, la letteratura ha iniziato a sistematizzare il campo degli agenti LLM per la cybersecurity, con survey che ne analizzano workflow, rischi e capacità offensive: [21] fornisce un survey critico sugli LLM per la difesa da cyberattacchi, [22] fornisce una tassonomia dei workflow difensivi (SOC, incident response) e dei rischi sistemici, mentre [23] analizza le capacità offensive degli agenti LLM, evidenziando come l'integrazione di LLM con API e strumenti possa ampliare la superficie di attacco. Parallelamente, [24] propone un framework di trustworthiness per agenti autonomi, identificando gap di sicurezza critici come l'iniezione indiretta di prompt e l'avvelenamento di memoria vettoriale. Nel dominio specifico del networking, [25] esplora l'adattamento di LLM a task di decision-making, mentre [26] propone un modello generativo pre-addestrato per la comprensione e la generazione di traffico di rete.

Una review sistematica condotta su 167 studi peer-reviewed (2022–2025) propone una tassonomia unificata threat-defence-evaluation e identifica nella security operations automation uno dei domini applicativi più promettenti [27]. Il lavoro qui presentato si colloca in questo filone, distinguendosi per due aspetti: (i) non addestra né adatta il modello,

ma lo usa come agente con tool calling su dati strutturati; (ii) valuta il contributo del ragionamento rispetto a un motore euristico deterministico su ground truth indipendente.

Tuttavia, nel monitoraggio di rete, l'accesso a dati dinamici (come i flussi in tempo reale) richiede un'integrazione ancora più profonda rispetto al RAG tradizionale. Ciò ha portato all'emergere di protocolli di comunicazione tra modelli e database come il MCP [6], che permette di impiegare gli LLM come interfacce per interrogare dati strutturati in linguaggio naturale.

3.5 Posizionamento del lavoro

Questo lavoro si colloca nel filone degli agenti LLM per la sicurezza di rete. A differenza dei classificatori ML/DL, che apprendono la decisione da dati etichettati, il sistema non addestra alcun modello: tool deterministici estraggono metriche aggregate e l'LLM le incrocia secondo istruzioni esplicite ed emette un verdetto motivato. Un classificatore a regole (`compute_verdict_scores`) svolge un doppio ruolo, di baseline e di controllo contro verdetti incoerenti. Il contributo non è un nuovo algoritmo di detection, ma la valutazione sperimentale, con ground truth su scenari CIC-IDS-2017, di quanto l'agente aggiunga o tolga rispetto alle sole regole, con analisi degli errori.

Il dataset CIC-IDS-2017, già introdotto nel Capitolo 2, è diventato uno standard de facto per la valutazione di sistemi di intrusion detection. Studi recenti lo utilizzano ampiamente per il benchmarking di modelli ML/DL, riportando accuratèzze superiori al 98% [28]. Questo lo rende un riferimento oggettivo e comparabile per misurare le prestazioni del sistema proposto.

In questo senso il lavoro è vicino a proposte come IDS-Agent [19], che adotta anch'esso il paradigma dell'agente LLM dotato di strumenti esterni anziché quello del classificatore fine-tuned; la differenza principale riguarda il dominio (IoT vs. traffico di rete generico), l'uso esplicito di un motore euristico a soglie come baseline di controllo, e la valutazione su un dataset con ground truth indipendente dal sistema (CIC-IDS-2017) anziché su traffico simulato ad hoc.

Il sistema combina la precisione dei dati strutturati estratti tramite DPI, un sistema di accesso dinamico tramite MCP e modelli LLM come interfaccia di interrogazione. Questa combinazione mira a colmare il divario di interpretabilità evidenziato dalla letteratura recente, trasformando query astratte in linguaggio naturale in analisi tecniche precise sui flussi di rete.

4. Architettura del sistema

4.1 Visione generale

Il sistema proposto per l'analisi del traffico di rete è basato su un'architettura modulare progettata per integrare LLM con dati strutturati derivanti da tecniche di DPI. L'obiettivo principale consiste nel consentire l'interrogazione dei dati di rete attraverso il linguaggio naturale, mantenendo al tempo stesso l'affidabilità e la precisione degli strumenti tradizionali di analisi [3][29].

L'architettura complessiva, illustrata in Figura 4.1, è composta da quattro componenti principali:

- **Modulo di gestione dei dati**, responsabile della raccolta, del preprocessing e della memorizzazione dei dati di rete;
- **Database relazionale MySQL**: utilizzato per l'archiviazione strutturata delle feature di rete attraverso un'architettura multi-tabella, mirata a separare e normalizzare le metriche estratte dai diversi analizzatori (CICFlowMeter e nDPI);
- **Sistema di accesso basato su MCP**, che permette al modello LLM di interrogare dinamicamente il database e ottenere informazioni pertinenti rispetto alla richiesta dell'utente;
- **LLM servito tramite API esterna**, incaricato di comprendere la query in linguaggio naturale e generare una risposta contestualizzata sulla base dei dati recuperati.

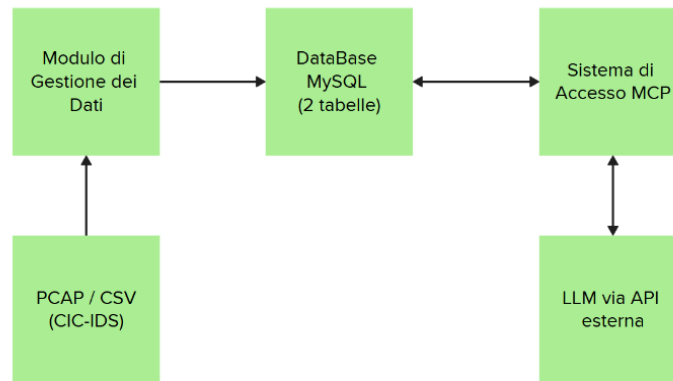


Figura 4.1: Architettura logica del sistema.

Quattro componenti principali: (1) modulo di gestione dei dati, che raccoglie, pre-processa e memorizza i flussi; (2) database MySQL, con architettura multi-tabella (`cic_flows/ndpi_flows`); (3) sistema di accesso basato su MCP (che accorpa client e server MCP per semplicità di rappresentazione; la separazione è dettagliata nella Figura 6.2); (4) LLM servito tramite API esterna.

La separazione tra livello dati, livello di accesso e livello di inferenza garantisce maggiore modularità, scalabilità e manutenibilità del sistema. Tale approccio consente inoltre la sostituzione o l'aggiornamento di singoli componenti senza modificare l'intera pipeline applicativa [30]. In particolare, l'architettura è stata progettata secondo un principio *model-agnostic*: il livello di inferenza è disaccoppiato dal resto della pipeline tramite un client che seleziona il provider LLM a runtime, e la stessa infrastruttura di raccolta ed esposizione dei dati (DB + MCP) resta invariata indipendentemente dal modello utilizzato. Il grado in cui questo agnosticismo è stato verificato sperimentalmente è discusso in dettaglio nel §5.3.

Il flusso operativo del sistema verrà descritto in dettaglio nella Sezione 4.5.

4.2 Modulo di gestione dei dati

Il modulo di gestione dei dati ha il compito di trasformare i flussi grezzi di rete in una rappresentazione compatibile con il modello linguistico. Esso costituisce il livello di acquisizione e strutturazione dell'informazione.

Le principali fasi operative sono le seguenti:

Raccolta dei dati. Il sistema acquisisce traffico di rete da dataset pubblici etichettati, utilizzati frequentemente nella ricerca accademica in ambito cybersecurity. Nello specifico, viene impiegato il dataset CIC-IDS-2017 prodotto dal Canadian Institute for Cybersecurity, che include file PCAP di traffico sia benigno sia contenente scenari di attacco complessi [7].

Preprocessing. I dati raccolti vengono sottoposti a una fase di preprocessing finalizzata alla riduzione del rumore e all'estrazione delle caratteristiche rilevanti. In particolare vengono considerate informazioni quali: protocollo di trasporto; indirizzi IP e porte; durata delle sessioni; numero di pacchetti e byte scambiati; frequenza delle comunicazioni; flag TCP; statistiche temporali sul flusso. Le feature selezionate vengono successivamente normalizzate e convertite in formato tabellare, così da poter essere utilizzate nei passaggi successivi della pipeline.

Memorizzazione. I dati preprocessati vengono memorizzati in un database relazionale MySQL [31]. A differenza di un approccio a tabella singola, lo schema è stato ingegnerizzato per accogliere due tabelle distinte (`cic_flows` e `ndpi_flows`), ottimizzando i tipi di dato (es. `TIMESTAMP(6)` per la precisione al microsecondo e `VARCHAR(45)` per il supporto nativo a IPv4/IPv6) e risolvendo le discrepanze strutturali tra gli estrattori.

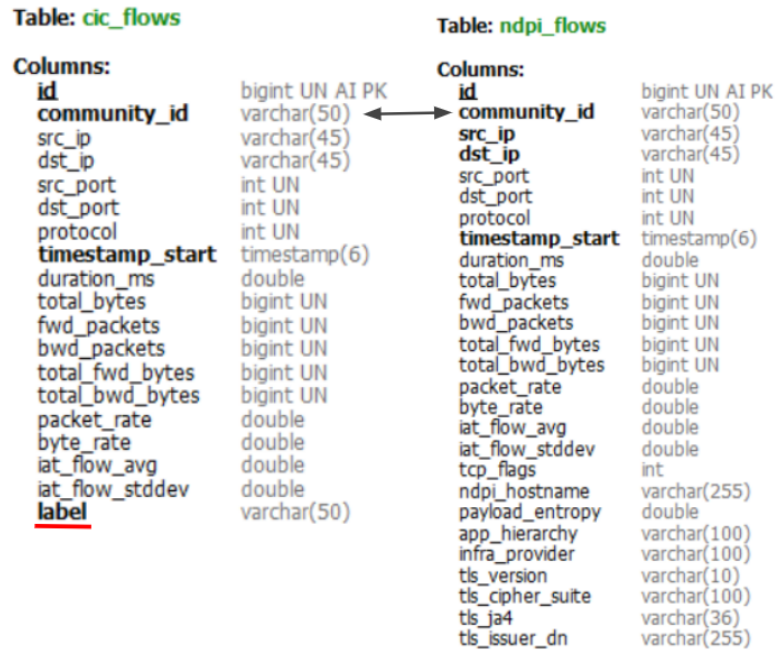


Figura 4.2: Schema del database relazionale.

Le due tabelle `cic_flows` e `ndpi_flows` sono correlate logicamente tramite `community_id` (chiave di join). La colonna `label`, presente solo in `cic_flows`, costituisce la ground truth ed è inaccessibile all'LLM: viene usata esclusivamente dal modulo di auditing per il confronto con il verdetto. La tabella `ndpi_flows` contiene le feature semantiche estratte da nDPI (protocollo L7, SNI, entropia del payload, provider infrastrutturale) ed è l'unica sorgente accessibile ai tool MCP.

4.3 Model Context Protocol

A differenza delle architetture di function calling proprietarie, in cui l'insieme dei tool disponibili è definito staticamente lato client e la loro scoperta richiede una riconfigurazione manuale, l'MCP adotta un modello di dynamic tool discovery: il server espone i propri descrittori di capacità in modo che il client li riceva automaticamente all'apertura della sessione [32]. Inoltre, l'MCP non si limita a veicolare dati grezzi: esso opera come un layer di mediazione che struttura l'accesso alle fonti e incapsula il dato grezzo di rete in un Context Wrapper semanticamente arricchito per l'LLM.

4.3.1 Architettura e funzionamento

Il protocollo abilita un'interazione strutturata suddivisa in tre fasi logiche, che permettono al modello di operare come un analista SOC:

1. **Interpretazione dell'Intento:** all'interno della categoria di indagine selezionata, l'LLM riceve la query in linguaggio naturale e, invece di generare codice SQL complesso soggetto a errori, invoca strumenti specifici (tool) esposti dal server.
2. **Astrazione e Retrieval:** il server MCP espone un insieme predefinito di tool, ciascuno associato a una query SQL parametrizzata e già ottimizzata verso il database; il modello LLM si limita a selezionare il tool più appropriato e a fornirne i parametri, senza generare direttamente codice SQL. Questa scelta progettuale limita la superficie di attacco (evitando, ad esempio, injection o accessi non previsti al database) e garantisce al contempo l'isolamento della logica applicativa dall'inferenza linguistica.
3. **Context & Prompt Wrapping:** i risultati estratti dal database vengono opportunamente contestualizzati e strutturati prima della restituzione, arricchiti con un campo `sintesi_smart` che aggiunge metadati interpretativi calcolati lato server (es. diagnosi preliminare, flag di anomalia). Questo strato riduce drasticamente il carico cognitivo del modello e il rischio di allucinazioni [17].

La scelta di MCP come layer di mediazione è supportata anche dalla documentazione tecnica di Microsoft Learn, che identifica tre vantaggi chiave per gli agenti LLM: dynamic tool discovery (gli agenti ricevono automaticamente la lista dei tool disponibili dal server), interoperabilità tra LLM (lo stesso server MCP può servire modelli diversi senza modifiche), e sicurezza standardizzata (metodo di autenticazione coerente su più server MCP) [32]. Questi vantaggi sono coerenti con le motivazioni architetturali discusse in questo capitolo.

4.3.2 Vantaggi del paradigma MCP

L'adozione di questo paradigma, ispirato alla Retrieval-Augmented Generation, offre vantaggi determinanti per il monitoraggio di rete:

- **Grounding dei dati:** il modello basa le sue conclusioni su prove documentali (i flussi reali) e non sulla conoscenza statica del training set.
- **Scalabilità Informativa:** il sistema si adatta a nuovi dataset o cambiamenti nello schema del database senza necessità di riaddestramento, aggiornando semplicemente la logica dei tool MCP.
- **Sicurezza e Tracciabilità:** ogni accesso al database è mediato da tool predefiniti, permettendo un auditing completo delle operazioni eseguite dall'IA sui dati sensibili di rete.

Formalmente, il comportamento del sistema può essere descritto dalla funzione:

$$Sistema(q) = G(API(R(q))) \quad (4.1)$$

dove q rappresenta la query dell'utente, R è la funzione di retrieval implementata dal server MCP che interroga il database, API rappresenta il servizio di inferenza e G la funzione generativa del modello. Questa decomposizione evidenzia come il successo dell'analisi dipenda non solo dalla potenza del modello linguistico, ma dalla qualità del recupero informativo (R) garantito dal protocollo MCP.

4.4 Modello LLM

Nota metodologica sulla scelta del modello. Il client seleziona a runtime il provider e il modello LLM tra due opzioni configurate — Qwen 3.8 27B, erogato tramite un endpoint dedicato messo a disposizione dal relatore, e GPT-OSS 120B, erogato tramite Groq — senza richiedere modifiche alla pipeline dati o alla logica del server MCP. I motivi per cui la sperimentazione si è concentrata su Qwen sono discussi in dettaglio nel §5.3.

Qwen 3.8 27B è un modello linguistico di grandi dimensioni sviluppato da Alibaba Cloud, con 27 miliardi di parametri [33]. GPT-OSS 120B è un modello open-source di dimensioni maggiori (120 miliardi), erogato tramite la piattaforma Groq. Entrambi sono compatibili con l'API OpenAI, il che ne consente l'intercambiabilità a livello architetturale.

Il modello viene impiegato per: comprendere richieste formulate in linguaggio naturale; interpretare i dati recuperati dal database tramite MCP; effettuare inferenze contestuali basate sulle evidenze raccolte; generare un report forense finale con verdetto giustificato, secondo una tassonomia a cinque classi (cfr. §6.4).

L'utilizzo tramite API permette di selezionare dinamicamente il provider in base alla disponibilità e ai vincoli operativi, mantenendo elevata flessibilità architetturale. L'infrastruttura di inferenza (provider, endpoint, scelta del modello) è discussa in dettaglio nel §5.3.

4.5 Pipeline operativa e flusso del dato

L'architettura del sistema si basa su una pipeline operativa che trasforma i dati grezzi di rete in valutazioni di sicurezza motivate attraverso quattro fasi principali. La Figura 4.3 riassume graficamente questo flusso.

4.5.1 Fase 1: ingestion e normalization

Il processo ha inizio con il reperimento dei file PCAP e dei relativi file CSV ufficiali distribuiti con il dataset, i quali si presentano originariamente in forma frammentata. Questa fase si occupa di:

- **Preprocessing dei CSV ufficiali:** accorpamento e bonifica dei file CSV nativi del dataset (per Giovedì e Venerdì, distribuiti in più file per giornata), con rimozione di header duplicati e gestione dei valori Infinity/NaN.
- **Estrazione locale:** i PCAP vengono elaborati in parallelo da CICFlowMeter (feature statistiche) e da ndpiReader (feature semantiche di livello 7).

- **Join e arricchimento:** i flussi estratti localmente con `CICFlowMeter` vengono correlati con i CSV ufficiali per iniettare la colonna `Label` (la classificazione dell'attacco).
- **Normalizzazione, correlazione e caricamento:** il sistema calcola in modo deterministico l'identificativo `community_id` (standard Zeek/Corelight [34]) e applica una compensazione degli offset di fuso orario, quindi carica i dati nelle due tabelle `cic_flows` e `ndpi_flows`.

I dettagli implementativi di questa fase (script Python, offset temporali) sono descritti in §6.1.

4.5.2 Fase 2: user request

L'utente non interagisce con il sistema tramite un'interfaccia a linguaggio naturale completamente libero, ma seleziona una delle cinque categorie di indagine predefinite dal sistema, ciascuna associata a un scenario investigativo specifico (cfr. §7.5). Il modello LLM analizza la richiesta partendo dall'ipotesi della categoria selezionata e seleziona dalla libreria del server MCP il tool più idoneo tra quelli disponibili (cfr. §6.2.1).

4.5.3 Fase 3: context enrichment (MCP Server)

Il server MCP, riceve l'invocazione del tool ed esegue le query SQL ottimizzate. In questa fase avviene l'Injection del Contesto:

- **Filtering:** invece di trasmettere l'intero database, l'MCP isola solo i risultati pertinenti e i relativi metadati di sintesi.
- **Enrichment:** il sistema arricchisce i dati grezzi eseguendo funzioni dedicate (es. `arricchisci_protocollo`) per rendere i dati interpretabili dall'LLM.
- **Gating operativo:** l'esecuzione dell'indagine è vincolata da un insieme di tool obbligatori (`TOOL_OBBLIGATORI`, cfr. §6.2.1) che devono essere invocati prima che il modello possa chiudere l'analisi; un meccanismo di controllo lato client blocca esplicitamente i tentativi di chiusura prematura.

4.5.4 Fase 4: reasoning & response

L'LLM riceve il pacchetto JSON arricchito dall'MCP. Grazie alle istruzioni contenute nel System Prompt, il modello opera una sintesi cognitiva correlando le diverse feature, per poi generare un report tecnico strutturato che si conclude con un verdetto tra cinque possibili categorie (cfr. §6.4): WEB_ATTACK_EXPLOIT, SCAN_BRUTEFORCE, DOS_VOLUMETRIC, BEACONING_C2, BENIGN.

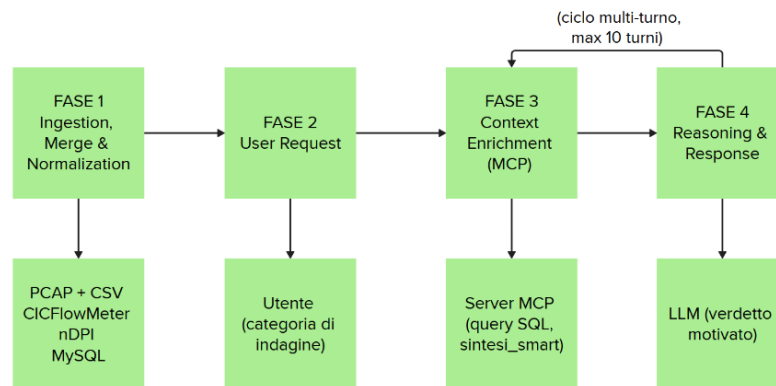


Figura 4.3: Pipeline operativa del sistema.

La pipeline trasforma il dato grezzo in un verdetto motivato attraverso quattro fasi: (1) Ingestion, Merge & Normalization; (2) User Request; (3) Context Enrichment; (4) Reasoning & Response. Il ciclo multi-turno tra Fase 3 e Fase 4 (fino a MAX_DRILLDOWN_TURNS = 10) permette all'LLM di raffinare l'indagine richiedendo nuovi tool sulla base delle evidenze via via raccolte.

5. Scelte progettuali

5.1 Introduzione

Nel processo di progettazione del sistema sono state prese diverse decisioni architetturali e tecnologiche, ciascuna valutata in base a criteri di efficienza, scalabilità e compatibilità con il contesto applicativo. Questo capitolo analizza in dettaglio le principali scelte effettuate, mettendole a confronto con possibili alternative e motivando le decisioni finali.

5.2 Scelta del database e ingegnerizzazione dello schema

Per la gestione e l'archiviazione dei dati di rete, la scelta è ricaduta su un database relazionale, nello specifico MySQL. Questa soluzione è risultata la più idonea per garantire la consistenza e le performance di indicizzazione necessarie all'analisi del traffico, vista la natura fortemente strutturata delle feature estratte.

Motivazioni

- Struttura tabellare ideale per gestire feature di traffico già parzialmente strutturate.
- Supporto nativo a SQL per interrogazioni articolate e ottimizzate.
- Ampia diffusione, affidabilità e documentazione esaustiva.

Architettura multi-tabella Lo schema è stato ingegnerizzato prevedendo due tabelle distinte: `cic_flows`, contenente i dati estratti tramite `CICFlowMeter` e arricchiti con le label ufficiali del dataset, e `ndpi_flows`, contenente l'identificazione dei protocolli a livello applicativo e le metriche di DPI generate da `ndpiReader`. Le due tabelle condividono la stessa 5-tupla di rete e sono correlate tramite il campo `community_id`, un identificatore derivato dalla 5-tupla stessa e utilizzato come chiave esterna per il join.

La separazione in due tabelle non nasce da un processo di normalizzazione formale, ma da un'esigenza pratica: preservare l'eterogeneità strutturale dei due estrattori ed evitare

una denormalizzazione forzata che, in una singola tabella unificata, avrebbe saturato di valori NULL tutti i campi presenti esclusivamente nell'output di nDPI.

ndpiReader è l'utility a riga di comando del framework nDPI. Processa file PCAP e produce un output strutturato (JSON Lines) contenente, per ogni flusso, il protocollo applicativo identificato, l'hostname (SNI), la versione TLS, e altri metadati di livello 7. Per garantire la massima efficienza e l'interoperabilità, i tipi di dato sono stati ottimizzati a livello nativo:

- **Precisione Temporale:** è stato adottato il tipo `TIMESTAMP (6)` per supportare la granularità al microsecondo, elemento critico per discriminare la sequenzialità dei pacchetti di rete.
- **Flessibilità di Rete:** gli indirizzi IP sono stati mappati come `VARCHAR (45)` per garantire il supporto nativo sia allo standard IPv4 sia allo standard IPv6.

5.2.1 Strategia di identificazione e correlazione dei flussi

Un problema centrale nell'integrazione di strumenti eterogenei risiede nella difficoltà di correlare i flussi registrati dai due estrattori. Il disallineamento tra le due fonti ha due cause distinte.

La prima è un **offset di fuso orario fisso**: CICFlowMeter registra i timestamp nel fuso locale della macchina di cattura, mentre nDPI utilizza UTC. Questo produce uno scarto sistematico (5 ore per la pipeline CICFlowMeter/Ufficiali e 3 ore per la pipeline nDPI), che viene corretto in fase di ingestion.

La seconda, è il **diverso criterio di partizionamento dei flussi**: CICFlowMeter chiude un flusso quando la sua durata totale supera una soglia di timeout (120 secondi), indipendentemente dalla presenza di traffico attivo, frammentando così una singola sessione di comunicazione in molteplici record; nDPI, al contrario, tende ad aggregare l'intera sessione in un unico flusso, anche se questa si estende per un periodo prolungato. Questo secondo problema non è correggibile con un offset e rende impossibile una corrispondenza 1:1 tra le due tabelle.

Se si fosse incluso il timestamp assoluto all'interno del calcolo di una chiave primaria statica, la minima variazione temporale o di fuso tra i due estrattori avrebbe generato chiavi non coincidenti per lo stesso identico evento di rete, rendendo impossibile un'operazione di JOIN relazionale diretta.

Per superare questo limite, l'architettura adotta l'indice logico **Community ID Flow Hashing**, uno standard di identificazione dei flussi sviluppato nell'ambito del progetto Zeek (ex Bro) [34]. Il *Community ID* viene calcolato applicando l'algoritmo di hashing SHA-1 alla codifica bidirezionale della 5-tupla (*IP sorgente, IP destinazione, porta sorgente, porta destinazione, protocollo di trasporto*), seguita da codifica Base64. La sua proprietà fondamentale è la bidirezionalità: lo stesso flusso osservato da due sonde diverse produce lo stesso identificativo, anche se sorgente e destinazione sono invertite.

Meccanismo di accoppiamento e gestione delle cardinalità. A livello implementativo, la correlazione tra le estrazioni proprietarie e i dati ufficiali non avviene tramite una de-duplicazione statica drastica, bensì attraverso un allineamento temporale relativo basato sul Community ID. I dettagli implementativi (calcolo del Community ID, merge_asof con tolleranza a 60 secondi, popolamento della tabella `flow_mapping`) sono riportati in Appendice A.

Vantaggi architetturali Questa strategia offre due vantaggi determinanti per il sistema:

- **Risoluzione della Discrepanza Temporale:** Spostando la correlazione dalle coordinate cronologiche rigide a quelle topologiche (la 5-tupla del Community ID) combinate con un'euristica a tolleranza temporale, si ottiene il recupero affidabile delle etichette di attacco/benigno senza perdere la struttura dei flussi registrati.
- **Integrità dei Dati e Performance:** L'impiego del `Community_ID` come colonna indicizzata nel database MySQL ottimizza notevolmente l'esecuzione delle query di JOIN e GROUP BY, aumentando la reattività dei tool a disposizione dell'LLM.

Alternative considerate e sviluppi futuri Durante la fase di progettazione sono stati valutati i database NoSQL, ma sono stati scartati a causa della minore efficienza nell'esecuzione di query di JOIN complesse, operazione che in questo lavoro costituisce il pilastro

fondamentale per correlare i flussi di CIC e nDPI. È stata inoltre inizialmente valutata (ma poi scartata) una chiave basata su hashing MD5 comprensiva del timestamp; per i motivi esposti sopra, tale approccio si è rivelato incompatibile con la correlazione tra estrattori a causa dell'offset temporale.

5.3 Scelta del modello e infrastruttura di inferenza

Il progetto è stato sviluppato secondo un approccio *model-agnostic*: il client (`client.py`) è in grado di dialogare con qualunque endpoint compatibile con l'API OpenAI, selezionando a runtime provider e modello tramite `utils.seleziona_modello_engine()`. In pratica, sono state configurate due combinazioni provider/modello:

- GPT-OSS 120B, erogato tramite Groq;
- Qwen 3.8 27B, erogato tramite un server dedicato ("Interhost") offerto dal relatore.

5.3.1 Motivazioni della scelta del modello

La prima configurazione valutata su Groq era basata su LLaMA 3.1 (8B/70B) [35], successivamente rimosso dalla piattaforma. Il passaggio a GPT-OSS 120B, pur architetturealmente compatibile, si è rivelato impraticabile a causa dei limiti di token imposti dal tier gratuito: un singolo scenario, caratterizzato da molteplici turni di tool-calling, esaurisce il budget disponibile prima che il modello possa produrre un verdetto completo. Tale configurazione resta selezionabile tramite `utils.seleziona_modello_engine()`, ma non è stata impiegata nella validazione sperimentale.

Il lavoro si è quindi concentrato sulla configurazione Qwen 3.8 27B via Interhost, che non impone limiti di token tali da ostacolare i cicli di test, e sulla quale sono stati progressivamente raffinati il system prompt e le regole di disambiguazione dei verdetti (cfr. §6.4). Sebbene il design resti in linea di principio compatibile con altri modelli (incluse famiglie diverse, come LLaMA, GPT o Claude, tramite semplice riconfigurazione

dell'endpoint), i risultati sperimentali presentati in questa tesi si riferiscono all'esecuzione su Qwen 3.8 27B.

5.3.2 Infrastruttura di inferenza e provider

Inizialmente è stata valutata l'esecuzione locale del modello; tuttavia, per massimizzare la reattività del sistema, si è optato per un'infrastruttura basata su API di inferenza esterne.

Motivazioni. L'opzione Groq garantisce, in linea teorica, una latenza estremamente ridotta grazie alle LPU (Language Processing Unit) dedicate. Delegando il calcolo a un'infrastruttura esterna, inoltre, si eliminano i colli di bottiglia legati alla memoria video (VRAM) e alla potenza di calcolo del sistema locale, permettendo alla pipeline di analisi dei dati di operare anche su dispositivi con risorse limitate. Come contropartita, l'approccio via API introduce dipendenza dalla connettività di rete e da servizi esterni, oltre ai limiti sui token imposti dai singoli provider.

Modularità per la sicurezza. Sebbene in questa fase di prototipazione si utilizzino endpoint esterni per validare l'architettura, la pipeline è nativamente predisposta per la transizione verso server di inferenza locali, senza dover modificare la logica dell'integrazione MCP o la struttura della pipeline di monitoraggio. Il limite operativo principale rimane il tier gratuito di Groq, i cui limiti di token rendono GPT-OSS 120B inutilizzabile in pratica.

Decisione. L'integrazione tramite provider compatibili OpenAI è stata preferita per garantire flessibilità di configurazione; nella pratica, il lavoro di affinamento dei prompt e di validazione sperimentale si è concentrato su Interhost/Qwen 3.8 27B, unica configurazione utilizzabile per completare l'analisi degli scenari.

5.3.3 Alternative considerate

- **Esecuzione Locale:** inizialmente valutata per la massima privacy, è stata messa in secondo piano perché l'hardware a disposizione, privo di risorse GPU adeguate, avrebbe imposto modelli di dimensioni ridotte o quantizzati, con latenze più elevate

e un ragionamento meno affidabile rispetto a modelli di grandi dimensioni erogati via API.

- **Small Language Models (SLM, < 7B parametri):** pur essendo estremamente veloci, hanno mostrato limitazioni nella generazione/interpretazione di dati strutturati complessi e nel rispetto rigoroso delle istruzioni di formato richieste dal protocollo a due stadi (cfr. §6.4), portando a preferire modelli di dimensioni maggiori.

Modularità e Flessibilità. Grazie all'astrazione offerta dal protocollo MCP, la pipeline di analisi resta indipendente dal modello specifico utilizzato, permettendo teoricamente lo swap tra diversi modelli con una modifica minima alla configurazione del client asincrono.

5.4 Scelta del dataset

Per la validazione del sistema è stato utilizzato il dataset pubblico CIC-IDS-2017 fornito dal Canadian Institute for Cybersecurity (CIC) selezionato per la sua capacità di rappresentare vari scenari di attacco [7].

Motivazioni

- **Standard accademico:** i dataset del CIC rappresentano lo standard de facto nella letteratura scientifica dedicata alla Network Intrusion Detection.
- **Completezza del Traffico:** il dataset offre acquisizioni di traffico reale (in formato PCAP) che includono sia flussi benigni che scenari di attacco diversificati (DDoS, Brute Force, XSS, Infiltrazioni di rete).
- **Dati Etichettati e Validati:** la presenza di etichette predefinite consente di verificare oggettivamente la precisione delle inferenze dell'LLM rispetto al ground truth. Poiché tali dataset sono stati nativamente generati e validati dagli autori tramite CIC-FlowMeter, l'integrazione di questo specifico tool nella pipeline (cfr. §6.1) non è solo una scelta tecnica, ma una necessità metodologica per garantire la corrispondenza tra le feature estratte e le etichette originali.

Limiti e Considerazioni

- **Pattern Artificiali:** i dataset generati in ambienti controllati possono presentare pattern di traffico parzialmente sintetici.
- **Distanza Temporale:** sebbene il traffico catturato risalga a qualche anno fa, le feature estratte rimangono fondamentali per la validazione della pipeline.

Decisione. L'uso del dataset del CIC è stato preferito alla cattura di traffico proprietario per garantire la massima riproducibilità dell'esperimento e una base di test riconosciuta dalla comunità scientifica.

5.5 Scelta dell'architettura MCP

L'integrazione del MCP rappresenta una scelta architetturale centrale, fondamentale per superare l'isolamento intrinseco dei modelli standalone. La scelta di MCP come layer di mediazione è supportata dalla letteratura recente: [36] mostra come MCP formalizzi le interazioni agent-tool tramite descrittori di capacità e validazione di schema, migliorando l'affidabilità e riducendo il tempo di recovery in caso di mismatch.

Motivazioni

- **Disaccoppiamento tra Logica e Dati:** MCP permette una netta separazione tra il “motore di ragionamento” (LLM) e la “base di conoscenza” (Database MySQL).
- **Efficienza rispetto al Fine-tuning:** a differenza del ri-addestramento, MCP mantiene il sistema flessibile.
- **Superamento del Knowledge Cutoff:** grazie all'accesso diretto allo stato operativo della rete, MCP risolve il problema delle informazioni obsolete.
- **Grounding e Riduzione delle Allucinazioni:** il protocollo agisce come sistema di ancoraggio, vincolando l'output del modello alla configurazione effettiva dei dati.
- **Governance del Processo Investigativo:** oltre a fornire dati, l'MCP — in combinazione con la logica lato client — impone un gating esplicito sulla completezza

dei tool che devono essere eseguiti prima che l'indagine possa dirsi conclusa (cfr. §6.2.1), impedendo chiusure premature o basate su evidenze parziali.

Alternative considerate

- **Prompt Engineering Statico:** l'inserimento manuale dei dati nel prompt è stato scartato perché fornirebbe uno snapshot statico non scalabile.
- **Sistemi RAG (Retrieval-Augmented Generation) Standard:** sebbene efficace per dati non strutturati, per questo lavoro si è preferita l'evoluzione verso l'architettura MCP, che permette una comunicazione più strutturata e bidirezionale tra il modello e il database MySQL di rete [6].

Decisione. L'adozione di MCP è stata ritenuta la soluzione più adeguata per garantire scalabilità e precisione, permettendo al sistema di evolvere da un semplice assistente testuale a un vero strumento di supporto decisionale per la sicurezza delle reti.

5.6 Riepilogo delle scelte

Componente	Scelta	Motivazione principale	Alternative
Database	MySQL	Query efficienti, tabelle separate	NoSQL
Modello LLM	Qwen 3.8 27B	Prompt raffinati su questo modello, nessun limite di token	LLaMA 3.1, GPT-OSS 120B
Infrastruttura	API Cloud (Interhost)	Nessuna VRAM locale, intercambiabilità	Esecuzione locale
Architettura	Model Context Protocol	Integrazione dinamica, gating	Prompt Engineering, RAG
Dataset	CIC-IDS-2017	Validazione oggettiva, standard	Cattura traffico proprietario

Tabella 5.1: Riepilogo delle scelte progettuali.

Le scelte progettuali effettuate definiscono un'architettura modulare e (in linea di principio) model-agnostic, dove l'intelligenza artificiale non è un elemento isolato ma un'interfaccia operativa integrata, sebbene nella pratica sperimentale il sistema sia stato sviluppato, raffinato e validato prevalentemente su un singolo modello a causa dei limiti di token che rendono inutilizzabile l'alternativa Groq.

6. Implementazione

6.1 Pipeline di elaborazione

La pipeline implementativa trasforma il dato grezzo in un'informazione semantica secondo il paradigma Extract-Transform-Load (ETL). L'implementazione si articola in quattro script Python eseguiti in sequenza, più il binario esterno `ndpiReader`:

1. **Preprocessing.py** — accorpa e bonifica i CSV nativi frammentati del dataset (uno per ciascuno dei cinque giorni lavorativi), rimuovendo header duplicati e gestendo i valori Infinity/NaN nelle colonne numeriche critiche. Produce i file `*-Pulito-Definitivo.csv` per ciascun giorno.
2. **Join_CSV.py** — correla i flussi estratti localmente via `CICFlowMeter` con i CSV ufficiali aggregati, iniettando la colonna `Label`. La correlazione avviene tramite `community_id` bidirezionale e `merge_asof` con tolleranza ± 60 s (dettagli in Appendice A).
3. **ndpiReader (binario esterno)** — eseguito in parallelo sui medesimi PCAP, produce l'output JSON Lines con le feature semantiche di livello 7.
4. **Popola_DB_CSV_Ufficiali.py** — bulk load dei flussi `CICFlowMeter` arricchiti con le label nella tabella `cic_flows`.
5. **Popola_DB_json_ndpiReader.py** — parsing e bulk load dei JSON nDPI nella tabella `ndpi_flows`.

Correzione degli offset temporali. I due estrattori registrano i timestamp con offset sistematici diversi rispetto all'orario nominale di cattura, dovuti alla differente gestione nativa del fuso orario. In fase di caricamento, tali offset vengono corretti esplicitamente: i timestamp provenienti dalla pipeline `CICFlowMeter/Ufficiali` vengono retrocessi di 5 ore (`Popola_DB_CSV_Ufficiali.py`), mentre quelli provenienti da `ndpiReader`

vengono retrocessi di 3 ore (`Popola_DB_json_ndpiReader.py`), in modo da riportare entrambe le sorgenti alla finestra oraria lavorativa reale (09:00–17:00). La correlazione logica tra i record delle due tabelle non dipende comunque dal timestamp corretto, ma è affidata alla chiave `community_id`, calcolata sulle sole 5-tuple di rete e quindi insensibile a qualunque disallineamento temporale residuo.

Il ciclo chiuso investigativo. Il completamento della pipeline non si esaurisce con la persistenza dei dati: si estende alla fase di consumo e validazione. Il client asincrono (`client.py/test_suite.py`) avvia un ciclo di investigazione sul database tramite MCP, agendo esclusivamente su `ndpi_flows` (fase di blind test). L'agente esegue fino a `MAX_DRILLDOWN_TURNS = 10` turni di interrogazione, regolati da gating sui tool obbligatori e da una cache anti-duplicazione (cfr. §6.2.2 e §6.4). Al termine di ogni scenario in `test_suite.py`, la funzione `controlla_ground_truth` interroga `cic_flows` per recuperare la distribuzione delle label reali; `calcola_esito_classificazione` confronta il verdetto LLM con la ground truth e produce TP/TN/FP/FN. La pipeline si configura quindi come sistema a ciclo chiuso, capace di auto-valutare le proprie capacità di rilevamento in modo automatizzato (metodologia completa in §7).

La Figura 6.1 mostra graficamente il bivio di estrazione parallela tra `CICFlowMeter` e `ndpiReader` e il successivo caricamento nel database MySQL.

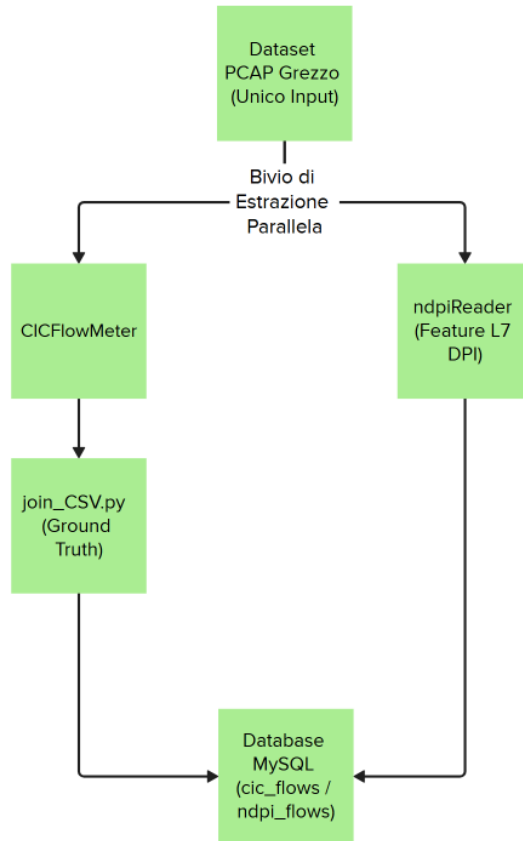


Figura 6.1: Pipeline di elaborazione.

6.2 Architettura MCP

L'integrazione tra il database e l'LLM non avviene tramite l'inserimento statico e massivo dei dati nel prompt, ma attraverso un'integrazione dinamica basata sul protocollo open-source Model Context Protocol (MCP) tramite il client. Questo approccio permette di superare i limiti della context window dei modelli attuali e garantisce che l'inferenza avvenga su dati sempre aggiornati [16].

6.2.1 Tassonomia dei tool MCP e scenari di investigazione

Il server MCP (server.py) espone quindici tool, organizzati in quattro macro-categorie funzionali. Tutti i tool operano esclusivamente sulla tabella `ndpi_flows`. Essendo que-

st'ultima nativamente priva del campo label (confinato nella tabella speculare cic_flows), l'architettura garantisce in modo strutturale l'integrità del blind test.

L'operatore non pone domande libere: seleziona una categoria di indagine, che fissa l'ipotesi di lavoro iniziale. La categoria non restringe i tool disponibili: tutti e 15 restano invocabili in ogni categoria, e alcuni (es. `get_rate_statistics`, `search_http_l7_anomalies`) servono a più categorie.

Analisi strutturale e sintesi

- **`get_traffic_summary(ip_target, start_time, end_time, limit)`**: panoramica cronologica dei flussi scambiati dall'host target, con diagnosi preliminare automatica (NORMALE/ANOMALO) e metadato `app_hierarchy` per identificare tunneling di protocolli su porte standard.
- **`get_aggregated_traffic_summary(ip_target, start_time, end_time)`**: raggruppa il traffico per (`dst_ip`, `dst_port`, protocollo, app), bypassando i limiti di campionamento dei singoli flussi — utile per capire quali porte/servizi concentrano il traffico.
- **`detect_beaconing(ip_target, start_time, end_time, dst_ip, top_n)`**: identifica comunicazioni periodiche tramite funzioni di partizionamento SQL, calcolando il Coefficiente di Variazione (CV) degli intervalli temporali tra connessioni successive.

Monitoraggio volumetrico (Anomalia Rate / DoS)

- **`get_rate_statistics(start_time, end_time, ip_target)`**: calcola metriche aggregate sulla finestra temporale — PPS medio e PPS di picco (su sotto-intervalli brevi), flussi e richieste Web al secondo, rapporto tra porte sorgente distinte e flussi, numero di flussi con pattern compatibile con Slowloris — e fornisce una diagnosi preliminare (NORMALE/ANOMALIA_RILEVATA).
- **`query_by_rate(threshold, start_time, end_time, metric, ip_address)`**: isola i flussi che violano soglie matematiche esplicite su `packet_rate` o `byte_rate`.

Investigazione endpoint (Top Talker & Host Profiling)

- **get_top_talkers(start_time, end_time, top_n, criterion):** vista aggregata preliminare degli host con maggior volume di traffico nella rete.
- **analyze_dpi_details(ip, start_time, end_time):** dettagli analitici DPI di livello 7 (ndpi_hostname, flag di entropia del payload, infra_provider, tls_version).
- **resolve_host_info(ip, start_time, end_time):** risolve informazioni contestuali (hostname, infra_provider) legate alla natura dell'IP.
- **analizza_connessione_by_community_id(cid):** recupera la singola sessione identificata dal community_id, consentendo il massimo livello di drill-down investigativo.
- **get_host_port_distribution(ip_target, start_time, end_time):** distribuzione delle porte di destinazione contattate, per rilevare Port Scan o tentativi di Brute Force mirati su porte specifiche.

Analisi comportamentale

- **search_connection_attempts(start_time, end_time, ip_target, target_port):** identifica tentativi multipli di connessione ravvicinati verso servizi sensibili.
- **search_http_l7_anomalies(ip_target, start_time, end_time, limit):** ispeziona i flussi Web per rilevare concentrazioni anomale o tentativi di Web Brute Force.
- **inspect_http_requests(ip_target, start_time, end_time, limit, offset):** estrae un campione grezzo di telemetria L7/HTTP, da usare come conferma di dettaglio successiva a search_http_l7_anomalies.
- **get_flow_features(ip, start_time, end_time):** parametri statistici temporali avanzati (Inter-Arrival Time) per applicare Temporal Reasoning e discriminare attività automatizzate.

Il motore euristico compute_verdict_scores Oltre ai tool di raccolta dati, il server espone **compute_verdict_scores(ip_target, start_time, end_time)**. Questo tool rappresenta il cuore della logica decisionale deterministica del sistema: internamente, invoca in

sequenza i tool di telemetria di base (`get_rate_statistics`, `get_host_port_distribution`, `detect_beaconing`, `search_http_l7_anomalies`, `search_connection_attempts`) e applica alle metriche raccolte un insieme di regole a soglia statica (definite in `config.Soglie`).

Grazie al meccanismo di cache di deduplicazione (§6.4.2), queste chiamate interne non comportano una nuova esecuzione delle query SQL se i tool di base sono già stati invocati nella stessa finestra di analisi con gli stessi parametri.

Quando più categorie raggiungono lo stesso punteggio massimo, il tool applica un tie-breaking interno basato sul confronto tra la forza relativa delle evidenze. Se nessun criterio dirime il pareggio, il tool restituisce esplicitamente `CONFLITTO_IRRISOLTO` e delega la decisione all'LLM, che deve motivarla sulla base delle evidenze grezze. Questo meccanismo è il punto in cui il contributo del modello è più visibile: nei casi di conflitto, l'euristica non ha una risposta, e il verdetto dipende interamente dal ragionamento dell'agente.

È importante sottolineare — ed è un vincolo ribadito ripetutamente nel system prompt (cfr. §6.4) — che questo punteggio **non costituisce un oracolo**: è un'euristica a soglia fissa che può produrre falsi positivi (traffico legittimo che sfiora casualmente una soglia) o falsi negativi (attacchi che rimangono sotto la soglia), e va sempre incrociato con almeno un'evidenza grezza indipendente prima di essere confermato nel report finale.

La calibrazione è stata condotta sull'insieme di sviluppo (193 scenari, cfr. §7.3) per ciascuna categoria, i valori delle metriche aggregate sono stati ispezionati sugli scenari noti, e le soglie sono state fissate per separare i pattern osservati con il margine più ampio possibile. I guardrail implementati sono simmetrici: `whitelist` e `deroga documentata` contro i falsi positivi, `DIVIETO_BENIGN_CON_ANOMALIE` e `blocco anti-falso-negativo` contro i falsi negativi.

Tool obbligatori e gating dell'indagine L'architettura impone un insieme di sei tool obbligatori (`TOOL_OBBLIGATORI`, definito in `config.py`): `get_rate_statistics`, `get_host_port_distribution`, `search_connection_attempts`, `search_http_l7_anomalies`, `detect_beaconing`, `compute_verdict_scores`.

Prima che l'LLM possa emettere il verdetto finale, il client (`client.py`) verifica esplicitamente che tutti i tool obbligatori siano stati eseguiti almeno una volta nel corso dell'indagine. Quando un tool rileva un segnale forte ($\text{score} \geq 0.95$), il client sollecita l'esecuzione

immediata di `compute_verdict_scores` ed evita ulteriori ispezioni di dettaglio con tool non obbligatori, senza però esonerare dai tool base. Se mancano ancora tools obbligatori, il client inietta un messaggio di sistema che blocca la chiusura e richiede esplicitamente l'esecuzione dei tool mancanti ([AVVISO_ANTI-BYPASS]). Un secondo meccanismo di controllo, l'anti-falso-negativo, impedisce inoltre di chiudere l'indagine con un verdetto BENIGN qualora le evidenze già raccolte mostrino anomalie non spiegate, forzando l'LLM a rivalutare esplicitamente i dati prima di confermare l'assenza di minacce.

6.2.2 Il middleware MCP: architettura e implementazione

L'integrazione tra l'LLM e il database di rete non avviene tramite una connessione diretta, ma attraverso il Model Context Protocol (MCP). Questo layer funge da intermediario intelligente, trasformando l'LLM da un generatore di codice potenzialmente fallibile a un orchestratore di funzioni specializzate, fondandosi su tre pilastri ingegneristici:

- **Filtraggio:** l'MCP non espone l'intero database, ma offre esclusivamente funzioni che estraggono feature mirate.
- **Abbattimento della Complessità Computazionale:** calcoli complessi (deviazione standard, partizionamento temporale con `LAG()`) sono incapsulati nelle query dei tool MCP ed eseguiti dall'engine MySQL, non dall'LLM.
- **Sicurezza e Validità Sintattica:** poiché l'LLM non scrive codice SQL autonomamente, si riduce il rischio di errori di sintassi o di vulnerabilità come la SQL Injection. Il modello richiama, tramite il client, esclusivamente le funzioni predefinite esposte dal server con parametri tipizzati tramite schemi JSON definiti nel Python SDK dell'MCP.



Figura 6.2: Flusso di comunicazione tra i componenti del sistema.

Flusso di comunicazione

L'agente LLM invia una richiesta al client asincrono, che la inoltra al server MCP. Il server MCP esegue la query SQL sul database MySQL e struttura il recordset grezzo in una risposta JSON, integrandolo con la sezione `sintesi_smart`, e lo restituisce al client sotto forma di risposta tool MCP. Il client media la comunicazione, applicando i controlli di gating (tool obbligatori, anti-falso-negativo) e restituisce la risposta all'agente LLM. Il client è il punto in cui vengono applicate le politiche di controllo dell'indagine; il server MCP è il punto in cui viene garantito l'isolamento del blind test (accesso esclusivo a `ndpi_flows`).

Implementazione del server MCP

Il server MCP (`server.py`) è implementato con **FastMCP** [37], una libreria Python che semplifica la definizione dei tool MCP. Il decoratore `@mcp.tool()` genera automaticamente gli schemi JSON di input a partire dai type hints (le annotazioni di tipo di Python che dichiarano i tipi attesi di parametri e valori di ritorno) delle funzioni Python, eliminando la necessità di scrivere manualmente i descrittori di capacità. Questo riduce il rischio di errori di disallineamento tra la firma della funzione e lo schema esposto al client, e rende il codice più leggibile e manutenibile. Le descrizioni semantiche dei 15 tool sono definite centralmente in `prompts.py` e seguono la struttura *COSA FA / METRICHE E INTERPRETAZIONE / QUANDO USARLO / QUANDO NON USARLO / DIFFERENZA DAI TOOL SIMILI*.

Meccanismi di controllo e isolamento

Isolamento del blind test. L'isolamento non è affidato a filtri software a runtime, ma alla separazione strutturale delle tabelle: tutti i tool espongono query confinate a `ndpi_flows`, che è nativamente priva della colonna `label` (confinata in `cic_flows`). Questo azzerà i rischi che le label ufficiali del dataset filtrino accidentalmente verso l'agente LLM a causa di errori di programmazione del middleware, e rende il blind test verificabile per semplice ispezione del codice.

Gating e cache. Il client (`client.py`) applica tre controlli prima di accettare la chiusura dell'indagine: (i) verifica che i sei tool obbligatori (`TOOL_OBBLIGATORI`) siano stati eseguiti; (ii) blocco anti-falso-negativo, che impedisce di chiudere con `BENIGN` in presenza di anomalie strutturate; (iii) anti-duplicazione, implementato su due livelli — una cache lato server (`mcp_cache_guard`, TTL 300 s) e uno storico di hash SHA-256 lato client. Per i tool paginabili, il client avanza automaticamente l'offset in caso di richiesta duplicata.

Pruning del contesto. Durante indagini prolungate il client applica un pruning progressivo (`applica_pruning_contesto`, `comprimi_messaggi_contesto`) che mantiene intatto il system prompt e il messaggio corrente con le evidenze fresche, mentre comprime o rimuove la cronologia più vecchia. Questo permette fino a 10 turni di drill-down senza saturare la context window.

Gestione degli errori

Ogni tool è progettato per restituire una risposta strutturata anche in caso di errore. Se una query SQL fallisce o restituisce un set vuoto, il tool non solleva un'eccezione che interrompe l'indagine, ma restituisce un messaggio di stato (es. `{"status": "error", "message": "..."}`) che l'LLM può interpretare. Questo approccio evita che un singolo fallimento blocchi l'intero ciclo investigativo e permette al modello di decidere se ritentare con parametri diversi o procedere con le evidenze già raccolte.

6.3 Preprocessing e feature engineering

Il preprocessing dei dati è la fase critica che garantisce l'integrità dell'analisi condotta dall'LLM. La sfida principale risiede nella natura eterogenea degli output forniti da nDPI e CICFlowMeter. Il sistema uniforma e correla queste sorgenti preservandone le caratteristiche native all'interno di un'architettura multi-tabella in MySQL, adottando le seguenti tecniche:

- **Normalizzazione temporale e formattazione:** gli estrattori gestiscono il tempo in modo differente (CICFlowMeter usa stringhe leggibili in formato locale, nDPI uti-

lizza Unix Epoch Time con microsecondi). Lo script di ingestion converte entrambi i formati nello standard TIMESTAMP(6), applicando le correzioni di offset descritte in §6.1 (-5h per CIC, -3h per nDPI) e utilizzando il community_id come chiave logica di associazione, insensibile a tali correzioni.

- **Scomposizione e Mappatura Relazionale:** i dati sono stati ripartiti nelle due tabelle distinte cic_flows e ndpi_flows, mantenendo come punto di contatto l'indice community_id.
- **Tipizzazione numerica e codifica:** le feature quantitative di volume (byte e pacchetti) vengono standardizzate su formati a precisione intera (BIGINT UNSIGNED) per evitare overflow, mentre i tassi in DOUBLE; i flag TCP sono codificati in una bitmap intera.
- **Aggregazione Comportamentale:** le feature comportamentali (PPS aggregati, CV degli intervalli, distribuzione delle porte) non sono precalcolate nel DB, ma calcolate a richiesta dai tool MCP sulla finestra dello scenario.

Nome Colonna	Tipo	Tabella Ospitante	Descrizione
id	BIGINT UNSIGNED [PK]	cic_flows / ndpi_flows	id univoco usato come chiave primaria
community_id	VARCHAR(50)	cic_flows / ndpi_flows	Community ID (SHA-1) della 5-tupla bidirezionale (chiave di JOIN)
src_ip / dst_ip	VARCHAR(45)	cic_flows / ndpi_flows	Indirizzi IP sorgente e destinazione
src_port / dst_port	INT UNSIGNED	cic_flows / ndpi_flows	Porte di trasporto sorgente e destinazione
protocol	INT UNSIGNED	cic_flows / ndpi_flows	Protocollo di trasporto (es. 6=TCP, 17=UDP)
timestamp_start	TIMESTAMP(6)	cic_flows / ndpi_flows	Inizio del flusso, corretto per l'offset dell'estrattore
duration_ms	DOUBLE	cic_flows / ndpi_flows	Durata della sessione in millisecondi
total_bytes	BIGINT UNSIGNED	cic_flows / ndpi_flows	Volume totale del traffico scambiato
fwd_packets / bwd_packets	BIGINT UNSIGNED	cic_flows / ndpi_flows	Conteggio pacchetti inviati e ricevuti
total_fwd_bytes / total_bwd_bytes	BIGINT UNSIGNED	cic_flows / ndpi_flows	Volume parziale upload (Fwd) e download (Bwd)
packet_rate	DOUBLE	cic_flows / ndpi_flows	Frequenza dei pacchetti (pacchetti/secondo)
byte_rate	DOUBLE	cic_flows / ndpi_flows	Throughput del flusso (byte/secondo)
iat_flow_avg / iat_flow_stddev	DOUBLE	cic_flows / ndpi_flows	Media e deviazione standard dell'Inter-Arrival Time
label	VARCHAR(50)	cic_flows	Etichetta ufficiale del dataset (ground truth)
tcp_flags	INT	ndpi_flows	Bitmap cumulativa dei flag TCP della sessione
ndpi_hostname	VARCHAR(255)	ndpi_flows	SNI estratto dal certificato TLS o Host Header HTTP
payload_entropy	DOUBLE	ndpi_flows	Flag binario {0,1}: 1 se nDPI segnala entropia del payload anomala ("Susp Entropy"), 0 altrimenti
app_hierarchy	VARCHAR(100)	ndpi_flows	Classificazione del protocollo applicativo L7 (es. TLS.OpenVPN)
infra_provider	VARCHAR(100)	ndpi_flows	Fornitore infrastrutturale dell'IP (es. AWS, Azure)
tls_version	VARCHAR(10)	ndpi_flows	Versione del protocollo di cifratura identificata
tls_cipher_suite	VARCHAR(100)	ndpi_flows	Suite crittografica estratta durante l'handshake
tls_ja4	VARCHAR(36)	ndpi_flows	Impronta digitale JA4 del client TLS
tls_issuer_dn	VARCHAR(255)	ndpi_flows	Distinguished Name dell'ente emittitore del certificato

Tabella 6.1: Campi delle tabelle cic_flows e ndpi_flows.

Le due tabelle non hanno la stessa cardinalità: cic_flows contiene 2.827.901 record contro 1.349.098 di ndpi_flows (circa 2,1×). La differenza dipende dai diversi criteri con cui i due software delimitano un flusso (timeout, terminazione della sessione), che frammentano la stessa comunicazione in un numero diverso di record. Di conseguenza il community_id non stabilisce una corrispondenza uno a uno, e i conteggi di flussi della ground truth (in unità CIC) non sono direttamente confrontabili con quelli visti dall'LLM (in unità nDPI).

I flag TCP sono codificati in un unico intero a 8 bit tramite shift posizionale standard di rete ($\text{tcp_flags} = \sum f_i \cdot 2^i$ con ordine FIN, SYN, RST, PSH, ACK, URG, ECE, CWR). Questa scelta comprime otto feature booleane in una singola colonna INT, interrogabile in SQL tramite mascheramento bit a bit.

La combinazione di app_hierarchy, iat_flow_stddev e payload_entropy abilita il ragionamento contestuale dell'LLM: come discusso nel Capitolo 2, l'IAT è una delle feature più discriminanti per la classificazione del traffico, e pattern regolari (bassa deviazione

standard) sono tipici di comunicazioni automatiche come il beaconing C2.

payload_entropy è memorizzato come DOUBLE nel database, ma il suo valore è semanticamente un flag binario $\{0, 1\}$. La pipeline di ingestion converte esplicitamente il valore a intero prima del caricamento nel database, coerentemente con la semantica di flag. La scelta di mantenere il tipo DOUBLE nello schema, è deliberata: permette in sviluppi futuri di estendere il campo a valori di entropia reali (Shannon entropy) senza alterare lo schema, mantenendo retrocompatibilità con i dati già caricati.

6.3.1 Disallineamento temporale e aggregazione dei flussi

Un limite emerso durante la fase di validazione riguarda il disallineamento tra le due fonti di dati, cic_flows (ground truth) e ndpi_flows (sorgente per l'LLM). Nonostante le correzioni di offset temporale applicate in fase di ingestion (§6.1), i due estrattori, CICFlowMeter e nDPI, adottano criteri di terminazione del flusso intrinsecamente diversi.

- **CICFlowMeter** è più granulare: chiude un flusso quando la sua durata complessiva supera la soglia di timeout (120 secondi), indipendentemente dalla presenza di traffico attivo, frammentando così una singola sessione di comunicazione in molteplici record.
- **nDPI** è più persistente: tende ad aggregare l'intera sessione in un unico flusso, anche se questa si estende per un periodo prolungato.

Nota tecnica. Questo disallineamento è un limite intrinseco dell'uso di ndpiReader come strumento di estrazione. ndpiReader è un tool di test, non un probe professionale: non è progettato per gestire nativamente la correlazione temporale tra flussi estratti da fonti diverse. Un probe come nprobe avrebbe gestito la correlazione in tempo reale, risolvendo alla radice il problema. La scelta di ndpiReader è stata però dettata dalla sua semplicità di integrazione in una pipeline di analisi offline. nprobe, è un probe progettato per operare su traffico live e richiederebbe un'infrastruttura di cattura e collezione più complessa. Per una pipeline di test offline come quella descritta in questa tesi, ndpiReader rappresenta quindi la scelta più diretta, ma ha richiesto l'introduzione della tabella flow_mapping per compensare il disallineamento temporale tra le due sorgenti.

Esempio concreto (scenario VAL_CAT_A_021): Un'analisi diretta del database rivela che la tabella `cic_flows` contiene 7 record distinti etichettati `Heartbleed`, ciascuno con una durata di circa 2 minuti; la tabella `ndpi_flows`, interrogata sulla stessa finestra, non restituisce alcun flusso. La causa è duplice: da un lato, `CICFlowMeter` frammenta la sessione in più record per effetto del timeout; dall'altro, `nDPI` aggrega la sessione in un unico flusso che inizia prima dell'inizio della finestra di analisi, e viene quindi escluso dal filtro temporale.

Conseguenze per la validazione: Questo disallineamento ha un impatto diretto sulla validazione e sull'operato dell'LLM:

1. **Falsi negativi sistemici:** Poiché i tool MCP interrogano esclusivamente `ndpi_flows`, l'agente LLM non trova alcun dato e conclude correttamente `BENIGN`, mentre la ground truth classifica la finestra come `WEB_ATTACK_EXPLOIT`: si genera così un falso negativo non imputabile al ragionamento del modello, ma a un limite della finestra di dati. Per questa ragione lo scenario è stato escluso dalla validazione sperimentale.

La tabella `flow_mapping`, introdotta per associare a ciascun `community_id` la label ufficiale del dataset, mitiga parzialmente il problema ma non lo risolve quando i timestamp di `nDPI` e del mapping non coincidono — come conferma la query di verifica (Code B.2), che per questo scenario restituisce 0 righe.

2. **Impatto sulla ground truth aggregata:** Poiché i due estrattori non hanno una corrispondenza 1:1, il confronto tra il verdetto dell'LLM e la ground truth non può che avvenire a livello aggregato, ovvero per ogni coppia (host, finestra di 15 minuti). Questo approccio, sebbene necessario, introduce un'ulteriore semplificazione, come discusso in §7.4.2.

6.4 Prompt engineering e architettura agentica MCP

La formulazione del prompt e la strutturazione del contesto operativo rappresentano l'elemento determinante per trasformare un modello linguistico generalista in un analista

SOC autonomo. Il sistema adotta tecniche di In-Context Learning, Grounding e Chain of Thought (CoT) per vincolare le risposte dell’LLM ai dati reali estratti, riducendo il rischio di allucinazioni [17].

A differenza dei tradizionali approcci statici di Text-to-SQL, l’architettura implementata sfrutta le capacità dinamiche del Model Context Protocol: l’LLM non genera query grezze, ma invoca selettivamente una suite di tool analitici astratti, valutando i risultati in cicli multi-turno.

6.4.1 Architettura del system prompt e tassonomia del verdetto

Il sistema adotta una tassonomia a cinque classi, coerente con le principali famiglie di attacco rappresentate nel dataset CIC-IDS-2017: WEB_ATTACK_EXPLOIT, SCAN_BRUTEFORCE, DOS_VOLUMETRIC, BEACONING_C2, BENIGN.

Classe	Etichette CIC	Segnale distintivo	Tool principali
DOS_VOLUMETRIC	DDoS, DoS Hulk, GoldenEye Slowloris, SlowHTTPTest	volume/RPS elevato, molte porte effimere flussi lunghi a basso volume	get_rate_statistics get_flow_features, get_rate_statistics
SCAN_BRUTEFORCE	PortScan FTP-Patator, SSH-Patator	molte porte su un singolo target tentativi ripetuti su una porta di gestione	get_host_port_distribution, search_connection_attempts search_connection_attempts
WEB_ATTACK_EXPLOIT	Web Brute Force XSS, SQL injection, Heartbleed, Infiltration	≥ 25 richieste su ≤ 4 target payload invisibile al DB (solo metadata)	search_http_l7_anomalies
BEACONING_C2	Bot	periodicità (CV) verso IP esterno	detect_beaconing

Tabella 6.2: Tassonomia delle cinque classi di verdetto.

Categoria	Soglia principale	Valore	Note
DOS_VOLUMETRIC	DOS_PPS_MIN	2000 pps	Soglia aggregata su finestra
DOS_VOLUMETRIC	DOS_L7_RPS_MIN	7 rps	RPS effettivo, non solo conteggio flussi
DOS_VOLUMETRIC	DOS_PPS_MIN_FALLBACK	700 pps	Attivata solo con traffico concentrato
SCAN_BRUTEFORCE	SCAN_PORTE_MIN	12 porte	Porte distinte contattate
SCAN_BRUTEFORCE	BRUTEFORCE_TENTATIVI_MIN	20 tentativi	Tentativi su porta di gestione
WEB_ATTACK_EXPLOIT	WEBBF_MIN_RICHIESTE	25 richieste	Richieste su ≤ 4 target
BEACONING_C2	CV_BEACON_JITTER_MAX	0.6	CV degli intervalli tra connessioni successive
BEACONING_C2	BEACON_MIN_CONNESSIONI	12 connessioni	Minimo connessioni per la conferma del beaconing

Tabella 6.3: Soglie numeriche principali utilizzate dal motore euristico.

Il System Prompt (prompts.SYSTEM_PROMPT_CONTENT) configura l’LLM come un Analista SOC esperto in Threat Hunting, operante in modalità blind test sulla tabella ndpi_flows, e integra:

- una **direttiva sul verdetto** ribadisce la natura euristica (non oracolare) di `compute_verdict_scores`, imponendo di incrociare sempre lo score con evidenze grezze indipendenti prima di confermarlo;
- una **regola di attribuzione delle evidenze**, che associa a ciascuna categoria la propria “prova primaria” (es. la natura applicativa del traffico per `WEB_ATTACK_EXPLOIT`, il volume/rate per `DOS_VOLUMETRIC`) e una gerarchia di default da applicare solo in caso di ambiguità comparabile tra evidenze concorrenti;
- una serie di **regole tassative di disambiguazione**, che fissano soglie numeriche esplicite per distinguere, ad esempio, un Web Brute Force a basso volume da un vero DoS applicativo L7 (discriminante: RPS effettivo, non il solo conteggio di flussi), o uno scan L4 da una vittima di flood in ricezione (guardrail anti-falso-positivo);
- una **categoria di indagine** (`FOCUS_CATEGORIE_CONTENT`), che fornisce all’LLM un’ipotesi di lavoro iniziale e un ordine di verifica coerente con il motivo per cui l’host è stato selezionato per l’indagine, ricordando esplicitamente che tale ipotesi va confermata o smentita sulla base delle evidenze, non assunta a priori.

Processo di emissione del verdetto a due stadi. L’emissione del verdetto finale segue un protocollo a due stadi, pensato per bilanciare la libertà di ragionamento del modello con la necessità di un output strutturato e affidabile:

Stage 1 – Esplorazione. L’LLM sceglie autonomamente i tool (fino a `MAX_DRILL_DOWN_TURNS` turni, `tool_choice=auto`). Il ragionamento in linguaggio naturale (Chain of Thought) è incoraggiato dal prompt ma non imposto né verificato: il modello può commentare le chiamate e, al termine, formula un giudizio finale in testo libero.

Il client vincola l’indagine su quattro fronti:

1. **Tool obbligatori.** I sei tool obbligatori devono essere stati eseguiti prima di poter concludere. L’eccezione del segnale forte ($\text{score} \geq 0.95$) non esonera da tutti i tool: salta solo i tool di telemetria base, mantenendo comunque obbligatorio `compute_verdict_scores`. Senza segnale forte, tutti i `TOOL_OBBLIGATORI` sono richiesti.

2. **Ordine di esecuzione.** `compute_verdict_scores` è accettato solo dopo che tutti i tool di telemetria base (`TOOL_OBBLIGATORI - {compute_verdict_scores}`) sono stati eseguiti; altrimenti la chiamata viene rifiutata con status: `rejected`.
3. **Blocco anti-falso-negativo.** Articolato su tre livelli: (i) nel loop di esplorazione, quando l’LLM tenta di chiudere con `BENIGN`; (ii) alla chiusura dello Stage 1 (Guard FN in Fase 2); (iii) nel ramo max-turni, con richiesta all’LLM di motivare strutturalmente la deroga su cinque punti obbligatori. Impedisce di chiudere con `BENIGN` in presenza di anomalie strutturate, salvo deroga motivata con dati concreti.
4. **Anti-duplicazione e anti-loop.** `storico_chiamate_hash + gestisci_e_calcola_hash_tool` bloccano chiamate identiche; `_controlla_loop_community_id` intercetta cicli su Community ID.

Criteri di determinazione del verdetto. Quando l’esplorazione termina, il client confronta il giudizio dell’LLM con quello dell’euristica automatica e applica le regole riportate nella Tabella 6.4, in ordine di priorità decrescente.

Priorità	Condizione	Esito
1	LLM ha espresso un verdetto chiaro (sicurezza alta)	Prevale il verdetto LLM
2	LLM ha espresso un verdetto con sicurezza media	Verdetto LLM accettato in via provvisoria; la Fase 2 può correggerlo
3	LLM dice BENIGN, ma l'euristica rileva un attacco con score forte	Prevale l'euristica (blocco anti-falso-negativo)
4	LLM dice BENIGN, ma l'euristica rileva un attacco con score debole o ridimensionato	Prevale il BENIGN dell'LLM; il conflitto è registrato nei log
5	Euristica in pareggio tra più categorie e LLM senza preferenza	Si applica l'ordine di priorità predefinito (categorie con prove concrete)
6	LLM non ha espresso un verdetto utilizzabile	Si usa il suggerimento dell'euristica, in via provvisoria
7	Nessun verdetto disponibile	BENIGN, con segnalazione di eventuali anomalie

Tabella 6.4: Regole di priorità per la determinazione del verdetto finale.

Le condizioni 1 e 2 riguardano il caso in cui l'LLM ha già espresso un giudizio esplicito; le condizioni 3 e 4 riguardano il caso opposto (LLM dice BENIGN, l'euristica contesta); le condizioni 5–7 coprono i casi di fallback.

Stage 2 – Report finale. L'LLM produce un report in formato JSON con due campi: il verdetto e la motivazione. Il client controlla che sia coerente con quanto stabilito nella Fase 1:

- **Se la Fase 1 ha prodotto un verdetto chiaro**, l'LLM deve conservarne la sostanza: non può trasformare un attacco in BENIGN, né promuovere un BENIGN in attacco. È invece ammesso correggere la categoria specifica (da una categoria di attacco a un'altra), purché l'LLM porti prove nuove che la Fase 1 non aveva considerato, elencandole in una sezione dedicata del report.

- **Se la Fase 1 era solo un fallback automatico** (l’LLM non aveva dato un giudizio esplicito), l’LLM è libero di correggere, anche verso “BENIGN”, purché lo dichiari esplicitamente.
- **Se la risposta non è conforme** (JSON malformato, verdetto non valido, oppure correzione priva delle dichiarazioni richieste), il client ritenta automaticamente fino a MAX_RETRY_REPORT volte, spiegando all’LLM cosa ha sbagliato.

6.4.2 Regole di trasparenza investigativa e logica anti-loop

Il system prompt incoraggia una regola di trasparenza: prima di ogni invocazione di tool, il modello dovrebbe generare un commento testuale che espliciti le anomalie riscontrate e motivi la necessità del successivo drill-down. Va precisato che questo vincolo non è imposto come blocco hard a livello di codice, ma è promosso attraverso le istruzioni del prompt.

Per mitigare il fenomeno in cui l’LLM invoca ripetutamente lo stesso tool con gli stessi parametri, il sistema implementa un doppio livello di protezione:

- **Cache di deduplicazione lato server (mcp.cache_guard in server.py):** ogni chiamata a un tool viene identificata da una chiave calcolata su nome funzione e parametri normalizzati (inclusa una firma di paginazione per offset/limit). Se la stessa chiave è già presente in cache con un timestamp entro il TTL configurato (CACHE_TTL_SECONDS = 300), il server non riesegue la query SQL, ma restituisce il risultato precedente accompagnato da un avviso esplicito che invita il modello a cambiare parametri o tool. Questo meccanismo opera su tutto lo storico della sessione investigativa entro la finestra di TTL, intercettando quindi anche duplicati non consecutivi.
- **Storico degli hash lato client (storico_chiamate_hash in client.py):** il client calcola indipendentemente un hash SHA-256 deterministico sul nome del tool e sugli argomenti normalizzati (engine.gestisci_e_calcola_hash_tool) e lo confronta con lo storico delle chiamate già effettuate nella stessa indagine. Se l’hash è già presente, la chiamata viene bloccata e all’LLM viene restituito un messaggio di errore che

lo invita a cambiare parametri o tool. Per i tool paginabili (`inspect_http_requests`, `search_http_17_anomalies`), il client avanza invece automaticamente l'offset in caso di richiesta duplicata, così da non sprecare un turno investigativo su una query identica.

Un controllo aggiuntivo (`engine._controlla_loop_community_id`) rileva specificamente sequenze di almeno due chiamate consecutive a `analizza_connessione_by_community_id`, iniettando un avviso di sistema che invita il modello a procedere alla sintesi o al verdetto finale, per evitare un uso dispersivo del tool di drill-down puntuale. Il controllo è stato introdotto in seguito all'osservazione empirica che l'LLM, tende a reiterare il drill-down su `community_id` malformati o non corrispondenti, consumando turni senza produrre nuova evidenza.

Perché servono i guardrail anti-loop. L'esigenza di meccanismi anti-loop non nasce da un difetto specifico di Qwen 3.8 27B, ma è una caratteristica comune agli agenti LLM che operano in contesti di tool-calling multi-turno. Senza una memoria a lungo termine o una chiara percezione dello stato dell'indagine, il modello può tentare di rieseguire le stesse query nella speranza di ottenere risultati diversi o per semplice conferma.

Va precisato che questi guardrail sono stati introdotti durante la fase di sviluppo con LLaMA 3.1 (8B/70B), dove il problema era particolarmente evidente nella versione da 8B. Con Qwen 3.8 27B il fenomeno è molto ridotto, ma i guardrail sono stati mantenuti come difesa architetturale: garantiscono l'efficienza e la convergenza dell'indagine indipendentemente dalla potenza del modello, e rendono il sistema robusto anche in caso di sostituzione del motore di inferenza.

Categoria di indagine L'operatore seleziona una categoria (A-E) che definisce l'ipotesi di lavoro nel prompt iniziale. Non è un'istruzione di classificazione: il prompt richiede di confermare, correggere o smentire l'ipotesi con le evidenze, e tutti i tool restano disponibili. La categoria E è l'analisi generica, senza ipotesi.

7. Metodologia sperimentale e validazione dei risultati

7.1 Obiettivo della validazione

La fase sperimentale mira a quantificare le prestazioni del sistema in condizioni d'uso realistiche, verificandone la capacità di operare come motore di correlazione su dati di rete e di produrre verdetti da confrontare con la ground truth. La validazione persegue quattro obiettivi principali.

1. **Validazione dell'Intent Accuracy e del mapping semantico.** Capacità dell'LLM di tradurre l'intento investigativo in una sequenza coordinata di invocazioni MCP, senza eccezioni sintattiche o deviazioni anti-loop. Questo obiettivo è misurato attraverso l'analisi qualitativa dei log (cfr. §7.6.6) e il conteggio delle investigazioni completate rispetto a quelle abortite.
2. **Misurazione della robustezza analitica (Grounding).** Calcolo di Accuracy, Precision, Recall e F1-Score in blind testing, confrontando le inferenze con la ground truth ufficiale. La procedura di auditing è automatizzata tramite `test_suite.py` e `utils.py` (cfr. §6) e produce le metriche riportate in §7.6.1. Il confronto avviene a livello di finestra host/15 minuti, non di singolo flusso, per i motivi discussi in §7.4.2.
3. **Analisi dell'efficienza temporale.** Scomposizione della latenza end-to-end nelle sue componenti (`T_sql`, `T_mcp`, `T_llm`), per identificare il collo di bottiglia del sistema. I risultati sono riportati in §7.7.3.
4. **Confronto con l'euristica locale.** Confronto diretto tra il verdetto dell'LLM e il verdetto dell'euristica `compute_verdict_scores` sugli stessi scenari, per isolare il contributo specifico dell'LLM rispetto a un classificatore a regole già calibrato (cfr. §7.6.2). Questo confronto è reso possibile dal fatto che entrambi i sistemi operano

sulle stesse evidenze estratte dai tool MCP: l'euristica è un classificatore a soglie statiche che non vede il ragionamento dell'LLM, mentre l'LLM ha accesso agli stessi score e può confermarli o correggerli sulla base delle evidenze grezze.

Obiettivi secondari. Oltre ai quattro obiettivi principali, la validazione include:

- **Test in modalità generica (cat_e)**, per misurare la capacità del sistema di categorizzare da zero, senza il suggerimento fornito dalla categoria di indagine (cfr. §7.6.4).
- **Test su traffico benigno reale**, per misurare i falsi positivi su traffico non appartenente a CIC-IDS-2017 (cfr. §7.6.5).

7.2 Setup sperimentale

Data source. Dataset CIC-IDS-2017, porzioni Lunedì–Venerdì, inserite nel database MySQL thesis_network tramite la pipeline di ingestione descritta in §6.1. Il database contiene due tabelle correlate logicamente tramite community_id: cic_flows (2.827.901 record, con la colonna label che costituisce la ground truth ufficiale) e ndpi_flows (1.349.098 record, privi di etichette, unica sorgente accessibile ai tool MCP).

La separazione tra le due tabelle (cic_flows per la ground truth, ndpi_flows per l'analisi) garantisce il blind test: l'LLM non ha accesso alle label, e la validazione avviene in una fase post-analitica automatizzata (test_suite.py + utils.py).

Le due tabelle presentano un disallineamento temporale strutturale: i timestamp di ndpi_flows e cic_flows non coincidono, nemmeno per lo stesso community_id. Un'analisi diretta del database conferma il fenomeno: lo stesso community_id può comparire in ndpi_flows su più giorni distinti (per riciclo di porte effimere su IP statici) mentre in cic_flows compare solo in una specifica fascia temporale. Una volta corretto in fase di ingestione l'offset di fuso orario fisso (§6.1), il drift residuo non è costante ma varia da caso a caso (anche per il diverso criterio di terminazione dei flussi dei due estrattori), rendendo impossibile una correzione tramite UPDATE. Per risolvere il problema è stata introdotta la tabella flow_mapping, descritta in Appendice C.

La tabella `flow_mapping` è stata popolata una tantum con la query SQL riportata in Appendice B, eseguita manualmente per ogni giornata del dataset per evitare timeout su tabelle di grandi dimensioni. Il popolamento non è parte della pipeline automatica di ingestion: una volta popolata, la tabella è persistente e viene aggiornata solo in caso di reingestion del dataset.

Modello. Qwen 3.8 27B, erogato tramite endpoint Interhost. L'alternativa GPT-OSS 120B via Groq è stata scartata per i limiti di token del tier gratuito, che rendevano impraticabile qualsiasi validazione (cfr. §5.3).

Middleware. Server MCP custom in Python (`server.py`), che espone 15 tool, tra cui il motore euristico `compute_verdict_scores`. La separazione tra livello dati e livello di inferenza è garantita dal design: il server MCP espone all'LLM esclusivamente i dati provenienti da `ndpi_flows`, mentre i record corrispondenti di `CICFlowMeter` sono usati esclusivamente come ground truth per l'auditing.

Modalità di esecuzione. Guidata, in cui la categoria di indagine è quella dello scenario (`cat_a`–`cat_e`), corrispondente a un'ipotesi di lavoro iniziale fornita all'LLM. La modalità generica (tutte le analisi partono da `cat_e`, senza ipotesi iniziale) è stata testata sull'holdout set; i risultati sono riportati in §7.6.4.

7.3 Costruzione del benchmark

Per valutare il sistema è stato costruito un benchmark di 242 scenari a partire dal dataset CIC-IDS-2017. Uno scenario non è un singolo test, ma un'unità di indagine composta da (i) una finestra temporale di 15 minuti con tolleranza di 2 minuti, (ii) un host target, (iii) una categoria di indagine A–E, (iv) un'etichetta di ground truth. Ogni scenario viene eseguito attraverso un ciclo multi-turno che può richiedere fino a 10 invocazioni di tool MCP, ciascuna corrispondente a una o più query SQL. Il benchmark comprende 242 scenari distinti, per un totale di 407 run effettive (193 test set + 49×3 holdout guidato + 49 holdout generico + 18 traffico reale); il numero complessivo di query SQL eseguite nel corso della campagna è nell'ordine delle migliaia. La numerosità degli scenari riflette la granularità dell'unità di indagine (host/finestra), non il numero di test effettuati. I 242

scenari corrispondono a 529.268 flussi (flussi mappati tra `ndpi_flows` e `cic_flows`), di cui 484.252 con l'etichetta di attacco attesa dallo scenario, 28.357 benigni negli scenari BENIGN e 16.659 nelle finestre d'attacco con etichetta benigna o con un'altra etichetta d'attacco. L'agente non classifica quindi 242 istanze isolate: per ogni scenario deve ricostruire il comportamento di un host a partire da centinaia o migliaia di flussi.

I 242 scenari corrispondono a circa 60 ore di finestre di osservazione (242 finestre da 15 minuti). In particolare, il test set (193 scenari, 283.813 flussi) copre circa 48 ore, mentre l'holdout set (49 scenari, 245.455 flussi) copre circa 12 ore. La procedura completa di generazione, le regole di labeling, le configurazioni delle soglie, la composizione dettagliata e le tabelle di distribuzione per regime e purezza sono riportate in Appendice C.

Il benchmark è suddiviso in due insiemi:

- **Test Set (193 scenari):** utilizzato durante lo sviluppo per calibrare le euristiche deterministiche e ottimizzare i prompt. Le performance su questo insieme vanno lette come accuratezza su dati noti, non come misura di generalizzazione.
- **Holdout Set (49 scenari):** finestre temporali non utilizzate durante la calibrazione, utilizzate come verifica di robustezza. Come discusso in §7.4.2, anche questo set deriva dalla stessa pipeline di labeling del Test Set e quindi non costituisce una validazione esterna in senso stretto. I 49 scenari dell'holdout permettono di riportare metriche di stabilità non influenzate dalla calibrazione, a differenza dei 193 del test set. Molti scenari osservano gli stessi attacchi del test set da un altro punto di vista (§7.4.2)

La verifica di allineamento tra `ndpi_flows` e `cic_flows` conferma che il 99,58% degli scenari (241/242) contiene effettivamente il traffico d'attacco atteso, con un numero di flussi malevoli pari o superiore alla soglia di qualificazione (Appendice C).

Modalità di esecuzione. Il sistema supporta due modalità: guidata, in cui la categoria di indagine è quella dello scenario (`cat.a`–`cat.e`), e generica, in cui tutte le analisi partono da `cat.e` (analisi forense libera, senza ipotesi iniziale). La modalità generica permette di misurare la capacità del sistema di categorizzare da zero, senza il suggerimento fornito dalla categoria di indagine. I risultati in modalità generica sono riportati in §7.6.4.

7.4 Metriche di valutazione

La valutazione del sistema si basa su tre dimensioni complementari: accuratezza della classificazione, coerenza semantica del ragionamento ed efficienza temporale.

Le tre dimensioni sono misurate su tutti gli scenari del benchmark (242 totali), con la distinzione tra test set (§7.6.1–7.6.2) e holdout set (§7.6.3–7.6.4).

7.4.1 Metriche quantitative

Il calcolo delle metriche avviene tramite una procedura di auditing automatizzata (test_-suite.py + utils.py, cfr. §6), che confronta il verdetto dell’LLM con la ground truth ufficiale (CICFlowMeter) e classifica l’esito in Vero Positivo (TP), Vero Negativo (TN), Falso Positivo (FP) o Falso Negativo (FN).

Le metriche sono calcolate separatamente sul test set (193 scenari, §7.6.1–§7.6.2), sull’holdout set guidato (49 scenari $\times$ 3 run, §7.6.3), sull’holdout set generico (49 scenari $\times$ 1 run, §7.6.4) e sul traffico locale (18 finestre, §7.6.5).

Tassonomia e matching. L’LLM produce un verdetto scegliendo tra le stesse cinque macro-categorie della ground truth (WEB_ATTACK_EXPLOIT, SCAN_BRUTEFORCE, DOS_VOLUMETRIC, BEACONING_C2, BENIGN), ma lo accompagna con una motivazione testuale che esplicita le evidenze a supporto. Il matching tra verdetto LLM e ground truth è quindi diretto (stessa tassonomia), e non richiede una mappatura tra livelli di granularità diversi. L’analisi qualitativa si concentra invece sulla coerenza tra motivazione ed evidenze.

Metriche binarie e multi-classe. Sono calcolate due tipologie di metriche:

- **Metriche binarie (minaccia vs benigno):** ogni categoria di attacco è considerata “positiva”, BENIGN è considerata “negativa”. Da questa lettura derivano Accuracy, Precision, Recall e F1-Score globali.
- **Metriche multi-classe (5 classi):** Accuracy stretta (corrispondenza esatta della categoria) e metriche per categoria (Precision, Recall, F1).

Le metriche sull'holdout set in modalità guidata sono calcolate come media su 3 esecuzioni indipendenti. Le metriche sul test set sono calcolate su una singola esecuzione, poiché il set è già stato usato durante la calibrazione e ripetizioni non aggiungerebbero informazione.

Il sistema considera come TP anche le minacce rilevate con categoria errata (es. DOS_-VOLUMETRIC invece di SCAN_BRUTEFORCE), perché il rilevamento della minaccia è considerato il contributo primario; la precisione della categoria è misurata separatamente dall'accuratezza stretta.

7.4.2 Limiti metodologici del benchmark

Ground truth aggregata e finestre sporche. Il confronto tra verdetto e ground truth avviene a livello di finestra host/15 minuti, non di singolo flusso, per i motivi discussi in §5.2.1. Questo introduce una semplificazione: il 48,35% delle finestre del benchmark contiene una miscela di traffico benigno e malevolo, e in 10 casi il segnale malevolo è sotto l'1% dei flussi totali. I dettagli sulla distribuzione dei regimi e della purezza sono in Appendice C.

Ne consegue che il confronto non permette di valutare la precisione sul singolo flusso, ma solo sulla finestra: una finestra è considerata positiva anche in presenza di un solo flusso malevolo. Questo riflette una difficoltà intrinsecamente maggiore del compito per-finestra rispetto a quello per-flusso: l'LLM non deve classificare un'istanza isolata, ma decidere se un host è sotto attacco a partire da una finestra in cui il segnale malevolo può essere minoritario e disperso in un contesto prevalentemente benigno. Le due valutazioni non sono quindi direttamente confrontabili, perché misurano compiti con difficoltà diverse.

Calibrazione e numerosità delle categorie deboli. Il test set (193 scenari) è stato usato durante la calibrazione di soglie e prompt del sistema. Le metriche aggregate (§7.6.1–§7.6.2) non misurano quindi la capacità di generalizzazione su dati mai visti, ma l'accuratezza su un insieme noto durante lo sviluppo. L'holdout set (49 scenari) mitiga parzialmente questo limite, ma proviene dallo stesso dataset (CIC-IDS-2017) e dalla stessa

pipeline di labeling: verifica la robustezza al cambio di punto di osservazione (host vittima anziché host attaccante), ma non la generalizzazione a distribuzioni diverse: molti scenari dell’holdout osservano le stesse sessioni d’attacco già presenti nel test set, poiché CIC-IDS-2017 contiene poche sessioni distinte per tipo di attacco. Per una validazione esterna sarebbe necessario un dataset completamente separato (es. CSE-CIC-IDS2018), non disponibile in questo lavoro.

Differenza nella costruzione dei due set. Il test set e l’holdout set non sono direttamente confrontabili per un secondo motivo, oltre alla non indipendenza della pipeline. La prima e la seconda generazione (che compongono il test set) raggruppano i flussi per `src_ip`: l’unità di analisi è l’host che origina il traffico, coerentemente con la struttura di `cic_flows`, dove l’etichetta è associata all’host sorgente. La terza generazione (che compone l’holdout set) raggruppa invece per `dst_ip`, perché (a) opera su `flow_mapping`, una tabella in cui l’etichetta è già associata al flusso dall’esterno via Community ID, e (b) la `dst_ip` rappresenta più naturalmente l’unità di analisi dell’agente LLM, che indaga un host vittima e non un host sorgente. Ne consegue che il test set tende a contenere scenari in cui il target dell’indagine è l’attaccante, mentre l’holdout set tende a contenere scenari in cui il target è la vittima. Di conseguenza, una parte degli scenari dell’holdout corrisponde agli stessi attacchi del test set osservati dal lato della vittima; i flussi complessivi (529.268) non sono quindi tutti distinti. Le due scelte sono complementari ma non direttamente confrontabili: il confronto tra le metriche sui due set non isola quindi solo l’effetto della calibrazione, ma riflette anche questa differenza di costruzione.

Distribuzione delle categorie e impatto sulle metriche aggregate. Le categorie meno rappresentate restano `DOS_VOLUMETRIC` (18 scenari totali), `WEB_ATTACK_EXPLOIT` (13) e `SCAN_BRUTEFORCE` (32), a fronte di `BEACONING_C2` (67) e `BENIGN` (112). Questo riflette la distribuzione reale del dataset CIC-IDS-2017, in cui il traffico benigno e il beaconing Botnet sono molto più abbondanti degli attacchi mirati. Ne consegue che le metriche aggregate — calcolate su tutti gli scenari senza pesare le categorie — sono dominate dalle due classi più rappresentate: `BENIGN` e `BEACONING_C2` costituiscono insieme l’84% del test set (162 su 193), mentre le tre categorie di attacco mirato

ne coprono il 16% (31 su 193). Un errore su uno scenario di WEB_ATTACK_EXPLOIT ha quindi un impatto marginale sull'Accuracy complessiva, ma un peso operativo molto maggiore di un errore su BENIGN. Le metriche per categoria (Tabella 7.4) sono pertanto da preferire all'aggregato per valutare la robustezza del sistema sulle classi di attacco.

Caso SCN_CAT_B.3. Il benchmark include un caso (SCN_CAT_B.3) in cui l'attacco è sotto la soglia di qualificazione della categoria B (10 flussi malevoli, Appendice C), pur essendo stato generato regolarmente: la finestra contiene 208 flussi BENIGN e 2 flussi DoS slowloris. Questo scenario non è stato rimosso, poiché rappresenta una condizione realistica: un attacco a bassa intensità nella finestra temporale che il sistema può legittimamente classificare come BENIGN. La sua presenza impedisce che le metriche aggregate siano artificialmente ottimistiche.

7.4.3 Coerenza semantica e grounding

Oltre ai numeri, viene analizzata la capacità dell'LLM di produrre output logicamente corretti e privi di allucinazioni. Il grounding è garantito dal design: l'LLM non ha accesso diretto al database, ma solo ai risultati dei tool MCP prodotti da query SQL deterministiche. Il verdetto dello Stage 2 deve coincidere con quello dello Stage 1 (vincolo di coerenza Stage 1 → Stage 2, cfr. §6.4). L'analisi qualitativa è stata condotta manualmente su un campione di 15 scenari, selezionati per coprire tutte e cinque le categorie e i casi di divergenza tra LLM ed euristica (§7.6.2). Per ciascuno scenario sono stati valutati: presenza di affermazioni non supportate dai dati, coerenza tra verdetto e motivazione, chiarezza della motivazione.

7.4.4 Analisi delle prestazioni

La latenza end-to-end è scomposta in tre componenti:

$$T_{Total} = T_{sql} + T_{mcp} + T_{llm} \quad (7.1)$$

dove:

- T_{sql} è il tempo di esecuzione delle query SQL sul database MySQL (misurato lato server);
- T_{mcp} è l'overhead del middleware MCP (serializzazione, trasporto stdio, cache);
- T_{llm} è il tempo di inferenza del modello (chiamate API, generazione dei token, tool calling).

La scomposizione permette di identificare il collo di bottiglia del sistema. Le metriche di tempo sono tracciate dal client per ogni scenario. I risultati sono riportati in §7.7.3 .

7.5 Scenari di test (categorie A–E)

Le cinque categorie di indagine corrispondono a profili investigativi distinti, presentati all'utente tramite un menu numerico all'avvio di `client.py`. Ogni categoria fornisce al modello un'ipotesi di lavoro iniziale (definita in `prompts.FOCUS_CATEGORIE_CONTENT`), ma non restringe i tool disponibili né vincola il verdetto finale: il prompt richiede esplicitamente di confermare, correggere o smentire l'ipotesi sulla base delle evidenze raccolte.

Le categorie sono state progettate per coprire le principali famiglie di minaccia rappresentate nel dataset CIC-IDS-2017 e per guidare l'indagine verso i segnali più discriminanti per ciascuna famiglia. In tutte le sottosezioni seguenti, le etichette CIC corrispondenti indicano le classi del dataset: esse non coincidono con i verdetti prodotti dal sistema.

Nota metodologica. Le categorie di indagine (A–E) non vanno interpretate come procedure vincolanti. Esse forniscono all'LLM un'ipotesi di lavoro iniziale e un suggerimento su quali segnali privilegiare, ma il prompt richiede esplicitamente di confermare, correggere o smentire l'ipotesi sulla base delle evidenze. L'autonomia del modello resta centrale: se le evidenze raccolte portano in una direzione diversa, l'LLM può (e deve) ignorare l'ipotesi di indagine iniziale. La categoria è quindi un orientamento investigativo che restringe lo spazio di ricerca dell'agente, senza costituire un vincolo procedurale: l'LLM resta libero di concludere in una categoria diversa se le evidenze lo giustificano.

7.5.1 Categoria A — analisi strutturale e applicativa L7

La categoria A si applica quando l'ipotesi di lavoro prevalente è un attacco applicativo mirato: exploit Web, brute force su endpoint applicativi, o attacchi L7 a basso volume. L'agente segue una cascata di priorità: prima verifica la presenza di pattern di exploit confermati (path traversal, XSS, SQL injection, header o metodi HTTP malformati), poi di tentativi di brute force su porte di gestione, infine di volumi anomali. Se il primo criterio è soddisfatto, l'agente emette il verdetto `WEB_ATTACK_EXPLOIT` senza ulteriori ispezioni (early stop). Il volume da solo non è sufficiente a distinguere un brute force lento da un flood: essendo $\text{volume} = \text{RPS} \times \text{durata}$, un brute force lento produce pochi flussi distribuiti su lungo tempo, mentre un flood produce molti flussi concentrati in pochi secondi. Per questo la cascata usa il RPS come discriminante temporale. Il verdetto di default, in assenza di tutti i criteri, è `BENIGN`. I segnali distintivi della categoria includono anomalie di entropia del payload, endpoint di login mirati, traffico asimmetrico e richieste concentrate su pochi target Web.

7.5.2 Categoria B — monitoraggio volumetrico

La categoria B si applica quando l'host target mostra un volume di traffico anomalo e l'ipotesi prevalente è un attacco di tipo Denial of Service. A differenza della categoria A, i criteri non sono sequenziali: ciascuno è di per sé sufficiente a emettere il verdetto `DOS_VOLUMETRIC`. L'agente valuta tre criteri alternativi: (1) DoS volumetrico L4/L7, basato su PPS medi o di picco, RPS e concentrazione di flussi Web; (2) banda satura, basata sul byte rate medio e sul rapporto tra picco e media; (3) dispersione estrema delle porte sorgente, basata sul rapporto di porte effimere e sul numero di porte sorgente uniche. Per il terzo criterio esiste un'eccezione: se la frazione di flussi verso porte di infrastruttura LAN è elevata, la dispersione è attribuibile a traffico di dominio (es. DNS interno) e non costituisce segnale di DoS. Il verdetto di default è `BENIGN`. I segnali distintivi della categoria includono PPS elevati, burst, dispersione di porte sorgente, flussi Slowloris e concentrazione di richieste Web.

7.5.3 Categoria C — profiling endpoint

La categoria C si applica quando l'host target sembra impegnato in attività di ricognizione (mappatura di porte e servizi aperti) o in un attacco lento e a basso volume pensato per non attivare le soglie di rate classiche. L'agente valuta due criteri alternativi. Il primo è PortScan o BruteForce: la presenza di un Port Scan strutturato con molte porte distinte contattate in modo ripetuto, oppure tentativi di autenticazione falliti o ripetuti su servizi di gestione. Il tool `search_connection_attempts` svolge qui un ruolo di conferma: verifica che le porte multiple siano contattate sulla stessa destinazione (scan) e non in fan-out su molte destinazioni diverse. Il secondo criterio è Slow-Rate L7 / Slowloris: sessioni HTTP/HTTPS di lunga durata con volume di byte ridotto. In questo caso il verdetto dipende dal numero di sessioni lente: poche sessioni isolate e concentrate su endpoint specifici configurano un tentativo di sondaggio mirato (`WEB_ATTACK_EXPLOIT`), mentre molte sessioni lente simultanee configurano una saturazione delle risorse del server (`DOS_VOLUMETRIC`). Il verdetto di default è `BENIGN`. I segnali distintivi per questa categoria includono la distribuzione delle porte di destinazione e la concentrazione dei tentativi di connessione su una coppia IP.

7.5.4 Categoria D — analisi comportamentale

La categoria D si applica quando l'ipotesi prevalente è la presenza di un impianto malware che comunica periodicamente con un'infrastruttura di comando e controllo (C2) esterna. A differenza delle categorie precedenti, l'analisi si concentra sul comportamento temporale e sulla persistenza, più che sul volume: il traffico C2 è tipicamente a basso volume ma con pattern temporali regolari. I tre criteri sono alternativi e ciascuno è di per sé sufficiente a emettere il verdetto `BEACONING_C2`; il verdetto finale resta comunque soggetto ai guardrail (whitelist, deroga `BENIGN`) discussi in §6.4. Il primo è il beaconing C2 classico, rilevato tramite l'Anomaly Score del tool `detect_beaconing` o la presenza di tag `CONFIRMED_BEACONING_C2`. Il secondo riguarda le comunicazioni verso porte non standard tipicamente usate da proxy o C2, anche in presenza di un Anomaly Score basso: alcuni impianti introducono jitter per abbassare artificialmente il coefficiente di variazione

(CV) e mascherare la periodicità, quindi il segnale va considerato malevolo anche quando lo score è basso. Il terzo criterio è il DNS tunneling, ovvero l'incapsulamento di traffico arbitrario in query e risposte DNS, tecnica usata per esfiltrare dati o veicolare comandi C2 aggirando i firewall. Le firme tipiche includono query molto lunghe, alta entropia nei nomi di dominio, uso anomalo di record TXT/CNAME e volumi elevati di query verso una singola destinazione. Il verdetto di default è BENIGN. I segnali distintivi della categoria includono periodicità (CV), comunicazioni verso IP esterni, porte C2 sospette e flussi Bot/C2 anche isolati.

7.5.5 Categoria E — analisi forense libera

La categoria E si applica quando non c'è un'ipotesi di lavoro predefinita sull'host target: l'agente deve esplorare il traffico senza un'ipotesi iniziale e determinare la natura dell'attività seguendo un ordine di priorità fisso. La valutazione segue una sequenza ordinata di quattro priorità: (1) exploit Web o L7, (2) scan o brute force, (3) DoS volumetrico o DoS L7, (4) beaconing C2 o botnet. Come nelle altre categorie sequenziali, la priorità 1 assorbe i casi di attacco applicativo confermato; solo in sua assenza si passa alla priorità 2, e così via. Anche in questo caso l'ordine è un suggerimento, non un vincolo. In assenza di violazioni esplicite delle regole, l'unica risposta valida è BENIGN: non viene forzato un verdetto di attacco solo perché uno score isolato è leggermente sopra zero.

La categoria E ha un ruolo specifico nella validazione: è utilizzata per il test in modalità generica, in cui tutte le analisi partono da questa categoria per misurare la capacità del sistema di categorizzare da zero, senza il suggerimento fornito da un'ipotesi di lavoro iniziale.

7.5.6 Sintesi delle categorie

La Tabella 7.1 riassume le cinque categorie con il contesto, l'ipotesi di lavoro e i segnali distintivi.

Categoria	Contesto	Ipotesi iniziale	Segnali distintivi
A (Analisi L7)	Traffico Web (80/443/8080/8443)	Attacco applicativo L7 (exploit, brute force, scanning)	Entropia payload, endpoint login, traffico asimmetrico
B (Monitoraggio Volumetrico)	Volume anomalo (PPS/banda)	DoS/DDoS volumetrico	PPS elevati, burst, dispersione porte sorgente
C (Profiling Endpoint)	Ricognizione / attacco lento	Port Scan	Distribuzione porte, tentativi concentrati
D (Analisi Comportamentale)	Comunicazioni periodiche verso l'esterno	Beaconing C2, Botnet, DNS Tunneling	Periodicità (CV), porte C2, flussi Bot/C2 isolati
E (Analisi Forense Libera)	Nessuna ipotesi predefinita	Analisi forense libera (cascata deterministica)	Nessun segnale predefinito

Tabella 7.1: Sintesi delle cinque categorie di indagine.

7.6 Risultati sperimentali

I risultati presentati in questa sezione si riferiscono alla campagna di benchmark condotta con il modello Qwen 3.8 27B. La campagna si compone di più parti, ciascuna su un set diverso:

- **Test set (193 scenari, modalità guidata):** benchmark principale su cui sono calcolate le metriche aggregate (§7.6.1–§7.6.2).
- **Holdout set guidato (49 scenari $\times$ 3 run):** verifica di robustezza su finestre non utilizzate durante la calibrazione, in modalità guidata (§7.6.3).
- **Holdout set generico (49 scenari $\times$ 1 run):** verifica della capacità di categorizzazione senza ipotesi iniziale (§7.6.4).
- **Traffico locale (18 finestre):** verifica dei falsi positivi su traffico benigno reale (§7.6.5).

I risultati sono integrati da un'analisi aggregata del grounding, cioè del grado in cui i verdeti emessi dal sistema risultano ancorati a evidenze concrete. Tale analisi è condotta a livello aggregato su tutte le run ed è descritta in §7.6.7.

7.6.1 Test set (193, guidata): metriche quantitative

Su 193 scenari, le metriche globali sono riportate nella Tabella 7.2.

Metrica	Valore
Accuracy	98,96%
Precision	98,90%
Recall	98,90%
F1-Score	98,90%
Accuracy stretta (categoria esatta)	98,96%

Tabella 7.2: Metriche globali di classificazione (193 scenari, 283.813 flussi).

Prestazioni quantitative. Il sistema raggiunge un'Accuracy del 98,96% e una Precision del 98,90% su 193 scenari, con 1 falso positivo e 1 falso negativo. Questi valori vanno letti con cautela, per tre motivi principali:

- Il 48,35% delle finestre contiene sia flussi benigni sia malevoli, e in 10 casi il segnale malevolo è sotto l'1% dei flussi totali (Appendice C).
- Le categorie WEB_ATTACK_EXPLOIT (13 scenari), DOS_VOLUMETRIC (18) e SCAN_BRUTEFORCE (32) hanno un numero ridotto di scenari.

Gli scenari con segnale debole sono sistematicamente più difficili da classificare, come dimostrato dall'analisi qualitativa (§7.6.6).

Nella Tabella 7.3 è riportata la matrice di confusione sulle 5 classi.

Reale \ Predetta	BENIGN	WEB	SCAN	DOS	BEACON
BENIGN	101	1	0	0	0
WEB_ATTACK_EXPLOIT	0	6	0	0	0
SCAN_BRUTEFORCE	0	0	16	0	0
DOS_VOLUMETRIC	0	0	0	9	0
BEACONING_C2	1	0	0	0	59

Tabella 7.3: Matrice di confusione (193 scenari, 5 classi).

Nota sul non-determinismo e sulla stabilità delle metriche. Le metriche riportate in questa sezione (Accuracy 98,96%, Precision 98,90%, 1 FP e 1 FN) si riferiscono a

una singola esecuzione della campagna di validazione sul test set. Il sistema, basato su un LLM con campionamento probabilistico in fase di generazione (temperatura > 0), non è completamente deterministico: a parità di input, scenari ambigui possono produrre verdetti diversi in esecuzioni diverse. Questo fenomeno è documentato in §7.7 (tipo D) e discusso in §8.4.

In particolare, il falso negativo riportato nella Tabella 7.3 (uno scenario BEACONING_C2 classificato come BENIGN) scompare se lo scenario viene rieseguito singolarmente: lo stesso input, in una diversa esecuzione, produce il verdetto corretto. Si tratta quindi di un caso di non-determinismo residuo, non di un errore sistematico.

Di conseguenza, i valori di Precision e Recall (98,90% ciascuno) e il conteggio 1 FP / 1 FN vanno letti come risultati osservati in una specifica campagna, e non come valori assoluti e definitivi del sistema. La stabilità complessiva resta comunque elevata: sull’holdout set, eseguito 3 volte in modalità guidata, l’Accuracy ha una deviazione standard di circa 1,2 punti percentuali (§7.6.3), e la coerenza Stage 1 → Stage 2 è al 99,02% (§7.6.7). Il non-determinismo si manifesta quindi a livello di singolo scenario, non di sistema.

Gli errori si concentrano su:

- BEACONING_C2: 1 falso negativo (classificato come BENIGN).
- BENIGN: 1 falso positivo (classificato come WEB_ATTACK_EXPLOIT).

La Tabella 7.4 riporta Precision, Recall e F1 per ciascuna categoria, con il relativo supporto (numero di scenari).

Categoria	Supporto	Precision	Recall	F1
WEB_ATTACK_EXPLOIT	6	85,71%	100%	92,31%
DOS_VOLUMETRIC	9	100%	100%	100%
SCAN_BRUTEFORCE	16	100%	100%	100%
BEACONING_C2	60	100%	98,33%	99,16%
BENIGN	102	99,02%	99,02%	99,02%

Tabella 7.4: Metriche di classificazione per categoria e supporto (193 scenari).

Tutte le categorie di attacco raggiungono Precision 100% e Recall 100%, con l'eccezione di BEACONING_C2 (Recall 98,33%, per l'FN) e WEB_ATTACK_EXPLOIT (Precision 85,71%, per il FP). La categoria BENIGN ha Precision e Recall 99,02%.

7.6.2 Confronto LLM vs euristica sul test set

Il sistema è stato confrontato con l'euristica a regole lato server (compute_verdict_scores) sugli stessi 193 scenari. La Tabella 7.5 riporta la distribuzione dei casi di conferma, correzione, peggioramento ed errore condiviso.

Esito	Conteggio	%
LLM conferma euristica (entrambi giusti)	188	97,41%
LLM conferma euristica (entrambi sbagliati)	1	0,52%
LLM corregge euristica (euristica sbaglia, LLM giusto)	3	1,55%
LLM peggiora euristica (euristica giusta, LLM sbaglia)	1	0,52%
Totale	193	100%

Tabella 7.5: Confronto tra verdetto dell'LLM e verdetto dell'euristica (193 scenari).

Il confronto è reso possibile dal fatto che entrambi i sistemi operano sulle stesse evidenze estratte dai tool MCP: l'euristica è un classificatore a soglie statiche che non ha accesso al ragionamento dell'LLM, mentre l'LLM ha accesso agli stessi score e può confermarli o correggerli sulla base delle evidenze grezze. La Tabella 7.5 misura pertanto il contributo incrementale del ragionamento dell'LLM rispetto alla sola regola statica. I casi di divergenza sono analizzati qualitativamente in §7.6.6 secondo una tassonomia a cinque tipi (A–E), e la loro interpretazione è sviluppata in §7.7.

7.6.3 Holdout set (49, guidata, 3 run)

L'holdout set è stato eseguito 3 volte in modalità guidata, per verificare la stabilità del sistema su finestre non utilizzate durante la calibrazione. I risultati sono riportati nella Tabella 7.6.

Run	Accuracy	Precision	Recall	F1	Acc. stretta
49_1	95,92%	97,44%	97,44%	97,44%	95,92%
49_2	95,92%	97,44%	97,44%	97,44%	95,92%
49_3	97,96%	100%	97,44%	98,70%	97,96%
Media	96,60%	98,29%	97,44%	97,86%	96,60%

Tabella 7.6: Metriche di classificazione sull’holdout set (49 scenari, 245.455 flussi).

Nota: nella run 49_1, un caso è stato abortito per timeout del server. Il caso è stato rieseguito manualmente e sostituito a quello fallito nei 49. Un secondo caso ha subito un rallentamento del server ma è stato recuperato senza rigenerazione: la sua latenza risente del blocco e contribuisce ad alzare la mediana end-to-end della run. Il valore di latenza della run 49_1 (Tabella 7.14) va quindi letto come influenzato dall’evento infrastrutturale, non come misura pulita del sistema.

La variabilità tra le run è contenuta: l’Accuracy varia tra il 95,92% e il 97,96%, con una deviazione standard di circa 1,2 punti percentuali. La run 49_3 ha una Precision del 100%, mentre le run 49_1 e 49_2 hanno 1 FP ciascuna: è lo stesso scenario, classificato erroneamente in due run e correttamente nella terza, a conferma del non-determinismo residuo discusso in §7.6.1 e §7.7 (tipo D).

La Tabella 7.7 riporta le metriche per categoria (media sulle 3 run).

Categoria	Supporto	Precision	Recall	F1
WEB_ATTACK_EXPLOIT	7	100%	85,71%	92,31%
DOS_VOLUMETRIC	9	100%	100%	100%
SCAN_BRUTEFORCE	16	100%	100%	100%
BEACONING_C2	7	91,67%	100%	95,55%
BENIGN	10	90,30%	93,33%	91,75%

Tabella 7.7: Metriche per categoria (holdout, media su 3 run).

7.6.4 Holdout set (49, generica)

L'holdout set è stato eseguito 1 volta in modalità generica (tutte le analisi partono da cat_e, senza ipotesi iniziale). I risultati sono riportati nella Tabella 7.8.

Metrica	Valore
Accuracy	95,92%
Precision	97,44%
Recall	97,44%
F1-Score	97,44%
Accuratezza stretta	95,92%

Tabella 7.8: Metriche di classificazione sull'holdout set (49 scenari, modalità generica).

Il risultato in modalità generica è identico a quello delle run guidate 49_1 e 49_2 (95,92% / 97,44% / 97,44% / 97,44%). Questo indica che, su questo set, il suggerimento della categoria non influenza significativamente il comportamento del sistema: l'LLM arriva allo stesso verdetto partendo da cat_e o dalla categoria specifica. Questo è un risultato importante perché suggerisce che il sistema è in grado di categorizzare da zero senza il bias dell'ipotesi iniziale.

Nota sulla latenza. La modalità generica mostra una latenza LLM mediana più alta rispetto alle run guidate (41,02 s vs 35–36 s, cfr. Tabella 7.14); la discussione è in §7.7.3

La Tabella 7.9 riporta le metriche per categoria in modalità generica.

Categoria	Supporto	Precision	Recall	F1
WEB_ATTACK_EXPLOIT	7	100%	85,71%	92,31%
DOS_VOLUMETRIC	9	100%	100%	100%
SCAN_BRUTEFORCE	16	100%	100%	100%
BEACONING_C2	7	87,50%	100%	93,33%
BENIGN	10	90,00%	90,00%	90,00%

Tabella 7.9: Metriche per categoria (holdout, modalità generica).

7.6.5 Test su traffico reale (18 finestre)

Traffico reale. Per valutare i falsi positivi su traffico non appartenente al dataset CIC-IDS-2017, è stato condotto un test su 4,5 ore di traffico reale totalmente benigno (18 finestre da 15 minuti, su 2 host distinti) generato dall'autrice. I risultati sono riportati nella Tabella 7.10.

Metrica	Valore
Finestre totali	18
Verdetto BENIGN corretto	18
Falsi positivi	0
Precision	100%

Tabella 7.10: Falsi positivi su traffico reale.

Il sistema non produce falsi positivi su 18 finestre. La whitelist dei servizi è stata estesa per coprire servizi moderni (2026) assenti o irriconoscibili nel dataset del 2017, come claude.ai (lanciato nel 2023) e le attuali firme di traffico di Teams e Google. Le soglie euristiche, invece, non sono state ricalibrate e restano quelle definite su CIC-IDS-2017.

7.6.6 Analisi qualitativa dei disaccordi sull'intera campagna di validazione

L'analisi dei disaccordi tra il verdetto dell'LLM e quello dell'euristica `compute_verdict_scores` rivela pattern distinti e sistematici. L'analisi è condotta sull'intera campagna di validazione (260 scenari: 193 del test set, 49 dell'holdout set e 18 del set su traffico reale), non solo sul test set, così da catturare anche i casi di non-determinismo e le correzioni incrociate che emergono su finestre non utilizzate durante la calibrazione. I pattern sono riassunti nella Tabella 7.11 e discussi per categoria in §7.7.

Tipo	N. casi	Scenari rappresentativi
A. Correzioni di FP euristici (LLM giusto, euristica sbagliata)	4/260	LOCAL_CAT_E_015, TEST_CAT_E_070, TEST_CAT_E_119, VAL_CAT_E_061
B. Correzioni di FN dell'LLM via post-euristica (Stage 1 BENIGN → Stage 2 attacco)	4/260	GEN_CAT_D_0001, GEN_CAT_D_0003, GEN_CAT_D_0004, VAL_CAT_D_003
C. FP sistemico da regola tassativa (entrambi sbagliano, esito stabile)	1/260	VAL_CAT_E_007
D. Non-determinismo su scenari ambigui (verdicti diversi su stesso input)	2/260	GEN_CAT_E_0376, TEST_CAT_D_033
E. FN condiviso (entrambi sbagliano, categorie diverse)	1/260	GEN_CAT_A_0001

Tabella 7.11: Classificazione dei disaccordi LLM vs euristica sull'intera campagna di validazione (260 scenari).

Nota alla Tabella 7.11. I prefissi TEST_, VAL_ identificano scenari del test set (193 scenari, usati durante la calibrazione di soglie e prompt); il prefisso GEN_ identifica scenari dell'holdout set (49 scenari, finestre non utilizzate durante la calibrazione) e il prefisso LOCAL_ identifica scenari appartenenti al set sul traffico reale. L'analisi dei disaccordi copre quindi l'intera campagna di validazione (260 scenari), non solo il test set. Si noti che i 3 casi GEN_* del tipo B si verificano esclusivamente in modalità generica: in modalità guidata, gli stessi scenari non producono direttamente FN. La distinzione è discussa in §7.7.

La Tabella 7.11 classifica i disaccordi osservati sull'intera campagna di validazione in cinque tipi: correzioni di falsi positivi euristici (4 casi), correzioni di falsi negativi dell'LLM via post-euristica (4 casi), falso positivo sistemico da regola tassativa (1 caso), non-determinismo su scenari ambigui (2 casi), e falso negativo condiviso (1 caso). L'analisi interpretativa di ciascun tipo è sviluppata in §7.7.

7.6.7 Analisi aggregata del grounding

Per valutare la qualità del grounding del sistema, è stata condotta un'analisi automatizzata su tutti i log di indagine prodotti durante la campagna di benchmark (407 run totali: $193 + 49 \times 3 + 49 + 18$). L'analisi si è concentrata su tre indicatori chiave:

1. **Validità del formato JSON:** la percentuale di report finali (Stage 2) che contengono un verdetto finale formalmente estraibile e ben formato.
2. **Citazione di hostname nel Thought:** la percentuale di Thought (ragionamenti) che menzionano almeno un hostname, indicando un'analisi basata su dati concreti e non solo su metriche aggregate.

3. **Coerenza Stage 1 → Stage 2:** la percentuale di scenari in cui il verdetto dello Stage 1 coincide con quello dello Stage 2, verificando la stabilità del ragionamento tra le due fasi della pipeline.

I risultati aggregati sono riportati nella Tabella 7.12.

Indicatore	Risultato
JSON Stage 2 valido	407/407 (100%)
Thought cita hostname	125/407 (30,71%)
Coerenza Stage 1 → Stage 2	403/407 (99,02%)
Allucinazioni rilevate	[] su 15 log

Tabella 7.12: Analisi aggregata del grounding (407 run).

Discussione. I risultati mostrano un sistema con un grounding solido su tre fronti.

- **JSON valido al 100%:** la pipeline Stage 2 produce sempre un output finale estraibile e ben formato. In tutti gli scenari il verdetto è stato emesso correttamente e catturato dall'analisi automatizzata, senza interruzioni o errori di parsing.
- **Coerenza Stage 1 → Stage 2 al 99,02%:** quando lo Stage 1 produce un verdetto identificabile, lo Stage 2 lo ricopia fedelmente rispettando il vincolo di coerenza imposto dal prompt. I 4 casi con coerenza false (Stage 1 BENIGN → Stage 2 BEACONING_C2) non sono errori, ma correzioni del sistema: il meccanismo di post-euristica ha forzato il verdetto euristico perché la deroga BENIGN non era ammessa. In tutti e 4 i casi, la ground truth conferma che il verdetto forzato era corretto (TP).
- **Citazione di hostname al 30,71%:** meno di un terzo dei ragionamenti menziona un hostname specifico. Questo indica che, in oltre due terzi dei casi, il modello si basa prevalentemente su metriche aggregate (PPS, porte, CV di periodicità, dispersioni) senza approfondire la natura specifica dei servizi contattati. Si tratta di un'area di miglioramento per la qualità del grounding: un Thought che cita esplicitamente hostname e provider è più facilmente verificabile da un analista umano e riduce il rischio di allucinazioni.

Verifica manuale delle allucinazioni. Il campo `allucinazioni_rilevate` è sempre vuoto. L'analisi automatica delle allucinazioni richiede un confronto semantico tra `Thought` e output dei tool, non implementabile con semplici espressioni regolari. È stata pertanto condotta un'analisi manuale su un campione di 15 log (5 per set: 193, 49, 18), selezionati per coprire le diverse categorie di verdetto e le situazioni di maggiore complessità (casi di deroga a un suggerimento euristico, verdetto BENIGN con citazione di `hostname`, verdetti di attacco). Per ciascun log sono state verificate tutte le affermazioni fattuali del `Thought` (`hostname`, IP, metriche, provider) confrontandole con gli output dei tool corrispondenti. In nessuno dei 15 log analizzati è stata rilevata un'allucinazione: ogni `hostname`, IP e metrica citata nel `Thought` trova riscontro in almeno un output di tool.

Coerenza semantica e grounding. Su un campione di 15 scenari non sono state rilevate allucinazioni né incoerenze tra verdetto e motivazione. Il vincolo di grounding è garantito dal design: l'LLM opera esclusivamente sui risultati dei tool MCP, prodotti da query SQL deterministiche.

7.7 Analisi dei risultati

I risultati della campagna di benchmark permettono osservazioni più precise sul comportamento del sistema e sul contributo specifico dell'LLM rispetto all'euristica.

7.7.1 Contributo dell'LLM rispetto all'euristica

Il confronto tra il verdetto dell'LLM e quello del motore euristico `compute_verdict_scores` sugli stessi 193 scenari (Tabella 7.5) consente di isolare il contributo specifico del ragionamento del modello rispetto a un classificatore a soglie statiche. La prima evidenza è che il verdetto euristico viene confermato nel 97,41% dei casi (188/193). I casi di divergenza sull'intera campagna di validazione (260 scenari) sono classificati in cinque tipi (Tabella 7.11), approfonditi di seguito e nella Tabella 7.13.

Tabella 7.13: Tipi di disaccordo tra LLM ed euristica.

Tipo	N.	Descrizione	Contromisura
A	4	Correzione di FP euristici su traffico legittimo	Riconoscimento semantico dei servizi (LLM)
B	4	Falso negativo su beaconing C2 confermato	Post-euristica forza il verdetto
C	1	Falso positivo sistemico da regola tassativa	Rivedere la regola su Web Brute Force
D	2	Non-determinismo su scenari ambigui	Voto multi-run o tie-breaking più forte (§9.3)
E	1	Falso negativo condiviso su Web Attack a basso volume	Soglia euristica più sensibile + ispezione dei metadati HTTP

A. Correzione di falsi positivi euristici. I 4 casi di tipo A condividono un pattern ricorrente: l'euristica segnala una minaccia (BEACONING_C2 o DOS_VOLUMETRIC) su traffico che l'LLM riconosce come legittimo — heartbeat applicativi nel primo caso, attività di backup/streaming nel secondo. Il contributo del modello è il riconoscimento semantico della natura del traffico, che l'euristica a soglie statiche non può fornire.

B. Correzione di falsi negativi via post-euristica. Il meccanismo di post-euristica (§6.4) ha corretto 4 FN dell'LLM, tutti BEACONING_C2. In questi casi lo Stage 1 concludeva BENIGN ignorando un segnale euristico pieno; il client forzava il verdetto euristico nello Stage 2. La ground truth conferma la correttezza in tutti e 4 i casi. Va notato che 3 dei 4 casi appartengono all'holdout set (GEN_*) e si verificano esclusivamente in modalità generica: in assenza di un'ipotesi di lavoro iniziale (cat_e), l'LLM tende a sottovalutare il segnale di beaconing e a chiudere con BENIGN, rendendo necessario l'intervento della post-euristica. In modalità guidata, gli stessi scenari non producono FN. Questo suggerisce che la categoria di indagine, pur non essendo un vincolo sul verdetto, riduce i falsi negativi del modello guidandone l'attenzione verso i segnali rilevanti.

C. Falso positivo sistemico da regola tassativa. Il sistema produce 1 FP (BENIGN →

WEB_ATTACK_EXPLOIT) di natura sistemica: il polling applicativo legittimo coincide con la configurazione che il prompt classifica come Web Brute Force tassativo. L'LLM è vincolato a confermarlo. È la controparte simmetrica del tipo E.

D. Non-determinismo su scenari ambigui. Su input ambigui lo stesso scenario può produrre verdetti diversi (es. GEN_CAT_E_0376: 3 volte BEACONING_C2, 1 volta BENIGN). La causa è la natura probabilistica del modello; i componenti deterministici non introducono varianza. È questo il fenomeno alla base della nota sul non-determinismo in §7.6.1.

E. Falso negativo condiviso. In 1 caso (GEN_CAT_A_0001) entrambi i componenti sbagliano: l'euristica propone DOS_VOLUMETRIC, l'LLM conclude BENIGN, la ground truth è WEB_ATTACK_EXPLOIT. È un attacco Web a basso volume che sfugge a entrambi. Come il tipo C, origina da un problema di calibrazione su WEB_ATTACK_EXPLOIT, ma in direzione opposta: il tipo C è un FP da regola troppo aggressiva, mentre il tipo E è un FN da soglia troppo conservativa.

Il contributo incrementale dell'LLM non si misura in termini di accuratezza complessiva — dove il classificatore a regole è già competitivo — ma su tre dimensioni: correzione di FP sistematici (tipo A), produzione di motivazione tecnica verificabile, selezione autonoma dei tool. I limiti residui sono sistemici (tipo C), probabilistici (tipo D) e condivisi (tipo E): il primo è correggibile agendo sulle regole, gli altri due richiedono interventi architetturali discussi in §9.3.

7.7.2 Comportamento su scenari con segnale debole

L'analisi qualitativa mostra che il comportamento del sistema su segnali deboli non è uniforme, e dipende dalla categoria di minaccia coinvolta. Su BEACONING_C2 il sistema privilegia la prudenza: le whitelist estese e la deroga BENIGN documentata contro i falsi positivi lo portano a non emettere un verdetto di attacco quando il segnale periodico è compatibile con un heartbeat applicativo. Su WEB_ATTACK_EXPLOIT, al contrario, la regola tassativa del prompt impone l'emissione di un verdetto di attacco anche in presenza di un segnale debole (concentrazione di richieste su pochi target), come discusso nel tipo C (paragrafo precedente). Il sistema non ha quindi una preferenza globale per la

prudenza: ha due regimi distinti, uno conservativo su BEACONING_C2 e uno aggressivo su WEB_ATTACK_EXPLOIT, entrambi deliberati.

7.7.3 Latenza e scalabilità

La tabella 7.14 mostra le latenze mediane scomposte nei vari componenti, come spiegato in §7.4.4, per i vari set valutati.

Si noti che la somma delle componenti LLM, MCP e SQL non esaurisce la latenza end-to-end: la differenza ($\approx 12\text{--}20$ s, variabile in funzione della dimensione dei payload MCP scambiati) è attribuibile all'overhead di orchestrazione del client e alla serializzazione dei payload MCP tra un turno e l'altro, non misurati separatamente in questa sede.

Set	Mediana LLM (s)	Mediana MCP (s)	Mediana SQL (s)	Mediana end- to-end (s)
193	34,54	1,71	0,12	55,28
49_1	35,79	2,42	0,16	58,13
49_2	35,52	2,26	0,12	57,27
49_3	36,24	2,14	0,13	55,80
49_generica	41,02	2,36	0,10	55,23
18	36,97	0,33	0,01	49,74

Tabella 7.14: Latenza mediana per set (in secondi).

Nota: la riga relativa alle 18 finestre di traffico reale si riferisce a scenari tutti BENIGN e analizzati con ipotesi generica (cat_e). I valori di latenza MCP e SQL, molto inferiori rispetto agli altri set, riflettono la minore complessità delle query eseguite su traffico privo di pattern di attacco.

La modalità generica (49_generica) mostra una latenza LLM mediana superiore (41,02 s vs 35–36 s) perché tutti gli scenari partono da cat_e, senza un'ipotesi di lavoro predefinita. L'assenza di un'ipotesi di indagine iniziale per categoria richiede al modello più turni di esplorazione autonoma per identificare e confermare il pattern di attacco, con un conseguente aumento del tempo di inferenza.

Due outlier (fino a 218–265 s) mostrano che il database può diventare collo di bottiglia in caso di volumi elevati (> 100.000 flussi): la scelta subottimale del piano di esecuzione è confermata dal contro-esempio GEN_CAT_C_0010 (stessa finestra di 15 minuti, volume $12\times$ inferiore, latenza SQL $25\times$ minore). Un deployment in produzione dovrebbe prevedere soglie di volume oltre le quali attivare strategie di campionamento, pre-aggregazione o delega a un motore analitico dedicato.

7.7.4 Sintesi

L'analisi dei risultati permette di trarre quattro conclusioni sul comportamento del sistema.

1. **Contributo dell'LLM.** Il contributo incrementale del modello non è di accuratezza — l'euristica è già competitiva — ma di correzione incrociata degli errori (FP euristici corretti dall'LLM, FN dell'LLM corretti dalla post-euristica) e di qualità dell'analisi.
2. **Grounding by-design.** L'architettura MCP garantisce che ogni affermazione del modello sia ancorata a record SQL verificabili. Discusso in §7.6.7.
3. **Comportamento su segnale debole.** Due regimi distinti (conservativo su BEA-CONING_C2, aggressivo su WEB_ATTACK_EXPLOIT): il primo riduce i FP ma produce FN in modalità generica, il secondo riduce i FN ma produce FP sistemici (tipo C). Discusso in §7.7.2.
4. **Stabilità complessiva.** A livello aggregato, il sistema mostra una variabilità contenuta: sull'holdout set, eseguito 3 volte in modalità guidata, l'Accuracy ha una deviazione standard di circa 1,2 punti percentuali tra una run e l'altra (§7.6.3), e la coerenza Stage 1 $\rightarrow$ Stage 2 è al 99,02% (§7.6.7). Il non-determinismo si manifesta quindi a livello di singolo scenario, non a livello di sistema.

8. Risultati e discussione

8.1 Interpretazione dei risultati

I risultati presentati nel Capitolo 7 permettono di formulare tre osservazioni sul contributo dell'architettura proposta.

Primo: l'integrazione funziona meglio dei singoli componenti. Il sistema completo (LLM + post-euristica) corregge 3 falsi positivi dell'euristica e 1 falso negativo dell'LLM sul test set. Questo significa che il beneficio non nasce dal modello o dall'euristica presi singolarmente, ma dalla loro collaborazione: ciascuno compensa i limiti dell'altro.

Secondo: il valore dell'LLM non è nell'accuratezza. Sul test set, l'euristica da sola è già molto competitiva, e l'LLM conferma il suo verdetto nel 97,41% dei casi. Il contributo reale del modello sta altrove: (i) corregge falsi positivi sistematici, (ii) produce una motivazione tecnica verificabile e priva di allucinazioni, (iii) seleziona autonomamente i tool MCP senza conoscere lo schema del database. Sono questi i punti in cui l'LLM aggiunge qualcosa che l'euristica non può dare.

Terzo: il sistema non è deterministico, e va accettato come tale. La variabilità tra esecuzioni — documentata in §7.7 (tipo D) — mostra che su scenari ambigui lo stesso input può produrre verdetti diversi. Questo non è un difetto da nascondere, ma una caratteristica da gestire: il sistema va interpretato come uno strumento di supporto decisionale, non come un classificatore infallibile. Le metriche aggregate vanno lette come risultati osservati in una specifica campagna, non come valori assoluti.

8.2 Vantaggi dell'approccio

Nonostante i limiti discussi, l'architettura basata su MCP offre vantaggi concreti rispetto ai sistemi tradizionali impiegati nel SOC.

- **Grounding delle evidenze:** L'LLM non accede mai direttamente al database: vede solo i risultati dei tool MCP, prodotti da query SQL deterministiche. Ogni afferma-

zione del modello è quindi ancorata a record verificabili. L’analisi manuale su 15 log non ha rilevato nessuna allucinazione.

- **Contestualizzazione semantica:** L’LLM interpreta la natura dei servizi e la direzione delle connessioni, riconoscendo pattern che sfuggono alle regole statiche. Ad esempio, distingue un heartbeat legittimo da un beacon C2, cosa che l’euristica a soglie non può fare.
- **Interfaccia trasparente e auditabile:** Ogni chiamata ai tool e ogni motivazione del report finale sono tracciate. Questo rende il sistema verificabile e affidabile in un contesto operativo.
- **Correzione incrociata:** L’integrazione LLM + euristica corregge gli errori sistematici di entrambi, risultando più robusta di ciascun componente preso singolarmente.

8.3 Confronto con approcci tradizionali

La Tabella 8.1 confronta l’approccio proposto con i sistemi tradizionali.

Dimensione	IDS/SIEM	ML Classico	LLM + MCP (Proposto)
Gestione traffico misto	Elevati FP se soglie basse	Sensibile a Data Drift	Elevata contestualizzazione (1 FP su 193)
Attacchi sub-volumetrici L7	Spesso non rilevati	Rilevati se addestrati	Rilevati via analisi contestuale L7 e pattern temporali (1 FN su 193)
Resistenza a Jittering	Fallimento con CV > 0.6	Richiede retraining	Identifica Jitter se guidato (59/60 BEACONING rilevati)
Trasparenza	Assente	Scarsa (SHAP)	Altissima (Report con evidenze)

Tabella 8.1: Confronto con approcci tradizionali.

Nota: per la dimensione “Resistenza a Jittering”, il rapporto indica il numero di scenari BEACONING_C2 classificati correttamente sul totale dei 60 scenari di quella categoria presenti nel test set.

8.4 Limiti e implicazioni

I limiti del sistema sono discussi in dettaglio nel §9.1 e sintetizzati nella Tabella 9.1. Qui si richiamano solo le implicazioni applicative che ne derivano.

Implicazioni per l'uso operativo. In un contesto operativo reale, dove un falso negativo è tipicamente più grave di un falso positivo, la direzione di miglioramento non è una ricalibrazione uniforme delle soglie, ma tre interventi distinti: (i) riduzione della varianza attraverso un voto multi-run o un meccanismo di tie-breaking più forte (tipo D, §7.7); (ii) correzione del criterio euristico di rilevamento del Web Brute Force, che attualmente combina una soglia euristica e una regola tassativa del prompt basate sulla sola concentrazione di richieste su pochi target, in modo da declassare il polling legittimo (tipi C di §7.7), e contestuale revisione della soglia per gli attacchi Web a basso volume (tipo E di §7.7); (iii) mantenimento del trade-off a favore del recall sulle categorie di attacco, che resta una scelta di design e non un difetto da correggere.

Il sistema implementa guardrail simmetrici: da un lato whitelist e deroga documentata contro i falsi positivi, dall'altro DIVIETO_BENIGN_CON_ANOMALIE e blocco anti-falso-negativo lato client, che impediscono di chiudere con BENIGN in presenza di anomalie strutturate non spiegate. L'effetto netto sui dati è una distribuzione degli errori in parte strutturale e in parte probabilistica: sul test set 1 FP e 1 FN, con l'FN che scompare alla riesecuzione. La Precision e il Recall vanno quindi letti come risultati di un sistema che non è deterministico, e non come valori assoluti e definitivi. Il vero fenomeno da indirizzare in un deployment SOC non è la ricalibrazione di Precision/Recall, ma il non-determinismo residuo. In un contesto operativo reale, inoltre, la soglia di emissione di un verdetto di attacco andrebbe calibrata in modo mirato per categoria, non in modo uniforme: su BEACONING.C2 per ridurre i falsi negativi in modalità generica (§7.7, tipo B), su WEB_ATTACK_EXPLOIT per distinguere il polling legittimo dagli attacchi a basso volume (§7.7, tipi C ed E).

Implicazioni per la latenza. La latenza mediana di $\approx 49\text{--}58$ s è compatibile con un uso near-real-time su finestre di 15 minuti, ma non con un uso interattivo su singoli flussi. Inoltre, due scenari outlier (fino a 264 s) mostrano che il database può diventare collo di bottiglia in caso di volumi elevati (> 100.000 flussi). Un deployment in produzione dovrebbe prevedere soglie di volume oltre le quali attivare strategie di campionamento o pre-aggregazione.

Un modello operativo per l'integrazione LLM/euristica. In letteratura, architetture

che combinano le decisioni di più stimatori per ridurre l'errore sistematico sono note come sistemi di classificazione multipla o ensemble methods [38][39]. Nel caso specifico, i due stimatori — l'euristica deterministica e l'LLM — operano sulle stesse evidenze ma con logiche differenti (rispettivamente a soglia e a contestualizzazione semantica). La loro combinazione produce un effetto di mutua correzione. Si propone pertanto di denominare questo modello operativo **validatore incrociato**, intendendo con validatore un componente che non decide autonomamente ma verifica e motiva, e con incrociato la doppia direzione della correzione (LLM corregge euristica; euristica corregge LLM).

Il termine validatore sottolinea che la funzione del sistema non è quella di decidere, ma di verificare e motivare; l'aggettivo incrociato rimanda alla doppia direzione della validazione, documentata quantitativamente nel Capitolo 7. Da questo modello derivano tre implicazioni operative.

- **Supporto all'analista.** Sintesi automatica dei flussi correlati e riduzione guidata del rumore di fondo, per diminuire il carico cognitivo dell'analista.
- **Standardizzazione del ragionamento investigativo.** L'uso di interfacce in linguaggio naturale consente anche ad analisti non esperti di interrogare database complessi, senza conoscere SQL né lo schema delle tabelle.
- **Correzione incrociata LLM/euristica.** Nei casi osservati, il sistema integrato si è rivelato più robusto di ciascun componente preso singolarmente, e può essere impiegato per ridurre i falsi allarmi senza perdere sensibilità.

Direzioni future. Nonostante i limiti, l'architettura model-agnostic e la separazione tra logica di accesso ai dati e motore di inferenza pongono le basi per sviluppi futuri: validazione su dataset diversi, sostituzione del modello, deployment on-premise per la privacy, apprendimento continuo delle soglie, ottimizzazione delle query SQL per ridurre la latenza. Queste direzioni sono discusse in §9.3.

9. Conclusioni e sviluppi futuri

9.1 Limiti del sistema

9.1.1 Un solo modello LLM e validazione esterna limitata

La validazione si è concentrata su Qwen 3.8 27B, l'unica configurazione utilizzabile per completare l'analisi degli scenari: l'alternativa GPT-OSS 120B via Groq è stata scartata per i limiti di token del tier gratuito, che rendevano impraticabile qualsiasi validazione (cfr. §5.3). I risultati non sono quindi generalizzabili ad altri LLM senza ulteriori esperimenti. L'architettura model-agnostic rende la sostituzione del modello un'operazione di configurazione, ma una riattivazione di GPT-OSS richiederebbe una revisione dei rami di codice specifici per quel modello prima di poter essere considerata affidabile.

Per quanto riguarda la validazione su traffico reale, il sistema è stato valutato principalmente su dataset accademico (CIC-IDS-2017). Come verifica preliminare è stato condotto un test su traffico benigno generato dall'autrice e processato con la stessa pipeline (4,5 ore di cattura, 18 finestre da 15 minuti, nessun falso positivo). Il campione è però limitato: non copre la varietà di scenari di traffico reale (VPN, streaming, backup, aggiornamenti di sistema). Inoltre, la whitelist dei servizi legittimi è stata estesa per coprire servizi moderni assenti o irriconoscibili nel dataset del 2017 (es. `claude.ai`, Teams, Google), coerentemente con quanto discusso in §7.6.5. Questa estensione è un intervento metodologico appropriato — la whitelist è parte integrante dell'euristica e va mantenuta aggiornata — ma il test su 18 finestre non consente di misurare quanto il sistema sarebbe in grado di generalizzare su destinazioni arbitrarie non incluse nella whitelist. Il limite non è quindi l'obsolescenza del dataset, ma la copertura della whitelist stessa: una valutazione su traffico reale più ampio e diversificato, con destinazioni non whitelisted, è necessaria per stimare la robustezza del sistema al di fuori dell'insieme di servizi già noti. A ciò si aggiunge la distanza temporale: il dataset è del 2017, il traffico reale del 2026, con differenze notevoli (QUIC, HTTP/3, TLS 1.3) che l'estensione della whitelist mitiga solo parzialmente.

9.1.2 Limiti intrinseci dell’architettura

Tre limiti riguardano l’architettura nel suo complesso. La Tabella 9.1 riassume l’insieme dei limiti discussi in questo capitolo, con il loro impatto e le mitigazioni adottate.

Latenza e scalabilità. La latenza mediana end-to-end è di $\approx 49\text{--}58$ s, di cui la maggior parte dovuta all’inferenza LLM (mediana $\approx 34\text{--}41$ s). Il sistema è compatibile con un uso near-real-time su finestre di 15 minuti, ma non con un uso interattivo su singoli flussi.

Dipendenza dalla qualità del dato sorgente. Il sistema si basa interamente sulle feature estratte da nDPI: se una minaccia non è rilevabile dalle feature (es. un payload malevolo in un flusso TLS classificato come traffico legittimo), il sistema non può rilevarla. Questo limite è intrinseco all’architettura: il sistema non analizza il payload, ma solo i metadati e le statistiche. L’analisi di payload completi richiederebbe tecniche di ispezione più invasive (es. decifratura TLS), con implicazioni di privacy e complessità computazionale.

Assenza di apprendimento continuo. Il sistema è stateless: il modello non “impara” dai nuovi attacchi rilevati, limitandosi a un’analisi one-shot per ogni scenario. Le soglie dell’euristica sono statiche e non si adattano automaticamente a nuovi pattern di traffico. Questo limite è in parte mitigato dalla flessibilità dell’architettura MCP (nuovi tool possono essere aggiunti senza modificare il modello), ma richiede intervento umano per l’aggiornamento delle regole.

Limite	Impatto	Mitigazione
Confronto a livello di finestra	Una finestra è positiva anche con un solo flusso malevolo	Appendice C; §7.4.2
Finestre sporche (48,35%)	Il 56,41% di esse ha rapporto malevoli/benigni ≤ 1	Appendice C: metrica “contiene attacco”
Test set calibrato	Metriche non generalizzabili	Holdout set (§7.6.3), mitigazione parziale
Categorie deboli (13–32 scenari)	Numerosità ridotta	Tre generazioni, holdout
Un solo modello LLM (Qwen)	Risultati non generalizzabili	Architettura model-agnostic
Rami di codice GPT-OSS	Non più mantenuti	Documentati in §9.1.1
Traffico reale limitato	4,5 ore, 18 finestre	Validazione su traffico reale esteso (cfr. §9.3)
Latenza $\approx 49\text{--}58$ s	Non real-time interattivo	Near-real-time su finestre 15 min
Dipendenza da nDPI	Errori a monte ereditati	Analisi payload a livello di flusso (cfr. §9.3)
Nessun apprendimento continuo	Soglie statiche	§9.3: whitelist dinamica
Non-determinismo	Verdetti diversi su stesso input	Documentato in §7.7; §9.3 (voto multi-run, sviluppo futuro)

Tabella 9.1: Sintesi dei limiti del sistema.

9.2 Contributi della tesi

Il contributo principale di questa tesi è la valutazione sperimentale di un'architettura LLM + MCP per il triage di sicurezza di rete, inteso come classificazione motivata del traffico a partire da dati strutturati estratti tramite DPI (§1.3). Nello specifico:

- **Valutazione empirica del contributo dell'LLM rispetto a un'euristica deterministica.** Su 242 scenari costruiti da CIC-IDS-2017, il lavoro isola il contributo incrementale del ragionamento del modello rispetto a un classificatore a soglie statiche, analizzando i casi in cui l'LLM corregge, conferma o disaccorda dal verdetto dell'euristica.
- **Analisi qualitativa dei tipi di errore e dei meccanismi di correzione incrociata.** Il lavoro documenta cinque tipi di disaccordo (A–E) osservati sull'intera campagna di validazione (260 scenari) e mostra come l'integrazione LLM + euristica corregga gli errori sistematici di entrambi, documentando anche i casi in cui la correzione non avviene (tipi C, D, E).

Dal punto di vista ingegneristico, il lavoro fornisce:

- **Architettura modulare e model-agnostic** per l'integrazione di LLM con dati di rete strutturati, basata su MCP.
- **Un'implementazione di riferimento** con 15 tool MCP, meccanismo di gating e isolamento strutturale del blind test.

I limiti metodologici (confronto a livello di finestra anziché di singolo flusso, calibrazione su un solo dataset, validazione su un solo modello LLM, non-determinismo residuo) sono documentati e costituiscono la base per gli sviluppi futuri.

9.3 Sviluppi futuri

Per superare i limiti riscontrati, il lavoro potrà essere esteso lungo le seguenti direttrici:

Validazione e generalizzazione. Sarà necessario validare il sistema su un holdout genuinamente indipendente (es. CSE-CIC-IDS2018) e esplorare architetture multi-agente con voting per ridurre la varianza su scenari ambigui [40][41][42].

Ottimizzazione tecnica. Si potrà sfruttare l'informazione direzionale già presente in nDPI per ridurre i falsi positivi da polling, ottimizzare le query SQL con indici composti e viste materializzate, e aggiornare dinamicamente la whitelist dei servizi tramite feed pubblici.

Deployment on-premise. La natura model-agnostic dell'architettura consente di sostituire l'endpoint API esterno con un server di inferenza locale, garantendo sovranità sui dati e riducendo la dipendenza da provider esterni.

9.4 Conclusioni e considerazioni finali

La presente tesi ha verificato la fattibilità e i limiti dell'integrazione di agenti basati su LLM e MCP in un sistema di Deep Packet Inspection per il triage di sicurezza di rete (§9.2). La valutazione sperimentale su 242 scenari costruiti da CIC-IDS-2017 ha mostrato che l'architettura raggiunge un'Accuracy del 98,96% e una Precision del 98,90% sul test set, con prestazioni stabili sull'holdout set. Il contributo specifico dell'LLM rispetto a un motore euristico deterministico non risiede nell'accuratezza complessiva — già competitiva nel classificatore a regole — ma nella correzione incrociata dei rispettivi errori sistematici, nella qualità della motivazione tecnica e nella flessibilità di interrogazione in linguaggio naturale. I limiti emersi — in particolare il non-determinismo residuo su scenari ambigui e la calibrazione su un singolo dataset — delineano le direzioni di sviluppo discusse in §9.3.

L'introduzione dei modelli linguistici nel dominio della cybersecurity apre nuove possibilità per l'analisi di rete, ma richiede rigore architetturale. L'intelligenza generativa non sostituisce una telemetria ben strutturata, ma può diventare un utile strumento di supporto al SOC se integrata con guardrails deterministici e meccanismi di escalation umana, cioè il passaggio della decisione finale a un analista nei casi in cui il verdetto automatico è ambiguo o ad alto impatto. Il sistema descritto in questo lavoro non si propone come sostituto

dell'analista umano. La sua funzione è quella di supportare il processo decisionale, riducendo il carico cognitivo associato alla correlazione manuale delle evidenze e migliorando la qualità e la tracciabilità dell'analisi.

Bibliografia

- [1] Raouf Boutaba, Mohammad A. Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano, and Oscar M. Caicedo. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9(1):16, 2018.
- [2] Richard Zuech, Taghi M. Khoshgoftaar, and Randall Wald. Intrusion detection and big heterogeneous data: a survey. *Journal of Big Data*, 2(1):3, 2015.
- [3] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE Symposium on Security and Privacy*, pages 305–316, 2010.
- [4] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Tobias Gramp, Christopher Kruegel, and Giovanni Vigna. Dos and don'ts of machine learning in computer security. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3971–3988, 2022.
- [5] Mohamed Amine Ferrag, Mthandazo Ndhlovu, Norbert Tihanyi, Lucas C. Cordeiro, Merouane Debbah, Thierry Lestable, and Narinderjit Singh Thandi. Revolutionizing cyber threat detection with large language models: A privacy-preserving bert-based lightweight model for iot/iiot devices. *IEEE Access*, 12:1–18, 2024.
- [6] Model Context Protocol. Introduction to model context protocol (mcp). <https://modelcontextprotocol.io>, 2024.
- [7] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, pages 108–116, 2018.
- [8] Luca Deri, Maurizio Martinelli, Tomasz Bujlow, and Alfredo Cardigliano. ndpi: Open-source high-speed deep packet inspection. In *2014 International Wire-*

- less Communications and Mobile Computing Conference (IWCMC)*, pages 617–622, 2014.
- [9] Fannia Pacheco. *Classification techniques for the management of the “Quality of Service” in Satellite Communication systems*. Theses, Université de Pau et des Pays de l’Adour, Nov 2019.
 - [10] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2):18–28, 2009.
 - [11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.
 - [12] Thuy T.T. Nguyen and Grenville Armitage. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys & Tutorials*, 10(4):56–76, 2008.
 - [13] Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *Proceedings of the 17th USENIX Security Symposium*, pages 139–154, 2008.
 - [14] Alberto Dainotti, Alistair King, kc Claffy, Ferdinando Papale, and Antonio Pescapè. Analysis of a ”/0” stealth scan from a botnet. In *Proceedings of the 2012 ACM Internet Measurement Conference (IMC)*, pages 1–14, 2012.
 - [15] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng. Hstids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE Access*, 6:1792–1806, 2017.
 - [16] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901, 2020.

- [17] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9459–9474, 2020.
- [18] Tarek Ali and Panos Kostakos. Huntgpt: Integrating machine learning-based anomaly detection and explainable ai with large language models (llms), 2023.
- [19] Yanjie Li, Zhen Xiang, Nathaniel D. Bastian, Dawn Song, and Bo Li. IDS-agent: An LLM agent for explainable intrusion detection in iot networks. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024.
- [20] Egil Karlsen, Xiao Luo, Nur Zincir-Heywood, and Malcolm Heywood. Benchmarking large language models for log analysis, security, and interpretation, 2023.
- [21] Lamia Chaari Fourati, Mirna Awad, Mariem Ben Ali, and Wael Jaafar. Large language models for cyberattack defense: a critical survey. *Knowledge and Information Systems*, 2026.
- [22] Sahaya Jestus Lazer, Kshitiz Aryal, Maanak Gupta, and Elisa Bertino. A survey of agentic ai and cybersecurity: Challenges, opportunities and use-case prototypes, 2026.
- [23] Minrui Xu, Jiani Fan, Xinyu Huang, Conghao Zhou, Jiawen Kang, Dusit Niyato, Shiwen Mao, Zhu Han, Xuemin Shen, and Kwok-Yan Lam. Forewarned is forearmed: A survey on large language model-based agents in autonomous cyberattacks, 2025.
- [24] Seyedakbar Mostafavi. Trustworthy agentic ai: A comprehensive cybersecurity and systems survey on threat landscapes, defense architectures, and open challenges, 2026.
- [25] Duo Wu, Xianda Wang, Yaqi Qiao, Zhi Wang, Junchen Jiang, Shuguang Cui, and Fangxin Wang. Netllm: Adapting large language models for networking. In *Pro-*

- ceedings of the ACM SIGCOMM 2024 Conference*, pages 661–678. ACM, Aug 2024.
- [26] Xuying Meng, Chungang Lin, Yequan Wang, and Yujun Zhang. Netgpt: Generative pretrained transformer for network traffic, 2023.
 - [27] Hamed Alqahtani and Gulshan Kumar. Large language models for cybersecurity intelligence: A systematic review of emerging threats, defensive capabilities, and security evaluation frameworks. *Computers, Materials & Continua*, 87(3):297–367, 2026.
 - [28] A. Biyouki, S. Lotfipour, and B. Haghi. An enhanced deep learning framework for intrusion classification in enterprise network using multi-branch cnn-attention architecture. *Scientific Reports*, 2025.
 - [29] Anna L. Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176, 2016.
 - [30] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley Longman Publishing Co., Inc., USA, 2 edition, 2003.
 - [31] Oracle Corporation. *MySQL 6.0 Reference Manual*. MySQL Press, 2010.
 - [32] Microsoft. Model context protocol (mcp) documentation. <https://learn.microsoft.com/en-us/windows/ai/mcp/>, 2024.
 - [33] Qwen Team. Qwen documentation. <https://qwen.readthedocs.io/en/latest/>, 2026. Documentazione ufficiale della famiglia di modelli Qwen, Alibaba Group.
 - [34] The Zeek Network Security Monitor. Community id flow hashing. <https://github.com/corelight/community-id-spec>. Accessed: 2026-10-06.
 - [35] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models, 2024.

- [36] Aswathnarayan Muthukrishnan Kirubakaran, Vinoth Punniyamoorthy, Adithya Parthasarathy, Akash Kumar Agarwal, Shiva Kumar Reddy Carimireddy, Nachiappan Chockalingam, and Akshay Deshpande. Operationalizing multi-agent interoperability via contract-driven model context protocols. In *SoutheastCon 2026*, pages 1–6, 2026.
- [37] PrefectHQ. FastMCP: The fast, pythonic way to build mcp servers and clients, 2024. GitHub repository.
- [38] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems. First International Workshop, MCS 2000, Cagliari, Italy, June 21–23, 2000, Proceedings*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2000.
- [39] Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons, New York, 2004.
- [40] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.
- [41] Shreshth Rajan. Multiver: Zero-shot multi-agent vulnerability detection. *arXiv preprint arXiv:2602.17875*, 02 2026.
- [42] Chia-Hong Chou, Arjun Sudheer, and Younghee Park. An llm-based agentic network traffic incident-report approach towards explainable-ai network defense. *Journal of Sensor and Actuator Networks*, 15(2):32, 2026.

A. Codice sorgente e risorse

A.1 Repository ufficiale del progetto

Al fine di garantire la riproducibilità degli esperimenti, l'intero ecosistema software è centralizzato in una repository Git pubblica: <https://github.com/AndreaNizzi/Tesi>

La repository è strutturata in moduli che rispecchiano le macro-fasi del Capitolo 6:

Directory	Contenuto	Sezione
/pipeline_data	Script di preprocessing, join, normalizzazione e bulk load verso MySQL	§6.1
/mcp_server	Implementazione dei tool e del middleware MCP	§6.2.1
/evaluation	Script di blind testing e calcolo metriche	§7

Tabella A.1: Struttura repository GitHub del progetto.

A.2 Estratti di codice significativi

Vengono riportati quattro frammenti che documentano scelte progettuali non ovvie e non ricostruibili dalla sola descrizione testuale. Il codice integrale è nella repository.

A.2.1 Community ID bidirezionale

Il `community_id` è la chiave logica che tiene insieme l'architettura multi-tabella: calcolato in modo deterministico sulla 5-tupla tramite l'implementazione standard Zeek/Corelight (SHA-1 su Base64), garantisce la bidirezionalità — lo stesso flusso osservato da `CICFlowMeter` e da `ndpiReader` produce lo stesso identificativo anche se le due sonde invertono sorgente e destinazione.

```
1 from ipaddress import ip_address
```

```

2 import communityid
3
4 CID = communityid.CommunityID()
5
6 def calc_community_id(sip, dip, sport, dport, proto):
7     try:
8         src_ip = ip_address(str(sip).strip())
9         dst_ip = ip_address(str(dip).strip())
10        proto_num = int(str(proto).strip())
11        s = int(str(sport).strip())
12        d = int(str(dport).strip())
13        if proto_num == 6:
14            tpl = communityid.FlowTuple.make_tcp(src_ip, dst_ip, s, d)
15        elif proto_num == 17:
16            tpl = communityid.FlowTuple.make_udp(src_ip, dst_ip, s, d)
17        elif proto_num == 1:
18            tpl = communityid.FlowTuple.make_icmp(src_ip, dst_ip, s, d)
19        elif proto_num == 58:
20            tpl = communityid.FlowTuple.make_icmp6(src_ip, dst_ip, s, d)
21        )
22        else:
23            return "error"
24        return CID.calc(tpl)
25    except Exception:
26        return "error"

```

Code A.1: Calcolo del Community ID.

A.2.2 Accoppiamento a tolleranza temporale

Il cuore algoritmico del recupero delle label. Invece di una de-duplicazione statica sulla chiave (che eliminerebbe interazioni legittime), il sistema esegue un `merge_asof` vincolato alla stessa `flow_key` — una chiave composita formata dal Community ID (identificativo bidirezionale della 5-tupla) e dall'IP sorgente — con tolleranza di 60 secondi. Questa tolleranza compensa il disallineamento residuo tra la pipeline locale (CICFlow-

Meter, timestamp al secondo) e i CSV ufficiali del dataset (timestamp troncati al minuto).
I record senza corrispondenza sono etichettati UNMATCHED e scartati a valle.

```

1 df_final = pd.merge_asof(
2     df_miei.sort_values("ts_key"),
3     df_uff[["flow_key", "ts_key", "duration_norm", label_col_off]]
4         .sort_values("ts_key"),
5     on="ts_key",
6     by="flow_key",
7     direction="nearest",
8     tolerance=pd.Timedelta(minutes=1),
9     suffixes=("", "_ref")
10 )
11 matched = df_final[final_label_col].notna().mean()
12 print(f"[INFO] Tasso di match: {matched:.1%}")
13 df_final["Label"] = df_final[final_label_col].fillna("UNMATCHED").
    astype(str)

```

Code A.2: Accoppiamento a tolleranza temporale.

A.2.3 Isolamento strutturale del blind test

Il punto architetturale centrale del framework di validazione: tutti i 15 tool espongono query confinate a ndpi_flows, nativamente priva della colonna label (confinata in cic_flows). Il confronto con il ground truth avviene solo ex-post, in fase di auditing.

```

1 @mcp.tool(description=prompts.DESC_GET_HOST_PORT_DISTRIBUTION)
2 def get_host_port_distribution(ip_target: str, start_time: str,
3     end_time: str) -> str:
4     query = text("""
5         SELECT dst_port, COUNT(*) as num_flussi, ...
6         FROM ndpi_flows      -- <- unica tabella accessibile
7         WHERE (src_ip = :ip OR dst_ip = :ip)
8             AND timestamp_start BETWEEN :start_time AND :end_time
9             GROUP BY dst_port ORDER BY num_flussi DESC;
10    """)

```

```
10 ...
```

Code A.3: Isolamento strutturale del blind test.

A.2.4 Tie-breaking e delega all'LLM

Quando più categorie raggiungono lo stesso punteggio massimo, il tool applica un tie-breaking basato sulla forza relativa delle evidenze; se nessun criterio dirime, restituisce CONFLITTO_IRRISOLTO e delega la decisione all'LLM.

```
1 elif set(vincitori) == {"SCAN.BRUTEFORCE", "DOS.VOLUMETRIC"}:  
2     pps_eff = max(float(mk.get("pps_aggregati") or 0), float(mk.get("burst_pps") or 0))  
3     web_rps_eff = max(float(mk.get("web_rps") or 0), float(mk.get("burst_web_rps") or 0))  
4     flussi_web_val = int(mk.get("flussi_web_totali") or 0)  
5  
6     margine_dos = max(  
7         pps_eff / max(float(s.DOS_PPS_MIN), 1.0),  
8         web_rps_eff / max(float(getattr(s, "DOS_L7_RPS_MIN", 10.0)),  
9         flussi_web_val / max(float(getattr(s, "  
10         DOS_L7_FLUSSI_ASSOLUTI_MIN", 1000)), 1.0),  
11         1.0),  
12         margine_scan = porte_uscita_target / max(int(s.SCAN_PORTE_MIN), 1)  
13         margine_bruteforce = (  
14             max_tentativi_bf / max(float(getattr(s, "  
15             BRUTEFORCE_TENTATIVI_MIN", 25)), 1.0)  
16             if bruteforce_l4_confirmed else 0.0  
17         )  
18         margine_scan_effettivo = max(margine_scan, margine_bruteforce)  
19  
20         vincitore_assoluto = "DOS.VOLUMETRIC" if margine_dos >=  
21         margine_scan_effettivo else "SCAN.BRUTEFORCE"  
22         note_logic.append(  
23             f"Pareggio SCAN/DOS risolto a favore di {vincitore_assoluto}"
```

```

21         f"(margin: dos={margine_dos:.2f}, scan={margine_scan_effettivo
22         :.2f})"
23         f"[porte_uscita_reali_target={porte_uscita_target}, "
24         f"max_tentativi_bf={max_tentativi_bf}], "
25         f"bruteforce_l4_confirmed={bruteforce_l4_confirmed})."
26     )
27 else:
28     vincitore_assoluto = "CONFLITTO_IRRISOLTO"
29     conflitto_a_pari_merito = vincitori

```

Code A.4: Tie-breaking e delega all'LLM.

A.3 Schema del database

Il DDL completo delle tabelle `ndpi_flows`, `cic_flows`, `flow_mapping` e `report_validazione`, con tipi effettivi e indici. La separazione fisica delle tabelle è il fondamento del blind test: `ndpi_flows` è l'unica sorgente accessibile all'LLM, mentre `cic_flows` contiene la ground truth (label) ed è inaccessibile all'agente. `flow_mapping` è la tabella di allineamento tra le due tabelle (Appendice B), e `report_validazione` memorizza la qualificazione degli scenari del benchmark (Appendice C.3).

```

1  -- Tabella di input per l'LLM: nativamente priva di colonna 'label'
2  CREATE TABLE ndpi_flows (
3      id BIGINT UNSIGNED AUTOINCREMENT PRIMARY KEY,
4      community_id VARCHAR(50) NOT NULL,
5      src_ip VARCHAR(45) NOT NULL,
6      dst_ip VARCHAR(45) NOT NULL,
7      src_port INT UNSIGNED NOT NULL,
8      dst_port INT UNSIGNED NOT NULL,
9      protocol INT UNSIGNED NOT NULL,
10     timestamp_start TIMESTAMP(6) NOT NULL,
11     duration_ms DOUBLE NOT NULL,
12     total_bytes BIGINT UNSIGNED NOT NULL,
13     fwd_packets BIGINT UNSIGNED NOT NULL,

```

```

14     bwd_packets BIGINT UNSIGNED NOT NULL,
15     total_fwd_bytes BIGINT UNSIGNED NOT NULL,
16     total_bwd_bytes BIGINT UNSIGNED NOT NULL,
17     packet_rate DOUBLE NOT NULL,
18     byte_rate DOUBLE NOT NULL,
19     iat_flow_avg DOUBLE NOT NULL,
20     iat_flow_stddev DOUBLE NOT NULL,
21     tcp_flags INT DEFAULT NULL,
22     ndpi_hostname VARCHAR(255) DEFAULT NULL,
23     payload_entropy DOUBLE DEFAULT NULL,
24     app_hierarchy VARCHAR(100) DEFAULT NULL,
25     infra_provider VARCHAR(100) DEFAULT NULL,
26     tls_version VARCHAR(10) DEFAULT NULL,
27     tls_cipher_suite VARCHAR(100) DEFAULT NULL,
28     tls_ja4 VARCHAR(36) DEFAULT NULL,
29     tls_issuer_dn VARCHAR(255) DEFAULT NULL,
30     KEY idx_timestamp (timestamp_start),
31     KEY idx_comm_time (community_id, timestamp_start),
32     KEY idx_ndpi_flows_src_time_dst (src_ip, timestamp_start, dst_ip,
33     dst_port),
34     KEY idx_flows_dst_ip_start (dst_ip, timestamp_start)
35 ) ENGINE=InnoDB;
36
37 -- Ground truth: contiene 'label', MAI esposta all'LLM
38 CREATE TABLE cic_flows (
39     id BIGINT UNSIGNED AUTOINCREMENT PRIMARY KEY,
40     community_id VARCHAR(50) NOT NULL,
41     src_ip VARCHAR(45) NOT NULL,
42     dst_ip VARCHAR(45) NOT NULL,
43     src_port INT UNSIGNED NOT NULL,
44     dst_port INT UNSIGNED NOT NULL,
45     protocol INT UNSIGNED NOT NULL,
46     timestamp_start TIMESTAMP(6) NOT NULL,
47     duration_ms DOUBLE NOT NULL,
48     total_bytes BIGINT UNSIGNED NOT NULL,
49     fwd_packets BIGINT UNSIGNED NOT NULL,

```

```

49     bwd_packets BIGINT UNSIGNED NOT NULL,
50     total_fwd_bytes BIGINT UNSIGNED NOT NULL,
51     total_bwd_bytes BIGINT UNSIGNED NOT NULL,
52     packet_rate DOUBLE NOT NULL,
53     byte_rate DOUBLE NOT NULL,
54     iat_flow_avg DOUBLE NOT NULL,
55     iat_flow_stddev DOUBLE NOT NULL,
56     label VARCHAR(50) DEFAULT 'BENIGN',
57     KEY idx_cic_timestamp (timestamp_start),
58     KEY idx_cic_label (label),
59     KEY idx_comm_time (community_id, timestamp_start),
60     KEY idx_cic_src_ip (src_ip, timestamp_start)
61 ) ENGINE=InnoDB;
62
63 -- Tabella di allineamento ndpi<=>cic (usata per auditing e terza
64    generazione)
65 CREATE TABLE flow_mapping (
66     community_id VARCHAR(128) NOT NULL,
67     ndpi_ts DATETIME(6) NOT NULL,
68     cic_ts DATETIME(6) NOT NULL,
69     label VARCHAR(64) NOT NULL,
70     drift_sec INT NOT NULL,
71     PRIMARY KEY (community_id, ndpi_ts),
72     KEY idx_fm_ndpi_ts (ndpi_ts),
73     KEY idx_fm_label (label),
74     KEY idx_fm_label_ts (label, ndpi_ts)
75 ) ENGINE=InnoDB;
76
77 -- Qualificazione degli scenari esistenti
78 CREATE TABLE report_validazione (
79     id VARCHAR(64) NOT NULL PRIMARY KEY,
80     categoria VARCHAR(16) DEFAULT NULL,
81     ip_target VARCHAR(64) DEFAULT NULL,
82     verdetto_atteso VARCHAR(64) DEFAULT NULL,
83     n_flussi INT DEFAULT NULL,
84     n_attacco INT DEFAULT NULL,

```

```

84     n_altra INT DEFAULT NULL,
85     purezza_pct DECIMAL(5,2) DEFAULT NULL,
86     soglia INT DEFAULT NULL,
87     labels_presenti TEXT,
88     KEY idx_purezza (purezza_pct),
89     KEY idx_n_attacco (n_attacco)
90 ) ENGINE=InnoDB ;

```

Code A.5: Schema del database.

Nota sugli indici. La scelta degli indici non è accessoria: le chiavi composite (community_id, timestamp_start) e (src_ip, timestamp_start, dst_ip, dst_port) coprono rispettivamente le query di drill-down via Community ID (analizza_connessione_by_community_id) e le query aggregate per host target (get_rate_statistics, get_flow_features, get_host_port_distribution). L'indice idx_fm_label_ts copre le query della terza generazione, che filtrano per label e finestra.

B. Allineamento tra ndpi_flows e cic_flows

Questa appendice integra §7.3 e Appendice C.4 con i dettagli tecnici della tabella flow_mapping, la tabella di allineamento introdotta per risolvere il disallineamento temporale tra ndpi_flows e cic_flows. Lo schema completo delle tabelle è riportato in Appendice A; qui si documentano la logica di popolamento e le verifiche empiriche.

B.1 Logica di popolamento

Per ogni coppia (community_id, ndpi_ts) si sceglie la riga di cic_flows con timestamp più vicino entro una finestra di ± 5 minuti, scartando i match con drift superiore a 60 secondi. La selezione del match minimo avviene tramite ROW_NUMBER() OVER (PARTITION BY community_id, ndpi_ts ORDER BY drift_sec), che elegge a rn = 1 il candidato con drift minore. La query è eseguita per giorno per evitare timeout su tabelle di grandi dimensioni.

```
1 INSERT INTO flow_mapping (community_id, ndpi_ts, cic_ts, label,
2   drift_sec)
3 WITH ranked AS (
4   SELECT
5     n.community_id,
6     n.timestamp_start AS ndpi_ts,
7     c.timestamp_start AS cic_ts,
8     c.label,
9     ABS(TIMESTAMPDIFF(SECOND, n.timestamp_start, c.timestamp_start))
10  ) AS drift_sec,
11  ROW_NUMBER() OVER (
12    PARTITION BY n.community_id, n.timestamp_start
13    ORDER BY ABS(TIMESTAMPDIFF(SECOND, n.timestamp_start, c.
14    timestamp_start))
15  ) AS rn
16 FROM ndpi_flows n
```

```

14 JOIN cic_flows c
15 ON n.community_id = c.community_id
16 AND c.timestamp_start BETWEEN
17 DATE_SUB(n.timestamp_start , INTERVAL 5 MINUTE)
18 AND DATE_ADD(n.timestamp_start , INTERVAL 5 MINUTE)
19 WHERE n.timestamp_start >= :giorno_inizio
20 AND n.timestamp_start < :giorno_fine
21 )
22 SELECT community_id , ndpi_ts , cic_ts , label , drift_sec
23 FROM ranked
24 WHERE rn = 1 AND drift_sec <= 60;

```

Code B.1: Logica di popolamento flow mapping.

La finestra BETWEEN ... AND ... esclude i falsi match a giorni di distanza; il filtro drift_sec ≤ 60 scarta gli accoppiamenti dubbi. Il matching risultante è uno a uno: ogni riga di ndpi_flows corrisponde ad al più una riga di cic_flows.

B.2 Verifica empirica del drift

Come discusso in §6.3.1, CICFlowMeter e nDPI registrano i timestamp dei flussi con criteri diversi: il primo usa il fuso locale della macchina di cattura, il secondo UTC. Senza correzione, i medesimi flussi nelle due tabelle risulterebbero sfasati di alcune ore (5 per CIC, 3 per nDPI), e l'accoppiamento tra cic_flows e ndpi_flows — necessario per associare ogni flusso alla sua label di ground truth — risulterebbe impossibile. La pipeline di ingestion applica quindi una compensazione esplicita degli offset in fase di caricamento (Popola_DB_CSV_Ufficiali.py sottrae 5 ore, Popola_DB_json-ndpiReader.py ne sottrae 3), riportando entrambe le sorgenti alla stessa finestra oraria di riferimento. A questo disallineamento di fuso se ne aggiunge un secondo, di natura diversa e non correggibile con un offset: i due strumenti adottano criteri di terminazione del flusso differenti. CICFlowMeter chiude un flusso quando la sua durata supera la soglia di timeout (120 secondi), frammentando una singola sessione in più record; nDPI aggrega invece l'intera sessione in un unico flusso. Ne consegue che cic_flows e ndpi_flows non

hanno una corrispondenza 1:1, e la correlazione tra le due tabelle è affidata alla tabella `flow_mapping`, che accoppia i flussi tramite `community_id` e `merge_asof` con tolleranza di 60 secondi.

La tabella `flow_mapping` — popolata con la query di Listato B.1, che applica un `merge_asof` con tolleranza di 60 secondi sulla coppia (`community_id`, `timestamp`) — contiene 1.341.850 righe. I valori di drift tra le due sorgenti conferma l'efficacia del riallineamento: tutte le righe hanno `drift_sec = 0`, cioè i `timestamp_start` delle due tabelle coincidono al secondo per ogni coppia (`community_id`, `ndpi_ts`). Nessun match presenta uno scostamento residuo; la tolleranza di 60 secondi nel `merge_asof` non è quindi risultata necessaria sul dataset corrente, ed è stata mantenuta come margine di sicurezza per eventuali disallineamenti non perfettamente compensati in pipeline diverse.

In sintesi, il riallineamento ha funzionato: la correlazione tra le due tabelle è esatta, il dataset è temporalmente pulito, e l'accoppiamento tra `cic_flows` e `ndpi_flows` — su cui si basa l'intero processo di generazione del benchmark (Appendice C.2) — è affidabile.

B.3 Query di verifica dell'allineamento

Per verificare che gli scenari del benchmark contengano effettivamente il traffico d'attacco atteso (Appendice C.4), per ogni finestra [`start.time`, `end.time`] si esegue una query che, tramite JOIN su `flow_mapping`, restituisce la distribuzione delle label CIC per lo scenario:

```
1 SELECT
2     f.label ,
3     COUNT(*) AS cnt
4 FROM ndpi_flows n
5 INNER JOIN flow_mapping f
6     ON f.community_id = n.community_id
7     AND f.ndpi_ts = n.timestamp_start
8 WHERE (n.src_ip = :ip OR n.dst_ip = :ip)
9     AND n.timestamp_start >= :start_time
```

```
10 AND n.timestamp_start < :end_time  
11 GROUP BY f.label  
12 ORDER BY cnt DESC;
```

Code B.2: Query di verifica dell'allineamento.

Da questa distribuzione si calcolano le due metriche introdotte in C.4: allineamento stretto (label dominante = verdetto atteso) e contiene attacco (numero di flussi che mappano al verdetto atteso $\geq$ soglia di qualificazione). Un esempio di fallimento di questa verifica è il caso VAL_CAT_A_021 discusso in §6.3.1. La query restituisce 0 righe, e lo scenario è stato escluso dalla validazione. I risultati aggregati sono discussi in C.4.

C. Costruzione del benchmark e dettagli metodologici

Questa appendice integra §7.3 con i dettagli tecnici della pipeline di generazione degli scenari, delle regole di labeling e della composizione del benchmark. Le tabelle di distribuzione per regime e purezza sono riportate in Appendice C.5.

C.1 Correzione della pipeline di ingestion e scarto degli UNMATCHED

Correzione della pipeline di ingestion. Prima di presentare la composizione dei set, è necessario un chiarimento su un intervento correttivo che ha preceduto l'adozione delle tre configurazioni. Nella prima versione della pipeline di ingestion, i flussi che non trovavano corrispondenza nella tabella ufficiale `cic_flows` (oggi etichettati come UNMATCHED) venivano inseriti nel database con etichetta BENIGN anziché essere scartati. Questo ha compromesso la ground truth degli scenari generati: finestre che contenevano in realtà traffico malevolo risultavano parzialmente o interamente benigne, perché i flussi UNMATCHED — mappati erroneamente a BENIGN — aumentavano artificialmente il conteggio dei flussi benigni, diluendo la quota di flussi malevoli. La pipeline è stata corretta introducendo un filtro esplicito sugli UNMATCHED, che ora vengono scartati prima del caricamento in `cic_flows`.

Criterio di scarto. Gli scenari scartati sono quelli in cui il disallineamento tra i timestamp di `cic_flows` e `ndpi_flows` — dovuto al diverso meccanismo di timeout dei rispettivi tool di flow extraction (CICFlowMeter e nDPI) — rendeva il segnale d'attacco atteso distribuito su una finestra temporale diversa da quella dichiarata. In questi casi l'attacco risultava di fatto non rilevabile nella finestra dello scenario, e la ground truth dichiarata non era più congruente con il traffico effettivamente presente in `ndpi_flow-`

ws. La verifica è stata condotta manualmente, applicando in modo uniforme il criterio di congruenza temporale: uno scenario è mantenuto solo se la finestra di `ndpi_flows` contiene almeno la soglia di qualificazione della sua categoria (Appendice C.3). Il criterio ha portato allo scarto manuale di 8 scenari della prima generazione e 9 della seconda. Un esempio rappresentativo di scenario scartato per questo motivo è `VAL_CAT_A_021` (§6.3.1): la finestra contiene 7 record Heartbleed in `cic_flows`, ma nessun flusso corrispondente in `ndpi_flows`, perché nDPI aggrega la sessione in un unico flusso che inizia prima dell'inizio della finestra.

C.2 Pipeline di generazione e regole di labeling

La costruzione del benchmark è automatizzata dallo script `genera_scenari_da_flow_mapping.py`, che opera sulla tabella `flow_mapping` — la tabella allineata tra `ndpi_flows` e `cic_flows` introdotta in Appendice B per risolvere il disallineamento temporale tra le due tabelle (cfr. §5.2.1). Lo script implementa una procedura di generazione completamente guidata dai dati.

Slotting temporale. I flussi sono raggruppati per (label CIC, ip_target, finestra da 900 secondi): ogni finestra rappresenta un intervallo di 15 minuti per un singolo host destinazione, coerentemente con la granularità di analisi del sistema. La scelta di 15 minuti è un compromesso tra la necessità di catturare pattern d'attacco completi (che possono durare da pochi secondi a diversi minuti) e quella di evitare finestre troppo lunghe che diluirebbero il segnale malevolo in traffico benigno di background. In fase di analisi, il sistema applica una tolleranza di 2 minuti in anticipo sullo `start_time`, portando la finestra effettivamente analizzata a 17 minuti: questa tolleranza simula il comportamento di un SIEM reale, che arrotonda all'indietro per non perdere l'inizio di un pattern d'attacco che può iniziare pochi secondi prima del minuto tondo. La finestra di 15 minuti resta l'unità di riferimento per la costruzione del benchmark e per il calcolo delle metriche.

Mappatura label → categoria. La funzione di mapping traduce le etichette ufficiali del dataset CIC-IDS-2017 nelle cinque classi del sistema:

- Web Attack – Brute Force, Web Attack – XSS, Web Attack – Sql Injection, Infiltration, Heartbleed → `WEB_ATTACK_EXPLOIT`
- DDoS, DoS Hulk, DoS GoldenEye, DoS slowloris, DoS Slowhttptest → `DOS_VOLUMETRIC`
- PortScan, FTP-Patator, SSH-Patator → `SCAN_BRUTEFORCE`
- Bot → `BEACONING_C2`
- BENIGN → `BENIGN`

Una label non riconosciuta solleva un errore esplicito, per evitare default silenziosi a `BENIGN`: questa scelta garantisce che nessuno scenario venga classificato come benigno per omissione di mappatura.

Regola BENIGN. Una finestra viene classificata `BENIGN` solo se ha almeno 20 flussi totali e l'IP non è né sorgente né destinazione di flussi malevoli nella finestra.

Regola anti-rumore. La soglia di flussi malevoli necessari per accettare uno scenario varia fra le tre generazioni ed è differenziata per categoria, perché la “forza” del segnale d’attacco varia molto da categoria a categoria: un DoS volumetrico genera migliaia di flussi in pochi secondi, mentre un Web Attack può esaurirsi in poche decine di flussi.

Output. Lo script `genera_scenari_da_flow_mapping.py` produce tre file:

- `test_scenarios_generati.json`: l’elenco degli scenari (id, categoria, IP, finestra temporale)
- `ground_truth_generati.json`: il verdetto atteso per ogni scenario (id, verdetto atteso)
- `report_scenari_generati.csv`: report dettagliato con `n_flussi_stimati`, `n_porte_dst_diverse`, `soglia_applicata`

Generazione	Script	Soglie (cat_a/b/c/d/e)	Note
1 ^a	<code>genera_scenari.py</code>	5 / 30 / 30 / 5 / 10	Restrittive; split formale TEST/VAL
2 ^a	<code>genera_scenari_v2.py</code>	1 / 1 / 1 / 1 / 10	Permissive; + margine dominanza 60% sui malevoli
3 ^a	<code>genera_scenari_da_flow_mapping.py</code>	3 / 10 / 5 / 3 / 20	Intermedie; slotting su <code>dst_ip</code>

Tabella C.1: Soglie per generazione del benchmark.

Le tre configurazioni riflettono un trade-off esplicito: la v1 massimizza la purezza a scapito della numerosità; la v2 massimizza il recupero di scenari deboli a scapito del rumore; la v3 cerca un compromesso, ed è quella usata per generare l’holdout set.

C.3 Configurazioni e composizione del benchmark

Qualificazione degli scenari esistenti. Per evitare che la terza generazione produca scenari sovrapposti a quelli già presenti, è stata introdotta la tabella `report_validazione`, che per ogni scenario di `test_scenarios` registra: `n_flussi`, `n_attacco`, `soglia`, `purezza_pct` e `labels_presenti`. Uno scenario è considerato valido quando $n_attacco \geq soglia$ e $n_flussi > 0$. Solo gli scenari validi sono usati come riferimento per l’esclusione delle sovrapposizioni nella terza generazione: un’eventuale sovrapposizione con uno scenario non valido non blocca la generazione di uno nuovo scenario.

`n_flussi` è il numero di flussi della finestra presenti in `flow_mapping`, cioè flussi ndpi con un flusso CIC corrispondente e quindi con etichetta; i flussi senza corrispondenza non sono contati.

Soglie di qualificazione degli scenari. Il campo `soglia` della tabella `report_validazione` (Appendice A) memorizza il numero minimo di flussi malevoli perché uno scenario sia considerato rappresentativo della sua categoria nella metrica “contiene attacco” (Appendice C.4). I valori sono uniformi per categoria e corrispondono a quelli della terza generazione (Tabella C.1): 3 per `cat_a`, 10 per `cat_b`, 5 per `cat_c`, 3 per `cat_d`, 20 per `cat_e`. Questa soglia è distinta sia dalle soglie del motore euristico (Tabella 6.3) sia dalle soglie di generazione delle prime due generazioni.

Composizione del test set. La prima generazione comprendeva 196 scenari, con una divisione formale in insieme di sviluppo (prefissi `TEST_*`) e insieme di validazione (prefissi `VAL_*`). A seguito di un riesame manuale, 8 scenari sono risultati non più validi, riducendo il set a 188. La seconda generazione (14 scenari) è stata prodotta per ovviare parzialmente alla contrazione delle categorie deboli; di questi però, 9 sono stati scartati per il criterio di congruenza temporale e solo 5 sono risultati validi (1 `DOS` + 4 `BENIGN`). I 188 scenari validi della prima generazione e i 5 della seconda sono stati combinati nel test set di 193 scenari, con prefissi `TEST_*`, `VAL_*` e `SCN_*`. Questo set è stato usato durante la calibrazione delle soglie e del prompt del sistema: le sue performance vanno quindi lette come accuratezza su un insieme noto, non come misura di generalizzazione.

Composizione dell’holdout set. La terza generazione ha prodotto 50 scenari candidati adottando le soglie intermedie {`cat_a`: 3, `cat_b`: 10, `cat_c`: 5, `cat_d`: 3, `cat_e`: 20} (Tabella C.1). Di questi, 1 è stato scartato manualmente (`GEN_CAT_A_0005`) per un errore nella procedura di verifica: lo scenario era valido. I 49 scenari rimanenti, con prefissi `GEN_*`, costituiscono l’holdout set.

Differenza tra `v1/v2` e `v3` nello slotting. La terza generazione differisce dalle precedenti anche per la scelta dell’host di riferimento: la `v1` e la `v2` raggruppano i flussi per `src_ip` (coerentemente con la struttura di `cic_flows`, dove l’etichetta è associata all’host che origina il traffico), mentre la `v3` raggruppa per `dst_ip`. La motivazione è duplice: (a) la `v3` opera su `flow_mapping`, una tabella in cui l’etichetta è già associata al flusso dall’esterno via Community ID; (b) la `dst_ip` rappresenta più naturalmente l’unità di analisi dell’agente LLM, che indaga un host vittima e non un host sorgente. Le due scelte sono complementari ma non direttamente confrontabili: il test set tende a scenari dove il target è l’attaccante, l’holdout dove il target è la vittima. Questa differenza è dichiarata come limite metodologico in §7.4.2.

Composizione finale. Il benchmark finale comprende 242 scenari: 193 nel test set (usati durante la calibrazione delle soglie e del prompt) e 49 nell’holdout set (finestre non utilizzate durante la calibrazione).

La Tabella C.2 riporta la composizione finale del benchmark, distinta per categoria e per set di provenienza (test / holdout). La Tabella C.3 riporta la distribuzione delle label CIC originali per gli scenari d’attacco.

Criterio di mantenimento dei casi limite. Sono stati mantenuti alcuni casi in cui il segnale d’attacco è presente ma debole o non dominante, per osservare il comportamento del sistema in condizioni di ambiguità. Questa scelta è intenzionale: un benchmark che escluda tutti i casi ambigui non misura la robustezza del sistema. Lo scarto per disallineamento temporale (Appendice C.1) ha riguardato gli scenari in cui il segnale d’attacco risultava non rilevabile nella finestra d’analisi.

I limiti metodologici del benchmark (test set calibrato, holdout non indipendente, numerosità delle categorie deboli) sono discussi in §7.4.2.

Categoria	Test set (scenari / flussi)	Holdout (scenari / flussi)	Totale (scenari / flussi)
WEB_ATTACK_EXPLOIT	6/2.002	7/2.784	13/4.786
DOS_VOLUMETRIC	9/60.397	9/68.289	18/128.686
SCAN_BRUTEFORCE	16/166.290	16/170.352	32/336.642
BEACONING_C2	60/30.356	7/441	67/30.797
BENIGN	102/24.768	10/3.589	112/28.357
Totale	193/283.813	49/245.455	242/529.268

Tabella C.2: Composizione del benchmark (242 scenari).

Categoria	Label CIC originale	N. scenari
WEB_ATTACK_EXPLOIT	Web Attack – Brute Force	7
	Web Attack – XSS	3
	Web Attack – Sql Injection	2
	Infiltration	1
DOS_VOLUMETRIC	DDoS	10
	DoS Hulk	6
	DoS GoldenEye	1
	DoS slowloris	1
SCAN_BRUTEFORCE	PortScan	20
	FTP-Patator	6
	SSH-Patator	6
BEACONING_C2	Bot	67
BENIGN	BENIGN	112

Tabella C.3: Distribuzione label CIC per categoria (242 scenari).

C.4 Verifica dell’allineamento ndpi/cic

Poiché l’agente LLM opera esclusivamente sulla tabella `ndpi_flows`, mentre la ground truth è definita in `cic_flows`, è stato necessario verificare che i dati a cui l’agente ha accesso contengano effettivamente il traffico etichettato come attacco. Per ogni scenario del benchmark (242 totali), sono stati confrontati i flussi di `ndpi_flows` con quelli di `cic_flows`, utilizzando la tabella `flow_mapping` come chiave di correlazione.

Metriche di allineamento. Sono state definite due metriche complementari:

- **Allineamento stretto:** la label dominante tra i flussi mappati coincide con il verdetto atteso. Questa metrica è appropriata per scenari in cui il traffico malevolo domina la finestra, ma non per quelli in cui il segnale è sporadico.

- **Contiene attacco:** il numero di flussi la cui label mappa al verdetto atteso è almeno pari alla soglia di qualificazione della categoria. Questa metrica misura la presenza del segnale d’attacco, indipendentemente dalla sua dominanza numerica.

La soglia di qualificazione utilizzata in questa metrica è la stessa memorizzata nel campo `soglia di report_validazione` (Appendice C.3), garantendo coerenza tra la generazione degli scenari e la loro validazione.

Risultati globali. La Tabella C.4 riporta i risultati aggregati.

Categoria	Allineato stretto	Contiene attacco	Totale
WEB_ATTACK_EXPLOIT	13 (100%)	13 (100%)	13
DOS_VOLUMETRIC	15 (83,3%)	17 (94,4%)	18
SCAN_BRUTEFORCE	29 (90,6%)	32 (100%)	32
BEACONING_C2	12 (17,9%)	67 (100%)	67
BENIGN	112 (100%)	112 (100%)	112
Totale	181 (74,8%)	241 (99,6%)	242

Tabella C.4: Allineamento ndpi/cic per categoria.

Tra i 18 scenari `DOS_VOLUMETRIC`, 17 contengono il segnale d’attacco in misura sufficiente a superare la soglia di qualificazione (≥ 10 flussi DoS, Appendice C.1), mentre un solo scenario — `SCN_CAT_B_3` — non lo contiene: nella sua finestra, sono presenti 208 flussi `BENIGN` e solo 2 flussi DoS `slowloris` (conteggio in `flow_mapping`). Di conseguenza:

- Allineamento stretto su `DOS_VOLUMETRIC`: 15/18 (83,3%). I 3 scenari mancanti sono `SCN_CAT_B_3` (solo 2 flussi DoS) e altri 2 in cui i flussi DoS sono presenti ma non dominanti (es. `TEST_CAT_B_067` con 25 flussi DoS su 72 totali, `GEN_CAT_B_0005` con 25 DoS su 232).
- Contiene attacco su `DOS_VOLUMETRIC`: 17/18 (94,4%). Manca solo `SCN_CAT_B_3`, l’unico scenario in cui il segnale è sotto la soglia minima di rilevabilità.

Questo scenario non è stato rimosso, poiché rappresenta una condizione realistica in cui la finestra temporale cattura solo una frazione marginale dell'attacco (2 flussi su una sessione slowloris più estesa): il segnale è sotto la soglia nella finestra, non per debolezza intrinseca dell'attacco. Il suo mantenimento impedisce che le metriche aggregate siano artificialmente ottimistiche.

I risultati mostrano che:

- Il 99,6% degli scenari (241/242) contiene effettivamente il traffico d'attacco atteso, con un numero di flussi malevoli pari o superiore alla soglia di categoria. Questo conferma che la pipeline di ingestion ndpi non perde il segnale d'attacco in modo sistematico.
- Nessuno scenario è privo di traffico in `ndpi_flows` e nessun flusso è privo di match in `flow_mapping`. Questo indica che il matching tra le due tabelle è completo per il sottoinsieme di flussi su cui l'agente opera.
- Tutti i 112 scenari BENIGN hanno label BENIGN, confermando che la ground truth benigna è corretta e che non ci sono casi di BENIGN sporcati da traffico malevolo.

Il caso del beaconing. La categoria BEACONING_C2 presenta un allineamento stretto molto basso (17,9%) ma un contenimento dell'attacco del 100%. Questo riflette la natura intrinseca del beaconing: i beacon C2 sono connessioni periodiche e sporadiche (tipicamente 9–30 flussi in 15 minuti), che in una finestra con centinaia o migliaia di flussi benigni non raggiungono mai la dominanza numerica. La metrica di allineamento stretto non è quindi appropriata per questa categoria, e va letta congiuntamente alla metrica di “contiene attacco”.

Casi anomali. Scenari con un disallineamento parziale:

- SCN_CAT_B_3: presenta 208 flussi BENIGN e solo 2 flussi DoS slowloris (in `flow_mapping`), sotto la soglia di qualificazione di cat_b (10 flussi, Appendice C.3). È l'unico scenario in cui la metrica “contiene attacco” è negativa. Il segnale d'attacco è presente, ma la finestra temporale cattura solo una frazione marginale

della sessione slowloris (2 flussi su una sessione più estesa): il segnale è sotto la soglia di rilevabilità nella finestra, non per debolezza intrinseca dell’attacco.

- GEN_CAT_B_0005: presenta 207 flussi BENIGN e 25 flussi DoS Hulk (in `flow_mapping`), per un totale di 232 flussi mappati. Il segnale d’attacco è presente (25 flussi DoS, sopra la soglia di qualificazione di `cat_b`) ma non dominante ($25/232 = 10,78\%$ di purezza). Questo scenario è rappresentativo del caso in cui il traffico d’attacco è “immerso” in traffico benigno, e per questo rientra tra i casi di regime “misto” (rapporto malevoli/benigni < 1). È stato mantenuto nel benchmark perché rappresenta una condizione realistica e perché l’attacco è comunque presente in misura sufficiente per essere rilevato.

C.5 Distribuzione delle finestre per regime e purezza

Una verifica diretta su `cic_flows` ha mostrato che 117 delle 242 finestre del benchmark (48,35%) sono “sporche”, ovvero contenenti sia flussi BENIGN sia flussi malevoli (Tabella C.5). Solo 13 finestre (5,37%) sono a dominanza d’attacco puro; la maggior parte delle finestre sporche (66 su 117, pari al 56,41%) ha un rapporto malevoli/BENIGN ≤ 1 , e in 10 casi il segnale malevolo è sotto l’1% dei flussi totali. La Tabella C.6 integra questa lettura con la distribuzione della purezza: 61 scenari su 242 (25,21%) hanno una purezza inferiore al 50%, cioè la categoria attesa è in minoranza numerica nella finestra d’analisi. Le due tabelle misurano aspetti distinti — la “sporcizia” riguarda la compresenza di benigno e malevolo, la purezza riguarda la quota della categoria attesa sul totale dei flussi mappati — e vanno lette congiuntamente.

Regime	Definizione	N. finestre	% sul totale
Solo benign	$n_m = 0$	112	46,28%
Solo attacco	$n_b = 0, n_m > 0$	13	5,37%
Attacco dominante	$n_m/n_b > 1$	51	21,07%
Misto	$0,01 \leq n_m/n_b \leq 1$	56	23,14%
Segnale debole	$n_m/n_b < 0,01$	10	4,13%
Sporche (tot.)		117	48,35%
Pulite (solo benign)		112	46,28%
Totale		242	100%

Tabella C.5: Distribuzione delle finestre per regime (242 scenari).

Le metriche di §7.6.1 vanno quindi lette con la consapevolezza che il compito è strutturalmente più difficile di quanto l'Accuracy suggerisca.

Purezza dello scenario e “sporczia” della finestra. La Tabella C.5 classifica le finestre in base al rapporto tra flussi malevoli e benigni, ma non dice quanto la categoria attesa sia rappresentata all'interno del totale. Per misurare questo aspetto, la tabella `report-validation` riporta, per ogni scenario, la purezza (percentuale di flussi che mappano al verdetto atteso sul totale dei flussi mappati).

Fascia di purezza	N. scenari	%
Ottimi ($\geq 90\%$)	160	66,12%
Buoni (80–90%)	12	4,96%
Scarsi (50–80%)	9	3,72%
Pessimi ($< 50\%$)	61	25,21%
Totale	242	100%

Tabella C.6: Distribuzione della purezza sui 242 scenari.

Un quarto degli scenari (61 su 242) ha una purezza inferiore al 50%, cioè la categoria attesa è minoranza numerica nella finestra.

Il caso specifico del beaconing C2. La maggior parte dei 61 scenari a bassa purezza appartiene alla categoria BEACONING_C2. La ragione è intrinseca alla natura del beaconing: un impianto C2 reale genera tipicamente 9–30 flussi periodici in 15 minuti, che in una finestra con centinaia o migliaia di flussi BENIGN (browsing, telemetria, aggiornamenti) non raggiungono mai la dominanza numerica. In altri termini, una finestra BEACONING_C2 con purezza $< 50\%$ non è una ground truth “sporca”: è il riflesso di un segnale sporadico in un contesto di traffico di fondo.

Distribuzione per categoria. La Tabella C.7 riporta la distribuzione dei regimi per categoria di indagine.

Categoria	Solo benign	Solo attacco	Attacco dominante	Misto	Segnale debole	Totale
WEB_ATTACK_EXPLOIT	0	1	4	8	0	13
DOS_VOLUMETRIC	0	0	17	0	1	18
SCAN_BRUTEFORCE	0	0	29	3	0	32
BEACONING_C2	0	12	1	45	9	67
BENIGN	112	0	0	0	0	112
Totale	112	13	51	56	10	242

Tabella C.7: Distribuzione dei regimi per categoria (242 scenari).

La distribuzione mostra pattern distinti per categoria: DOS_VOLUMETRIC e SCAN_BRUTEFORCE hanno quasi tutte le finestre ad attacco dominante (17/18 e 29/32), riflettendo la natura di questi attacchi. WEB_ATTACK_EXPLOIT ha una distribuzione mista (8 misto, 4 attacco dominante, 1 solo attacco): gli attacchi Web tendono a essere concentrati su poche destinazioni ma con volumi paragonabili al traffico benigno circostante. BEACONING_C2 è fortemente sbilanciata verso misto (45/67) e segnale debole (9/67), per la natura sporadica dei beacon. BENIGN ha tutte le finestre “solo benign” per costruzione.

Implicazione per la validazione. Il confronto tra verdetto LLM e ground truth non avviene a livello di singolo flusso, ma a livello di finestra host/15 minuti: una finestra è considerata positiva anche in presenza di un solo flusso malevolo. Questo rende il confronto non direttamente comparabile con una valutazione per-flusso: il sistema non deve classificare un’istanza isolata, ma decidere se un host è sotto attacco a partire da una finestra in

cui il segnale malevolo può essere minoritario e disperso in un contesto prevalentemente benigno.

Dichiarazione sull'uso di strumenti di IA generativa

In conformità con le linee guida del Dipartimento di Informatica, si dichiara che strumenti di Intelligenza Artificiale generativa sono stati utilizzati durante la stesura di questa tesi.

Nello specifico:

- **Revisione stilistica e correzione di bozze:** Sono stati utilizzati strumenti di IA per migliorare la scorrevolezza, la chiarezza e la correttezza grammaticale del testo.
- **Generazione di codice e script:** L'IA è stata utilizzata come supporto alla scrittura e al debugging del codice Python sviluppato per la pipeline di analisi e la suite di test.

Tutti i contenuti generati o suggeriti dall'IA sono stati verificati, rielaborati e compresi criticamente dall'autrice. La responsabilità del contenuto finale di questa tesi rimane interamente a carico della sottoscritta.