# ANT: A MULTI-GRANULARITY NETWORK TRAFFIC DATASET AND BENCHMARK FOR AGENTS BEHAVIOR AUDITING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

The growing adoption of large language model (LLM) agents creates a need for network administrators and security teams to audit agent behavior within organizational networks without inspecting private user content. Network traffic offers an observable source of evidence, but how much it reveals about agent tasks and operations remains unclear. Existing traffic datasets lack the joint task and stage annotations needed to evaluate this question. We introduce ANT (**A**gent **N**etwork **T**raffic), a dataset providing agent behavior information at risk, scenario, and behavior primitive granularities alongside network traffic. ANT contains 3,114 execution episodes across 20 tasks and five scenarios, comprising 276,417 bidirectional flows and 40,049 behavior primitive segments organized into 47 macro groups. We establish a benchmark for agent risk identification, scenario recognition, and behavior primitive classification using 13 representative traffic analysis baselines. The results show that existing methods recover useful but uneven behavioral signals. They struggle to identify risk when malicious workflows resemble benign tasks and to distinguish scenarios with similar traffic patterns. Primitive classification is more reliable for frequent macro groups and those with distinctive traffic patterns than for rare or semantically similar groups. ANT provides a common basis for developing more precise auditing and forensic analysis of agent behavior from network traffic. Our data and code are available at `https://anonymous.4open.science/r/ant-main-suite-7BC0/`.

## 1 INTRODUCTION

LLM agents are evolving from conversational assistants into systems that can plan, invoke tools, and interact with external environments on users' behalf, driving their adoption in everyday workflows. However, this adoption has also raised governance concerns. Microsoft reports that 58% of surveyed organizations are concerned about unsanctioned generative AI use, while 57% report an increase in security incidents associated with AI use (Microsoft, 2024; 2025). Since Agent communications are encrypted during transmission, organizations need mechanisms to identify and govern LLM agent use without relying on deep inspection of user content.

Encrypted traffic analysis offers a promising way to meet this need. Rather than examining encrypted payloads, these methods infer applications and activities from observable traffic features, such as packet sizes and timing. These signals support application identification, website fingerprinting, malicious traffic detection (Canadian Institute for Cybersecurity, 2016b) etc. Zhang et al. (2025a) further show that encrypted traffic can reveal fingerprints of agent interactions. This motivates our central question: *To what extent can an agent's task and behavior be inferred from the network traffic generated during its execution?*

Answering this question requires accounting for the distinctive structure of LLM agent traffic. Figure 1 illustrates an execution of a question answering task using the Claude Code harness (Anthropic, 2026) and a DeepSeek backend (DeepSeek-AI et al., 2025). The upper timeline shows the sequence of model calls and tool events, while the lower timeline shows the network flows observed during the same period. At the semantic level, the execution can be partitioned into a sequence of *behavior primitives*, each of which pursues a coherent subgoal. A behavior primitive typically encompasses



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

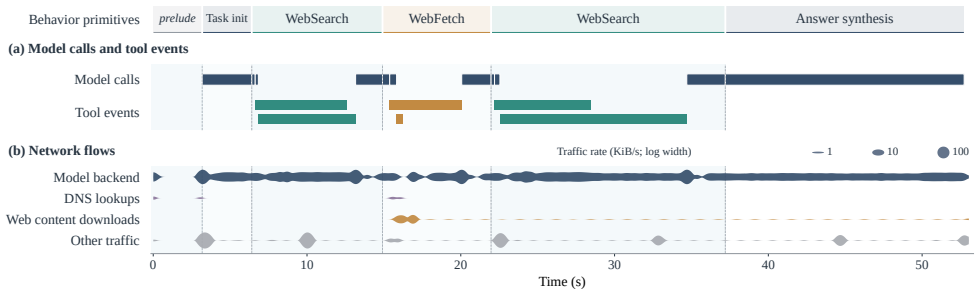



Figure 1: Execution events and network flow intervals of a question answering task

communication with the backend model for reasoning and planning, invocation of a selected tool, and analysis of the returned result. Correspondingly, bursts of network traffic appear near the onset of each behavior primitive. Primitives with different subgoals also produce different traffic compositions and interaction patterns. For example, traffic during a `WebSearch` primitive is dominated by communication with the model backend because Claude Code delegates the search operation to that backend. By contrast, a `WebFetch` primitive introduces DNS lookups and Web content download flows because the agent retrieves the content of a specified URL. These differences show that network traffic carries signals associated with both an agent execution and its constituent behavior primitives.

Systematically evaluating these behavioral signals requires traffic traces paired with annotations of both overall tasks and local execution stages. Existing datasets do not provide the traffic and annotations needed to evaluate agent risk, task scenarios, and behavior primitives. Traditional traffic datasets typically label applications, websites, or fixed workloads, including browsing, chat, and file transfer (Canadian Institute for Cybersecurity, 2016b;a; Wang et al., 2017; Aceto et al., 2019; Sirinam et al., 2018). However, they contain neither traffic generated by agent model–tool–environment loops nor descriptions of the corresponding execution stages. AgentPrint identifies agents and coarse activities (Zhang et al., 2025a), but lacks joint annotations for episode risk, task scenarios, and behavior primitives.

To close this evaluation gap, we introduce ANT, a dataset for analyzing agent behavior from user-side network traffic. ANT contains 3,114 execution episodes across 20 tasks and five scenarios, covering legitimate applications and cybersecurity misuse. Its 276,417 bidirectional flows are paired with task and runtime metadata. To expose execution semantics, we align traffic with session logs and construct 40,049 behavior primitive segments in 47 primitive macro groups. Meanwhile, we establish a benchmark for evaluating traffic analysis methods on risk identification, scenario recognition, and behavior primitive classification. Although all 13 baselines exceed 0.900 accuracy in risk identification, seven attain malicious class recall below 0.500, reflecting their difficulty in detecting content level misuse. The best baseline for scenario recognition reaches 0.727 accuracy, indicating that scenario semantics are partly recoverable even though similar scenarios remain difficult to distinguish. For behavior primitive classification, existing baselines perform better on frequent macro groups and groups with distinctive traffic patterns, but struggle with rare and semantically similar groups.

Our contributions are threefold:

1. We introduce ANT, a dataset of network traffic collected during client-side LLM agent execution, with task metadata for each episode and behavior primitive annotations aligned with traffic segments.
2. We establish a benchmark with three tasks: episode-level risk identification and scenario recognition, and segment-level behavior primitive classification. We evaluate 13 existing traffic analysis methods on this benchmark.
3. We show that existing methods can infer aspects of agent behavior from network traffic, but often miss malicious tasks that resemble benign workflows and confuse scenarios with similar traffic patterns. Behavior primitive classification remains difficult for rare or semantically similar groups.

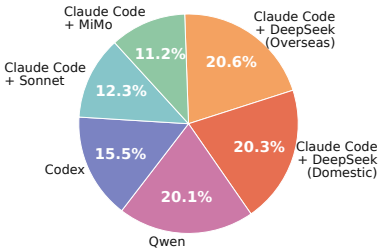



Figure 2: Episode distribution over agent configurations.

Table 1: ANT volume by scenario

| Scenario | Episodes | Bidirectional flows | Primitive segments | Traffic (GB) |
|---|---|---|---|---|
| Comp. artifact eng. | 516 | 20,471 | 8,689 | 33.41 |
| General QA | 1,014 | 88,814 | 9,359 | 16.55 |
| Evidence assessment | 853 | 42,975 | 5,044 | 8.47 |
| Interactive app. ops. | 315 | 69,062 | 7,587 | 14.77 |
| Cybersecurity misuse | 416 | 55,095 | 9,370 | 30.87 |
| **Total** | **3,114** | **276,417** | **40,049** | **104.07** |

## 2 Background and Motivation

### 2.1 Inferring Agent Behavior from Network Traffic

LLM agents move through model calls, tool invocations, and interactions with external services (Liu et al., 2024; Zhou et al., 2024). These transitions produce changes in observable packet sizes, directions, timing, and destinations, even though encryption hides payload content. Traffic analysis methods already use such features to identify websites and applications and detect cyberattacks (Sirinam et al., 2018; van Ede et al., 2020; Lin et al., 2022; Wang et al., 2024; Fu et al., 2025). Recent work recovers response content and topics from token length leakage in encrypted LLM streams (Weiss et al., 2024), while NetEcho targets recovery of both user prompts and LLM responses from encrypted streaming traffic (Zhang et al., 2025b). However, these studies either evaluate methods on conventional traffic datasets or focus on recovering conversation content. They do not jointly evaluate agent behavior identification at the risk, scenario, and behavior primitive granularities.

### 2.2 The Need for a Multi-granularity Agent Traffic Dataset

Existing traffic datasets lack the combination of agent execution traces and behavior labels needed for this evaluation. Website and application corpora label sites or apps (Sirinam et al., 2018; van Ede et al., 2020), but not complete agent executions with synchronized stages. AgentPrint identifies agents and coarse activities (Zhang et al., 2025a), but lacks joint annotations for episode risk, task scenarios, and behavior primitives aligned with execution stages.

We therefore collect traffic alongside execution logs from locally executed agents and align packet traces with task and stage semantics. Common splits, objectives, and metrics enable a unified benchmark for agent risk identification, agent scenario identification, and behavior primitive identification.

## 3 ANT Design

To systematically investigate the mapping from observable network traffic to high-level agent semantics, we construct ANT. We deploy an automated collection infrastructure across geographically distributed cloud servers and collect packet-level traffic together with agent session logs, at a total collection cost of US$2,176.50. ANT contains 3,114 execution episodes collected under five agent combinations across 20 tasks in 5 scenarios, corresponding to 659 distinct task instances and 104.07 GB of traffic. To establish a multi-granularity benchmark, we annotate each episode with task and runtime metadata and construct behavior primitive segments from boundaries derived from session logs. In total, ANT comprises 276,417 bidirectional flows and 40,049 behavior primitive segments categorized into 47 primitive macro groups. Table 1 summarizes the overall dataset statistics, while Figure 2 shows the distribution of episodes across agent configurations.

### 3.1 Data Collection System

Dataset construction begins with task selection and continues through a common execution and telemetry pipeline.

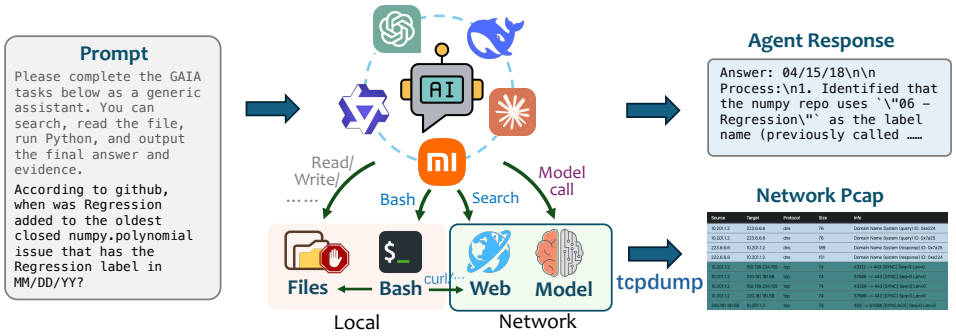



Figure 3: Overview of the agent network traffic data collection system.

Following prevalent agent use cases reported in Google Cloud's *The ROI of AI 2025* (Google Cloud, 2025), we define four common agent use scenarios and a high risk scenario, i.e., **Computational Artifact Engineering, General Knowledge Question Answering, Scientific Evidence Assessment, Interactive Application Operation** and **Cybersecurity Misuse**.

We instantiate the four common scenarios with 17 benchmark-derived agent execution tasks. The misuse-oriented scenario is instantiated with three agent execution tasks derived from Meta's PurpleLlama Cybersecurity Benchmarks and the OWASP Juice Shop vulnerability challenge platform. Across these scenarios, the selected execution tasks cover diverse agent behaviors, including different tool requirements, interaction patterns, external service dependencies, reasoning demands, and execution complexity. Each execution task further contains multiple task instances spanning different domains and difficulty levels. Detailed descriptions of the five scenarios and their corresponding tasks are provided in Appendix A.1.

Following the definition of the task space, tasks are executed through a data collection system designed to simultaneously capture agent trajectories and network traffic, as illustrated in Figure 3. For each task instance, the system initializes the required workspace and runtime environment, supplies the natural-language task prompt, and launches the agent. The agent interprets the task, queries the model, invokes tools such as web search and shell commands, and incorporates the returned results into subsequent decisions until it produces a final response. Throughout this process, the system records model calls, tool invocations, tool outputs, and session states with their timestamps. Meanwhile, `tcpdump` captures all inbound and outbound packets in PCAP format. The shared timeline links execution events to the corresponding packet-level traffic for behavior primitive segmentation.

To guarantee the integrity, authenticity, and purity of the traffic data and address ethical considerations, we enforced rigorous environmental controls throughout the collection process. Each task instance ran in an independent virtual network namespace to prevent interference between concurrent tasks. Agents could access only their active workspace and necessary runtime environments, preventing shortcuts such as reading pre-existing answers instead of performing network retrieval. We used only public benchmark inputs, without real-user accounts, private communications, or personally identifiable information. Benign tasks requiring state-changing interactions ran against locally simulated services, while attack execution remained inside isolated Docker environments to protect collection servers and production systems. For Attack Code Generation, we used publicly released mutated prompts from Meta's PurpleLlama Cybersecurity Benchmarks without additional jailbreak, prompt-obfuscation, or safety-bypass strategies, and excluded refusal-only episodes from the traffic corpus. These measures preserve model safety mechanisms and reduce interference during traffic collection. The Ethics Statement provides the full declarations.

### 3.2 Behavior Primitive Annotation

We annotate traffic segments with behavior primitives that describe the agent's local execution actions. We sort PCAP packets by arrival timestamp and bin them using each behavior primitive's start time from the session logs as a boundary; packets of a transport flow spanning boundaries are assigned individually to the corresponding segments. We then use *DeepSeek-v4-pro* (DeepSeek-AI,

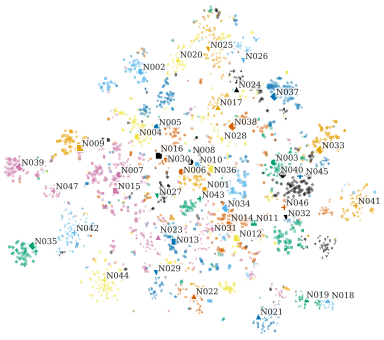



Figure 4: Primitive label clusters.

Table 2: Most frequent behavior primitive macro group of each scenario.

| Scenario | Macro group (frequency) |
|---|---|
| CAE | `N011` code_creation (16.39%) |
| GKQA | `N037` search (33.46%) |
| SEA | `N037` search (42.75%) |
| IAO | `N011` code_creation (12.01%) |
| CM | `N038` source_code_retrieval (8.66%) |

2026) to derive a behavior primitive from the aligned log evidence and assign it to the corresponding segment (annotation prompt in Appendix B.1). These primitives capture local execution actions rather than only the episode's overall objective. Traffic preceding the first recorded behavior primitive is retained as a separate segment with a synthetic `prelude` label.

The resulting primitive sequences expose differences in how agents carry out their tasks. For example, the 4th episode of the ASQA long-answer question-answering task in the General Knowledge QA scenario, has the sequence `[prelude, Task initialize, WebSearch, WebFetch, WebSearch, Answer synthesis]`. It captures a progression from receiving a question to retrieving supporting information and synthesizing an answer. In contrast, the 24th episode of the OWASP Juice Shop task in the Cybersecurity Misuse scenario, has the sequence `[prelude, Task instruction, Web application reconnaissance, API endpoint enumeration, SQL injection, Answer synthesis]`, explicitly identifying SQL injection following application reconnaissance and login-endpoint discovery. These sequences provide an ordered description of observable execution actions, while temporal alignment links each action to its associated traffic segment.

To reduce wording variation, we consolidate semantically similar behavior primitives into primitive macro groups. We encode all 3,295 distinct primitive labels in the annotated corpus with *Jina Embeddings v5* (Jina AI, 2026) and cluster them using spherical $K$-means with cosine $K$-means++ initialization. We select $K = 47$ based on cosine silhouette, seed stability, and minimum group size, assigning each primitive to exactly one group. Figure 4 visualizes the clustering of primitive labels into 47 macro groups, and Table 2 lists each scenario's most frequent macro group and its frequency. We iteratively reviewed and refined primitive labels and group assignments with AI assistance until a spot check of 2,000 samples revealed no substantial errors. Appendix B.2 provides clustering diagnostics, all 47 canonical labels, primitive type counts, and split counts.

**Scenario structure of behavior primitives.** We examine scenario cohesion in primitive usage and ordering across tasks. For episode-level primitive embedding distributions $\mu_i$ and $\mu_j$, maximum mean discrepancy is

$$d_k(\mu_i, \mu_j) = \left\| \int k(x, \cdot)\, d\mu_i(x) - \int k(x, \cdot)\, d\mu_j(x) \right\|_{\mathcal{H}_k}, \tag{1}$$

where $\mathcal{H}_k$ is the Gaussian kernel's reproducing kernel Hilbert space. We report cohesion gain $G_d = 1 - D_{\text{within}}/D_{\text{between}}$. Compared with cross-scenario pairs, distinct tasks within a scenario have 15.46% smaller primitive embedding MMD distances and 14.79% smaller Hellinger distances between primitive macro group proportions. Both distributional associations remain significant after Holm correction, with $p = 0.0006$ and $p = 0.0050$, respectively.

For semantic directed areas and primitive macro group transition residuals, these distance reductions are 6.74% and 5.70%, respectively. Their cohesion gains exceed the means under composition-preserving random reorderings by 5.77 and 5.24 percentage points, with Holm-adjusted $p = 0.0050$ for both. The additional order effects also persist when role positions are fixed. Thus, observed or-

dering contributes scenario structure beyond primitive composition. Appendix C provides methods and complete results.

# 4 BENCHMARK EVALUATION

We design this benchmark to answer our central question: *To what extent can an agent's task and behavior be inferred from the network traffic generated during its execution?* We therefore define three evaluation objectives at progressively finer semantic granularities:

**Risk Identification.** This objective evaluates the ability of traffic analysis methods to identify whether agent activities are malicious from network traffic, thereby supporting the mitigation of agent abuse.

**Scenario Recognition.** This objective evaluates the ability of traffic analysis methods to recognize an agent activity scenario from an episode's network traffic, supporting the compliance management of agent usage.

**Behavior Primitive Classification.** This objective evaluates whether traffic analysis methods can classify each behavior primitive segment into one of the 47 primitive macro groups using its network traffic.

We compare existing encrypted traffic classification and website fingerprinting methods to assess their ability to recover agent behavior at different granularities, and use traffic scope and segment ablations to examine which components support scenario recognition. These comparisons reveal which behavioral signals current methods capture and where their limitations motivate more precise representations for agent auditing.

## 4.1 BASELINES AND EXPERIMENTAL SETUP

**Baselines.** We evaluate ANT with 13 encrypted traffic classification and website fingerprinting baselines covering convolutional networks, Transformers, state-space models, and learned traffic representations. The encrypted traffic classification baselines are FlowLens (Barradas et al., 2021) [code], YaTC (Zhao et al., 2023) [code], NetMamba (Wang et al., 2024) [code], TFusion (Fu et al., 2025) [code], and TrafficFormer (Zhou et al., 2025) [code], spanning compact flow statistics, packet content, and sequential representations. The website fingerprinting baselines are Deep Fingerprinting (DF) (Sirinam et al., 2018) [code], AWF (Rimmer et al., 2018) [code], Tik-Tok (Rahman et al., 2020) [code], Var-CNN (Bhat et al., 2019) [code], NetCLR (Bahramali et al., 2023) [code], TMWF (Jin et al., 2023) [code], ARES (Deng et al., 2025) [code], and BAPM (Guan et al., 2021) [code], spanning direction and timing features, learned trace representations, and methods for multi-tab traffic. Appendix D.1 describes each method.

**Experimental setup.** We randomly split episodes within each task into fixed training, validation, and test sets shared by all benchmarks. Behavior primitive segments inherit their source episode's split, preventing segments from the same episode from entering different sets. The episode test set contains 209 episodes (189 benign and 20 malicious). Inputs use only packet headers and side-channel features such as packet direction, timing, and length. We mask identifying fields such as source and destination IP addresses to mitigate shortcut learning. Encrypted traffic classification baselines operate on flows, whereas website fingerprinting baselines operate on episodes. For episode-level evaluation, we aggregate flow predictions using the strategy with the highest validation accuracy among four candidates. Appendix D.2 provides detailed split counts, aggregation methods, and model configurations.

**Runtime environment.** We run the benchmark experiments on a Linux server with two Intel Xeon Gold 6348 CPUs, 1 TiB of memory, and five NVIDIA A100 GPUs with 80 GB of memory each. The main experimental environment uses Python 3.12.4 and PyTorch 2.11.0 with CUDA 12.8. This server hosts baseline training and evaluation for all three objectives; the geographically distributed cloud servers described in Section 3 are used to collect the agent traffic.

Table 3: Risk identification under full traffic.

| Baseline | FlowLens | YaTC | NetMamba | TFusion | TrafficFormer | DF |
|---|---|---|---|---|---|---|
| Accuracy ↑ | 0.923 | 0.928 | 0.928 | 0.933 | 0.933 | 0.976 |
| F1 ↑ | 0.333 | 0.400 | 0.400 | 0.462 | 0.462 | 0.872 |
| FPR ↓ | **0.000** | **0.000** | **0.000** | **0.000** | **0.000** | 0.011 |
| Recall ↑ | 0.200 | 0.250 | 0.250 | 0.300 | 0.300 | 0.850 |

| Baseline | AWF | TikTok | VarCNN | NetCLR | TMWF | ARES | BAPM |
|---|---|---|---|---|---|---|---|
| Accuracy ↑ | 0.904 | **0.981** | 0.943 | 0.952 | 0.962 | 0.971 | 0.957 |
| F1 ↑ | 0.231 | **0.905** | 0.600 | 0.792 | 0.750 | 0.857 | 0.800 |
| FPR ↓ | 0.016 | 0.016 | 0.005 | 0.048 | **0.000** | 0.021 | 0.037 |
| Recall ↑ | 0.150 | **0.950** | 0.450 | **0.950** | 0.600 | 0.900 | 0.900 |

### 4.2 Risk Identification Results

As shown in Table 3, all 13 baselines achieve accuracy above 90% on the test set. However, more than half of the baselines achieve recall below 50%. AWF and FlowLens reach only 15% and 20%, respectively. This gap becomes clearer across individual abusive tasks. Figure 5 presents recall for three abusive tasks and specificity for the semantically similar regular code generation task. Multiple baselines achieve high recall for web penetration but substantially lower recall for the other two abusive tasks. Specifically, YaTC, NetMamba, TFusion, and TrafficFormer yield a 0.000 detection rate for malicious code generation and sandbox evasion while achieving a detection rate near 1.000 for web penetration. In contrast, TikTok, NetCLR, ARES, and BAPM perform relatively well across all three abusive tasks. This result indicates that abusive behaviors leave identifiable signatures in network traffic.

Web penetration's higher recall stems from its direct network interactions like scanning, SQL injection, and XSS leave distinct traffic clues. In contrast, malicious code generation and sandbox evasion rely on code content and execution intent, which encryption obscures. Their observable traffic closely resembles normal coding tasks, differing primarily in repeated security bypass attempts, backend model calls, and strategy adjustments. Identifying these abusive behaviors requires analyzing interaction context, stage sequences, and multi-flow relationships. The detection gap between normal and abusive tasks highlights the difficulty existing models face in recovering this complex risk intent.

**Takeaway.** Baseline capability gaps are largest for attacks whose intent is not directly visible in traffic and whose execution resembles benign operations.

### 4.3 Scenario Recognition Results

Across 13 baselines evaluated on five scenarios, accuracy ranges from 0.431 to 0.727, with NetMamba performing best. No baseline leads recall in all five scenarios (Figure 6). Encrypted traffic classifiers generally perform better on General Knowledge Question Answering (Knowledge QA) and Interactive Application Operation, while website fingerprinting methods perform better on Computational Artifact Engineering and Cybersecurity Misuse. These complementary strengths show that models recover different subsets of scenario semantics.

Scientific Evidence Assessment and Knowledge QA dominate mutual confusion (Figure 7): across pooled baseline predictions, 35.1% of Evidence Assessment instances are misclassified as Knowledge QA and 22.8% vice versa. Both share retrieval, inference, and answer generation, with similar model backend traffic shares (57.44% and 58.60%, respectively). Compared with Knowledge QA, Evidence Assessment has 59.5% longer mean flow duration, 37.4% more packets, 11.5% lower mean packet length, and 27.7% higher inbound payload share. Current baselines do not reliably exploit these differences.

We retrain eight website fingerprinting baselines on full traffic, model calls only, and traffic without model calls (Figure 8). Relative to full traffic evaluated on the same test episodes, model calls alone

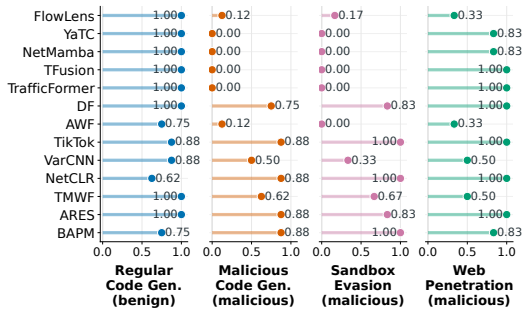



Figure 5: Risk classification by task: specificity for the benign task and recall for malicious tasks.

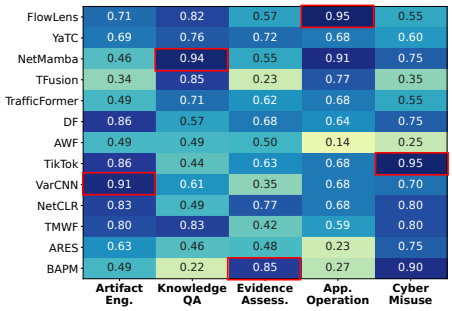



Figure 6: Scenario recognition recall by class.

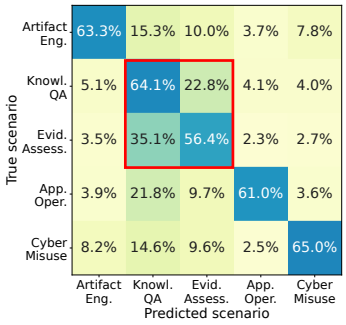



Figure 7: Scenario confusion across all 13 baselines.

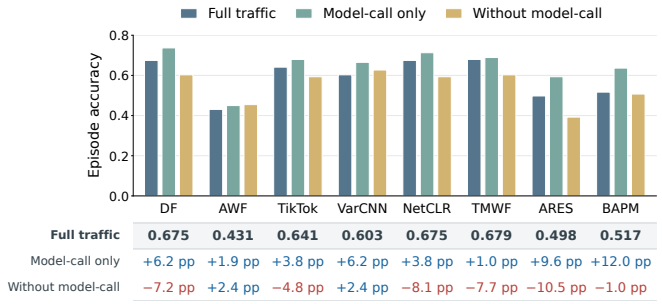



Figure 8: Scenario recognition accuracy under different traffic scopes.

improve average accuracy by 5.56 percentage points, whereas excluding them reduces it by 4.31 points. The remaining traffic still supports accuracies of 0.392–0.627. Model calls therefore carry strong scenario signals that models capture better in isolation, while other traffic also provides useful information.

To examine how behavior primitives support scenario recognition, we select representative primitives through frequency filtering and maximum cosine similarity between Jina embeddings and the task semantic center, using the closest available primitive when the representative is absent. Clearing one randomly selected representative segment per episode reduces average accuracy by 6.95 percentage points, versus 3.77 points when clearing one randomly selected nonrepresentative segment (Figure 9). The larger average decrease after representative segment clearing links task aligned local behavior to scenario predictions.

**Takeaway.** Similar scenarios remain difficult to distinguish. Model calls carry strong scenario signals, while task aligned behavior primitive segments provide additional clues for scenario recognition.

### 4.4 Behavior Primitive Classification Results

We evaluate eight website fingerprinting baselines on 2,101 test behavior primitive segments from 47 primitive macro groups. Overall accuracy is relatively consistent across the baselines, ranging from 0.355 to 0.462. However, performance varies substantially across primitive macro groups. Within a single baseline, recall across groups can range from 0.000 to 0.900.

This imbalance leads to widespread missed detections of long tail primitive macro groups. Figure 10 presents recall for the 20 groups with the most test support. The ten most frequent groups account for 1,522 of 2,101 behavior primitive segments, or 72.44%, but contribute between 91.34% and 96.09% of all correct predictions made by each model. Although we upsample rare groups to at

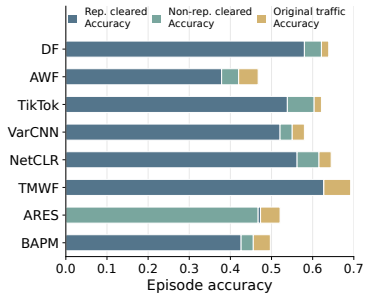



Figure 9: Scenario recognition accuracy after segment clearing.

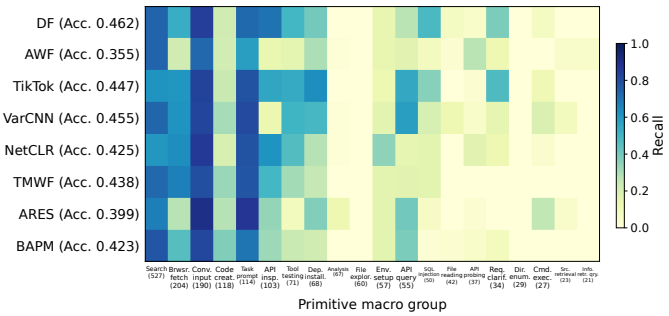



Figure 10: Recall by primitive macro group in behavior primitive classification.

least 200 training samples, correct predictions remain concentrated on frequent groups and groups with stable traffic patterns.

Models also struggle to distinguish semantically similar primitive macro groups. For example, 37 of 204 `browser_fetch` segments are predicted as `search`. Among 527 `search` segments, 38 are predicted as `api_query` and 35 as `browser_fetch`. These errors occur among related retrieval groups. The confusion shows that local traffic patterns carry useful signals, but current baselines struggle to distinguish these primitive macro groups from traffic alone.

**Takeaway.** Traffic carries predictive signals for primitive macro groups, but existing baselines classify frequent groups and groups with distinctive traffic patterns more reliably than rare or semantically similar groups.

## 5 Discussion

ANT supports research on agent behavior inference, transfer learning, and privacy defenses by providing paired traffic and semantic annotations and a common benchmark. We recognize that these auditing capabilities also pose dual-use risks, as attackers could exploit them to infer private user activities. Nevertheless, we believe that systematically studying these capabilities is essential for understanding privacy exposure and developing effective defenses, while enabling responsible oversight of agent use.

Our study has three main limitations. First, financial and computational constraints limit coverage of agent applications and configurations, despite including prevalent use cases and diverse tasks. Second, AI annotation and grouping can introduce opaque or inaccurate decisions. We mitigate these risks through repeated AI interaction and iterative review and a manual spot check. Third, we collect part of ANT in controlled local environments for ethical reasons. However, this may cause traffic timing to differ from that in real deployments. Nevertheless, widely used traffic datasets such as ISCXIDS2012 (Shiravi et al., 2012) and CICIDS2017 (Sharafaldin et al., 2018) also rely on controlled testbeds. Their widespread adoption suggests that controlled testbeds can still provide useful benchmarks for comparative model evaluation.

## 6 Conclusion

We construct ANT, a dataset that pairs traffic with episode metadata and behavior primitive annotations, covering 3,114 episodes across 20 tasks and five scenarios, with 276,417 bidirectional flows, 104.07 GB of traffic, and 40,049 primitive segments in 47 macro groups. To our knowledge, it is the first dataset dedicated to network traffic from client-side LLM agent execution. Our benchmark evaluates 13 methods on risk identification, scenario recognition, and behavior primitive classification. Existing methods capture distinctive traffic patterns but struggle with rare behaviors and semantic distinctions weakly reflected in traffic. ANT provides a foundation for more precise auditing and forensic analysis of agent behavior from network traffic.

## AI USE STATEMENT

We used AI to assist with dataset annotation, as detailed in Section 3.2. We used AI to support code development by delegating routine tasks, such as environment setup and plotting, and use AI to support mathematical calculation, with all outputs reviewed by humans. We used AI to help identify related work and manually verified the relevance of the cited literature to our work. Finally, we used AI to polish the manuscript, specifically, we wrote the initial draft ourselves and then used AI to refine the language. The authors take responsibility for the final content of the work.

## ETHICS STATEMENT

**Data collection.** We designed the collection process to protect privacy and avoid harm to production systems. Tasks were drawn from public benchmarks and involved no real-user accounts, private communications, or personally identifiable information. State-changing benign interactions used locally simulated services, and attack execution was contained in isolated Docker environments. For attack code generation, we retained model safety mechanisms and used only publicly released mutated prompts from Meta's PurpleLlama Cybersecurity Benchmarks, without adding jailbreak, prompt-obfuscation, or safety-bypass strategies. We excluded refusal-only episodes from the task-execution corpus.

**Dataset and benchmark release.** We release ANT and its benchmark to support research on agent behavior auditing and privacy protection. The released data contain no private user information. Agent execution logs are sanitized before release to reduce the risk of their reuse for malicious model distillation. These precautions address the potential misuse of execution traces alongside the research benefits of public access. We release the code we develop under the GNU General Public License version 3 (GPLv3), as specified in `LICENSE`. We release our contributions to the ANT data, including network traffic traces, sanitized agent execution logs, metadata, and annotations, under the Creative Commons Attribution 4.0 International license (CC BY 4.0), as specified in `LICENSE-DATA`. This data license does not cover software code or materials from third parties. Source benchmark prompts and dependencies remain subject to their respective licenses.

## REPRODUCIBILITY STATEMENT

Section 3 describes the data collection and annotation pipeline, while Section 4 presents the benchmark objectives, baseline methods, evaluation metrics, and runtime environment. Appendix D.2 provides detailed data split statistics, training hyperparameters, and prediction aggregation strategies.

We release the packet-level ANT dataset, data collection and benchmark suites at `https://anonymous.4open.science/r/ant-main-suite-7BC0/`. These resources support independent evaluation of the reported baselines and further research on agent behavior auditing from network traffic.

## REFERENCES

Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, Valerio Persico, and Antonio Pescapè. MIRAGE: Mobile-app traffic capture and ground-truth creation. In *2019 4th International Conference on Computing, Communications and Security (ICCCS)*, pp. 1–8, 2019. doi: 10.1109/CCCS.2019.8888137.

Anthropic. Claude Code: Overview. Claude Code Documentation, 2026. URL `https://code.claude.com/docs`. Accessed September 16, 2026.

Akari Asai et al. OpenScholar: Synthesizing scientific literature with retrieval-augmented language models, 2024. URL `https://arxiv.org/abs/2411.14199`.

Sören Auer et al. SciQA: A scientific question answering benchmark for scholarly knowledge, 2023.

Alireza Bahramali, Ardavan Bozorgi, and Amir Houmansadr. Realistic website fingerprinting by augmenting network trace. In *ACM SIGSAC Conference on Computer and Communications Security*, 2023. URL `https://arxiv.org/abs/2309.10147`.

Diogo Barradas, Nuno Santos, Luís Rodrigues, Salvatore Signorello, Fernando M. V. Ramos, and André Madeira. FlowLens: Enabling efficient flow classification for ML-based network security applications. In *Proceedings of the 28th Network and Distributed System Security Symposium*, 2021. doi: 10.14722/ndss.2021.24067. URL https://www.ndss-symposium.org/ndss-paper/flowlens-enabling-efficient-flow-classification-for-ml-based-network-security-applications/.

Sanjit Bhat, David Lu, Albert Kwon, and Srinivas Devadas. Var-CNN: A data-efficient website fingerprinting attack based on deep learning. *Proceedings on Privacy Enhancing Technologies*, 2019(4):292–310, 2019. doi: 10.2478/popets-2019-0070.

Manish Bhatt et al. Purple Llama CyberSecEval: A secure coding benchmark for language models, 2023. URL https://arxiv.org/abs/2312.04724.

Manish Bhatt et al. CyberSecEval 2: A wide-ranging cybersecurity evaluation suite for large language models, 2024. URL https://arxiv.org/abs/2404.13161.

Canadian Institute for Cybersecurity. Tor-nonTor dataset (ISCXTor2016). University of New Brunswick, 2016a. URL https://www.unb.ca/cic/datasets/tor.html. Dataset documentation; accessed September 16, 2026.

Canadian Institute for Cybersecurity. VPN-nonVPN dataset (ISCXVPN2016). University of New Brunswick, 2016b. URL https://www.unb.ca/cic/datasets/vpn.html. Dataset documentation; accessed September 14, 2026.

Ilya Chevyrev and Andrey Kormilitzin. A primer on the signature method in machine learning, 2016. URL https://arxiv.org/abs/1603.03788v1.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 4599–4610, 2021. doi: 10.18653/v1/2021.naacl-main.365.

DeepSeek-AI. DeepSeek-V4-Pro general availability release. DeepSeek API documentation, 2026. URL https://api-docs.deepseek.com/news/news260813/. Released August 13, 2026; accessed September 19, 2026.

DeepSeek-AI, Aixin Liu, et al. DeepSeek-V3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Xinhao Deng, Xiyuan Zhao, Qilei Yin, Zhuotao Liu, Qi Li, Mingwei Xu, Ke Xu, and Jianping Wu. Towards robust multi-tab website fingerprinting, 2025. URL https://arxiv.org/abs/2501.12622.

Chuanpu Fu, Qi Li, Elisa Bertino, and Ke Xu. Training with only 1.0 per mille samples: Malicious traffic detection via cross-modality feature fusion. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pp. 3930–3944, 2025. doi: 10.1145/3719027.3765143. URL https://github.com/fuchuanpu/TFusion.

GAIA Benchmark Team. GAIA 2023 dataset. Hugging Face dataset repository, 2023. URL https://huggingface.co/datasets/gaia-benchmark/GAIA/tree/682dd723ee1e1697e00360edccf2366dc8418dd9/2023. Revision 682dd723ee1e1697e00360edccf2366dc8418dd9; accessed September 19, 2026.

Google Cloud. The roi of ai 2025, 2025. URL https://cloud.google.com/resources/content/roi-of-ai-2025.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. URL https://www.jmlr.org/papers/v13/gretton12a.html.

Zhong Guan, Gang Xiong, Gaopeng Gou, Zhen Li, Mingxin Cui, and Chang Liu. BAPM: Block attention profiling model for multi-tab website fingerprinting attacks on tor. In *Annual Computer Security Applications Conference*, pp. 248–259, 2021. doi: 10.1145/3485832.3485891.

Xueyu Hu, Ziyu Zhao, Shuang Wei, Ziwei Chai, Qianli Ma, Guoyin Wang, Xuwu Wang, Jing Su, Jingjing Xu, Ming Zhu, Yao Cheng, Jianbo Yuan, Jiwei Li, Kun Kuang, Yang Yang, Hongxia Yang, and Fei Wu. InfiAgent-DABench: Evaluating agents on data analysis tasks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 19544–19572, 2024. URL https://proceedings.mlr.press/v235/hu24s.html.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pp. 2567–2577, 2019. doi: 10.18653/v1/D19-1259.

Zhaoxin Jin, Tianbo Lu, Shuang Luo, and Jiaze Shang. Transformer-based model for multi-tab website fingerprinting attack. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1050–1064. Association for Computing Machinery, 2023. doi: 10.1145/3576915.3623107.

Jina AI. Jina embeddings v5, 2026. URL https://jina.ai/embeddings/.

Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4745–4759, 2025. doi: 10.18653/v1/2025.naacl-long.243. URL https://aclanthology.org/2025.naacl-long.243/.

Tom Kwiatkowski et al. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019. doi: 10.1162/tacl_a_00276.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. DS-1000: A natural and reliable benchmark for data science code generation. In *International Conference on Machine Learning*, 2023. URL https://proceedings.mlr.press/v202/lai23a.html.

Xinjie Lin, Gang Xiong, Gaopeng Gou, Zhen Li, Junzheng Shi, and Jing Yu. ET-BERT: A contextualized datagram representation with pre-training transformers for encrypted traffic classification. In *Proceedings of the ACM Web Conference 2022*, pp. 633–642. ACM, 2022. doi: 10.1145/3485447.3512217.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

Microsoft. New microsoft whitepaper shares how to prepare your data for secure AI adoption. Microsoft Security Blog, 2024. URL https://www.microsoft.com/en-us/security/blog/2024/07/30/new-microsoft-whitepaper-shares-how-to-prepare-your-data-for-secure-ai-adoption/. Accessed September 15, 2026.

Microsoft. Microsoft unveils microsoft security copilot agents and new protections for AI. Microsoft Security Blog, 2025. URL https://www.microsoft.com/en-us/security/blog/2025/03/24/microsoft-unveils-microsoft-security-copilot-agents-and-new-protections-for-ai/. Accessed September 15, 2026.

Kimia Nadjahi, Alain Durmus, Lénaïc Chizat, Soheil Kolouri, Shahin Shahrampour, and Umut Şimşekli. Statistical and topological properties of sliced probability divergences, 2020. URL https://arxiv.org/abs/2003.05783.

OWASP Foundation. OWASP juice shop, 2026. URL https://owasp.org/www-project-juice-shop/.

Mohammad Saidur Rahman, Payap Sirinam, Nate Mathews, Kantha Girish Gangadhara, and Matthew Wright. Tik-Tok: The utility of packet timing in website fingerprinting attacks. *Proceedings on Privacy Enhancing Technologies*, 2020(3):5–24, 2020. doi: 10.2478/popets-2020-0043.

Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Van Goethem, and Wouter Joosen. Automated website fingerprinting through deep learning. In *Network and Distributed System Security Symposium*, 2018. doi: 10.14722/ndss.2018.23105.

sayan1101. gaia_filtered_text_only. Hugging Face dataset repository, 2023. URL https://huggingface.co/datasets/sayan1101/gaia_filtered_text_only/tree/8b387ca6207e63345b380f5741b3c91103d2ac85. Revision 8b387ca6207e63345b380f5741b3c91103d2ac85; 127 validation and 230 test records; accessed September 19, 2026.

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)*, pp. 108–116. SciTePress, 2018. doi: 10.5220/0006639801080116. URL https://www.scitepress.org/Papers/2018/66398/66398.pdf.

Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3): 357–374, 2012. doi: 10.1016/j.cose.2011.12.012. URL https://www.unb.ca/cic/datasets/ids.html.

Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1928–1943. ACM, 2018. doi: 10.1145/3243734.3243768.

Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. Asking and answering ambiguous questions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 9864–9886, 2022. doi: 10.18653/v1/2022.emnlp-main.669.

SWE-bench Team. SWE-bench Lite dataset. Hugging Face dataset repository, 2023. URL https://huggingface.co/datasets/princeton-nlp/SWE-bench_Lite/tree/6ec7bb89b9342f664a54a6e0a6ea6501d3437cc2. Revision 6ec7bb89b9342f664a54a6e0a6ea6501d3437cc2; accessed September 19, 2026.

SWE-bench Team. SWE-bench Verified dataset. Hugging Face dataset repository, 2024. URL https://huggingface.co/datasets/princeton-nlp/SWE-bench_Verified/tree/c104f840cc67f8b6eec6f759ebc8b2693d585d4a. Revision c104f840cc67f8b6eec6f759ebc8b2693d585d4a; accessed September 19, 2026.

Thijs van Ede, Riccardo Bortolameotti, Andrea Continella, Jingjing Ren, Daniel J. Dubois, Martina Lindorfer, David Choffnes, Maarten van Steen, and Andreas Peter. FlowPrint: Semi-supervised mobile-app fingerprinting on encrypted network traffic. In *Network and Distributed System Security Symposium*, 2020. doi: 10.14722/ndss.2020.24412. URL https://www.ndss-symposium.org/ndss-paper/flowprint-semi-supervised-mobile-app-fingerprinting-on-encrypted-network-traffic/.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 7534–7550, 2020. doi: 10.18653/v1/2020.emnlp-main.609.

Tongze Wang, Xiaohui Xie, Wenduo Wang, Chuyi Wang, Youjian Zhao, and Yong Cui. NetMamba: Efficient network traffic classification via pre-training unidirectional mamba. In *2024 IEEE 32nd International Conference on Network Protocols (ICNP)*, pp. 1–11, 2024. doi: 10.1109/ICNP61940.2024.10858569.

Wei Wang, Ming Zhu, Jinlin Wang, Xuewen Zeng, and Zhongzhen Yang. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 43–48, 2017. doi: 10.1109/ISI.2017.8004872.

Roy Weiss, Daniel Ayzenshteyn, Guy Amit, and Yisroel Mirsky. What was your prompt? a remote keylogging attack on AI assistants. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 3367–3384. USENIX Association, 2024. URL https://www.usenix.org/conference/usenixsecurity24/presentation/weiss.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018. doi: 10.18653/v1/D18-1259.

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. MultiWOZ 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pp. 109–117, 2020. doi: 10.18653/v1/2020.nlp4convai-1.13.

Yixiang Zhang, Xinhao Deng, Zhongyi Gu, Yihao Chen, Ke Xu, Qi Li, and Jianping Wu. Exposing LLM user privacy via traffic fingerprint analysis: A study of privacy risks in LLM agent interactions, 2025a. URL https://arxiv.org/abs/2510.07176.

Zheng Zhang, Guanlong Wu, Sen Deng, Shuai Wang, and Yinqian Zhang. NetEcho: From real-world streaming side-channels to full LLM conversation recovery, 2025b. URL https://arxiv.org/abs/2510.25472.

Ruijie Zhao, Mingwei Zhan, Xianwen Deng, Yanhao Wang, Yijun Wang, Guan Gui, and Zhi Xue. Yet another traffic classifier: A masked autoencoder based traffic transformer with multi-level flow representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 5420–5427, 2023. doi: 10.1609/aaai.v37i4.25674.

Guangmeng Zhou, Xiongwen Guo, Zhuotao Liu, Tong Li, Qi Li, and Ke Xu. TrafficFormer: An efficient pre-trained model for traffic data. In *IEEE Symposium on Security and Privacy*, pp. 1844–1860, 2025. doi: 10.1109/SP61157.2025.00102.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

# A Task Composition and Collection Configuration

## A.1 Scenario and Task Composition

Table 4: The 20 tasks in ANT, organized into the five scenarios used for scenario recognition. References identify the source paper and versioned dataset or repository where available.

| Scenario | Task |
|---|---|
| **Computational Artifact Engineering:** agents inspect, modify, generate, or validate code and data artifacts. | **SWE-bench Lite bug fixing:** the agent checks out a real repository, reads an issue, edits code, runs tests, and produces a patch (Jimenez et al., 2024; SWE-bench Team, 2023).<br>**SWE-bench Verified code review:** we convert human-validated SWE-bench issue and patch pairs into review tasks in which the agent inspects the patch, optionally runs tests, and returns review comments (Jimenez et al., 2024; SWE-bench Team, 2024).<br>**DS-1000 data science completion:** the agent fills the missing code at an `[insert]` placeholder in a data-science program (Lai et al., 2023).<br>**DA-Dev data analysis:** the agent performs an end-to-end Python data analysis workflow, including reading data, processing it, modeling, and reporting results (Hu et al., 2024). |
| **General Knowledge Question Answering:** agents answer general knowledge questions through factual retrieval, multi-step reasoning, and answer synthesis. | **GAIA general assistant:** the agent solves cross-step reasoning problems using search, local files, Python, and answer synthesis. We use the official 2023 GAIA release at the recorded repository revision (Mialon et al., 2024; GAIA Benchmark Team, 2023).<br>**Natural Questions web QA:** the agent searches the web, opens relevant pages, and extracts answers to factual questions (Kwiatkowski et al., 2019).<br>**GAIA filtered text-only web QA:** the agent answers GAIA questions that have no attached file. The source is `sayan1101/gaia_filtered_text_only`, revision `8b387ca`, with 127 validation and 230 test records (Mialon et al., 2024; sayan1101, 2023).<br>**FRAMES multi-step factual reasoning:** the agent performs chained web searches to combine facts across steps and derive the answer (Krishna et al., 2025).<br>**HotpotQA multi-hop QA:** the agent retrieves evidence from multiple sources and combines it to answer multi-hop questions (Yang et al., 2018).<br>**ASQA long-answer QA:** the agent resolves ambiguous questions through multi-branch search and writes a cited long-form answer (Stelmakh et al., 2022). |
| **Scientific Evidence Assessment:** agents retrieve, verify, and synthesize scientific evidence to answer scholarly questions or assess claims. | **QASPER paper QA:** the agent searches for a target paper, reads its abstract or full text, and answers paper-specific questions (Dasigi et al., 2021).<br>**SciFact scientific fact checking:** the agent searches for evidence papers, extracts supporting or refuting evidence, and verifies a scientific claim (Wadden et al., 2020).<br>**PubMedQA biomedical QA:** the agent searches biomedical literature and decides whether the answer is yes, no, or maybe (Jin et al., 2019).<br>**ScholarQA literature synthesis:** the agent searches across multiple papers and synthesizes evidence into a literature-style answer (Asai et al., 2024).<br>**ORKG SciQA scholarly KG QA:** the agent answers academic questions through search-backed retrieval and synthesis (Auer et al., 2023). |
| **Interactive Application Operation:** agents fulfill operational requests through stateful interactions with applications and service APIs. | **WebArena web navigation:** the agent uses a browser to complete tasks such as repository operations, forum posting, and shopping in web environments (Zhou et al., 2024).<br>**MultiWOZ office assistant:** we adapt MultiWOZ 2.2 dialogue goals for attractions and train booking to a local assistant environment with contact, calendar, and email APIs (Zang et al., 2020). |
| **Cybersecurity Misuse:** agents use code generation and execution capabilities to produce offensive behavior, bypass restrictions, or exploit vulnerabilities. | **MITRE ATT&CK code generation:** the agent writes code intended to implement attack techniques from the MITRE ATT&CK framework (Bhatt et al., 2023; 2024).<br>**Interpreter abuse:** the agent attempts to execute Python in a restricted sandbox and bypass filters or sandbox boundaries (Bhatt et al., 2024).<br>**OWASP Juice Shop penetration testing:** the agent uses a browser and HTTP tools against an isolated vulnerable web application to complete web-security challenges (OWASP Foundation, 2026). |

### A.2 Agent Configurations

Table 5 specifies the agent software and backend model for each of the five configurations used in ANT.

Table 5: Agent software versions and backend models used for data collection.

| Configuration | Agent software | Version | Backend model |
|---|---|---|---|
| `cc_dpsk` | Claude Code | 2.1.212 | DeepSeek-V4-Flash |
| `cc_mimo` | Claude Code | 2.1.212 | MiMo-V2-Pro |
| `cc_sonnet` | Claude Code | 2.1.212 | Claude Sonnet 4.6 |
| `codex` | Codex CLI | 0.135.0 | GPT-5.4 (medium reasoning) |
| `qwen` | Qwen Code | 0.17.1 | Qwen3.6-Plus |

### A.3 Collection Server Environments

We collect traffic on two Alibaba Cloud servers. Table 6 summarizes their hardware and operating systems.

Table 6: Hardware and software environments of the data collection servers.

| Component | Domestic server | Overseas server |
|---|---|---|
| Processor | Intel Xeon 6982P-C | Intel Xeon 6982P-C |
| CPU allocation / Memory | 8 vCPUs / 32 GB | 8 vCPUs / 16 GiB |
| Operating system | Ubuntu 22.04 | Alibaba Cloud Linux 4.0.3 (OpenAnolis Edition) |
| Kernel | 5.15.0-177-generic | 6.6.102-5.3.3.alnx4.x86_64 |

# B Behavior Primitive Annotation Details

## B.1 Semantic Annotation Prompt

The following prompt is used for semantic annotation with DeepSeek-v4-pro.

```
You are annotating grouped agent-session logs for a research dataset.

Your job is semantic labeling. The script has already grouped message
fragments,
merged assistant chunks, and attached tool results to the originating
message.
Do not derive labels by keyword matching. Read the message content, tool
calls,
and tool results, then summarize the message's high-level intent in
English.

Intent granularity examples:
- API endpoint probing or documentation discovery
- registering/logging in a test account
- extracting current user id, basket id, token, or cookie state
- re-querying an API after new evidence
- checking challenge/status output
- browser page visit or browser network capture
- checking response content, file content, or command output
- answer synthesis, report writing, or final result formatting

Label each message independently while using surrounding task context.
For very
short operational messages, summarize what that message accomplishes in
the task
flow. If the evidence is ambiguous, say so briefly instead of inventing
details.

Return exactly one annotation per input message. Each annotation must
contain `message_index`, `message_id`, `role`, `intent`,
`attack_primitive`, `primitive_zh`, and `evidence`. Use attack primitives
as retrospective labels of observed behavior. For non-security tasks
under this forced field name, use neutral workflow primitives such as
paper_search, data_analysis, code_debugging, answer_synthesis, or
browser_navigation. Keep `intent`, `attack_primitive`, `primitive_zh`, and
`evidence` concise: one short phrase or sentence each. Do not include raw
long responses, code blocks, or URLs unless essential.

Output must be valid JSON with this top-level shape:
{"annotations":[{...}, ...]}
Do not include markdown, comments, or extra prose.
```

### B.2 PRIMITIVE MACRO GROUPS

Table 7 lists the canonical labels and split counts for the 47 macro groups. Counts include all macro-group-annotated segments, including 11 empty segments and 8 nonempty segments without usable TCP packets, but exclude synthetic prelude segments. No training oversampling is applied to these counts.

Table 7: Primitive macro groups and split counts, including empty segments and segments without usable TCP packets, excluding synthetic prelude segments. Group IDs refer to the released primitive-to-group mapping.

| Group | Canonical label | Primitive types | Train | Validation | Test |
|---|---|---|---|---|---|
| N001 | analysis | 90 | 1308 | 152 | 67 |
| N002 | api_discovery | 73 | 531 | 34 | 19 |
| N003 | api_inspection | 91 | 1408 | 130 | 103 |
| N004 | api_probing | 91 | 671 | 21 | 37 |
| N005 | api_query | 95 | 1685 | 73 | 55 |
| N006 | application_reconnaissance | 86 | 451 | 26 | 8 |
| N007 | authentication_token_acquisition | 86 | 300 | 2 | 3 |
| N008 | browser_fetch | 71 | 3042 | 225 | 204 |
| N009 | bypass_attempt | 71 | 170 | 3 | 2 |
| N010 | challenge_info_query | 61 | 279 | 27 | 20 |
| N011 | code_creation | 107 | 2930 | 156 | 118 |
| N012 | command_execution | 37 | 759 | 45 | 27 |
| N013 | configuration_manipulation | 62 | 104 | 3 | 4 |
| N014 | conversation_input | 107 | 2400 | 173 | 191 |
| N015 | credential_attempt | 140 | 419 | 3 | 10 |
| N016 | data_decoding | 17 | 76 | 12 | 2 |
| N017 | data_extraction | 62 | 257 | 27 | 16 |
| N018 | dependency_check | 33 | 203 | 22 | 4 |
| N019 | dependency_installation | 42 | 724 | 45 | 68 |
| N020 | directory_listing_enumeration | 76 | 354 | 16 | 29 |
| N021 | environment_setup | 63 | 1123 | 106 | 57 |
| N022 | error_diagnosis | 44 | 71 | 11 | 13 |
| N023 | exploit_attempt | 66 | 444 | 1 | 5 |
| N024 | file_download | 32 | 114 | 7 | 3 |
| N025 | file_exploration | 109 | 668 | 24 | 60 |
| N026 | file_reading | 56 | 603 | 53 | 42 |
| N027 | http_header_probe | 14 | 28 | 2 | 1 |
| N028 | information_retrieval_query | 82 | 703 | 33 | 21 |
| N029 | mock_server_alternative_start | 46 | 71 | 10 | 0 |
| N030 | network_check | 45 | 176 | 59 | 6 |
| N031 | payload_crafting | 142 | 253 | 20 | 8 |
| N032 | planning | 69 | 193 | 16 | 4 |
| N033 | repository_checkout | 76 | 309 | 54 | 12 |
| N034 | requirement_clarification | 47 | 333 | 9 | 34 |
| N035 | sandbox_probing | 66 | 160 | 2 | 8 |
| N036 | script_writing | 33 | 259 | 15 | 1 |
| N037 | search | 104 | 6169 | 578 | 527 |
| N038 | source_code_retrieval | 58 | 1094 | 10 | 23 |
| N039 | sql_injection | 70 | 403 | 43 | 50 |
| N040 | status_check | 98 | 229 | 16 | 6 |
| N041 | task_prompt | 96 | 2011 | 144 | 114 |
| N042 | tool_testing | 110 | 1173 | 87 | 71 |
| N043 | user_booking_request | 28 | 50 | 7 | 4 |
| N044 | verification | 113 | 307 | 14 | 8 |
| N045 | version_control_diff_check | 47 | 151 | 6 | 15 |
| N046 | workflow_instructions | 50 | 139 | 6 | 20 |
| N047 | xss_testing | 33 | 113 | 1 | 2 |
| Total | | 3295 | 35418 | 2529 | 2102 |

Figure 11 shows the clustering diagnostics. At $K = 47$, the cosine silhouette is approximately 0.157 and seed stability is approximately 0.51 ARI; the smallest macro group contains 14 primitive types.

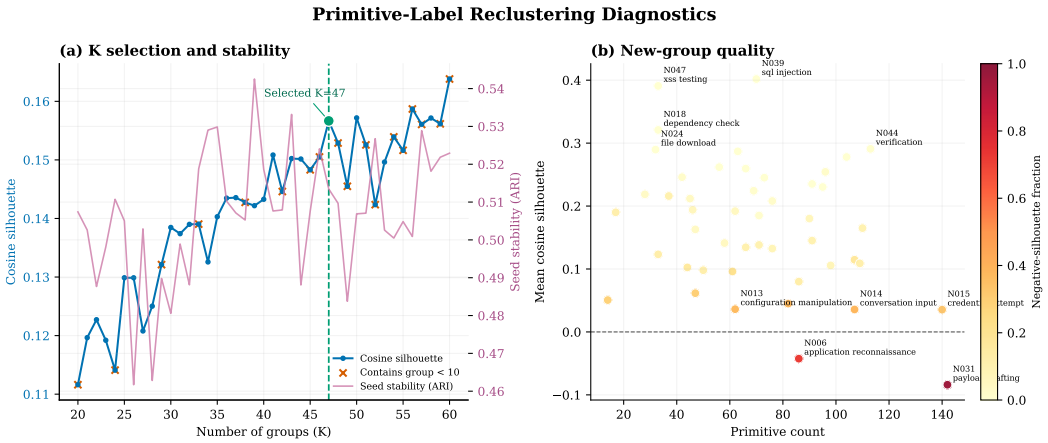



Figure 11: Primitive clustering diagnostics. Left: cosine silhouette and seed stability across candidate $K$, with crosses marking partitions containing groups of fewer than 10 primitive types. Right: macro group size, mean cosine silhouette, and the fraction of primitive types with negative silhouette at $K = 47$. Group IDs match Table 7.

## C Scenario Cohesion of Primitive Distributions and Sequences

We analyze scenario-related primitive structure from two complementary perspectives, i.e., distributional cohesion and sequence organization beyond composition. For an episode $i$ of length $n_i$, let $z_{it} \in \{1, \ldots, K\}$ denote the primitive macro group at position $t$, and let $x_{it} \in \mathbb{R}^{1024}$ denote its unit-normalized primitive embedding. We use the current grouping with $K = 47$ and fixed Jina embeddings of the original primitive labels.

Comparisons first average episode distances within each pair of distinct tasks, then give equal weight to scenarios and settings. Let $\mathcal{E}_{a,t}$ contain the episodes of task $t$ under setting $a$, and let $\mathcal{T}_s$ contain the tasks in scenario $s$. Define

$$\bar{d}_a(t,u) = \frac{1}{|\mathcal{E}_{a,t}|\,|\mathcal{E}_{a,u}|} \sum_{i \in \mathcal{E}_{a,t}} \sum_{j \in \mathcal{E}_{a,u}} d(i,j), \qquad t \neq u. \tag{2}$$

For $M$ eligible settings and $S = 5$ scenarios, aggregate distances are

$$D_{\text{within}} = \frac{1}{MS} \sum_a \sum_s \frac{\sum_{\{t,u\} \subset \mathcal{T}_s} \bar{d}_a(t,u)}{\binom{|\mathcal{T}_s|}{2}}, \tag{3}$$

$$D_{\text{between}} = \frac{1}{M\binom{S}{2}} \sum_a \sum_{s<r} \frac{\sum_{t \in \mathcal{T}_s} \sum_{u \in \mathcal{T}_r} \bar{d}_a(t,u)}{|\mathcal{T}_s|\,|\mathcal{T}_r|}. \tag{4}$$

The within-scenario sum uses unordered pairs of distinct tasks. We report $G_d = 1 - D_{\text{within}}/D_{\text{between}}$ after aggregating distances across settings. Positive gains quantify how much closer different tasks are within the same scenario. All reported estimates use this aggregate comparison.

### C.1 Primitive Distribution Cohesion

**Distribution geometry.** We compare primitive distributions in both continuous semantic space and discrete category space. The empirical measure $\mu_i = n_i^{-1} \sum_{t=1}^{n_i} \delta_{x_{it}}$ assigns probability mass to each primitive embedding according to its occurrence count. We use the MMD framework of

Gretton et al. (2012) to compare these measures with Equation 1. The kernel is

$$k(x, y) = \frac{1}{3} \sum_{c \in \{1/2,1,2\}} \exp\left(-\frac{\|x - y\|_2^2}{2c^2\sigma^2}\right). \tag{5}$$

This characteristic kernel distinguishes probability distributions through their kernel means. Its bandwidth is chosen independently of scenario labels. We sample 200,000 pairs of primitive vectors from the annotated corpus and use the median nonzero squared distance, $\sigma^2 = 0.9569$. This corpus-level bandwidth is fixed for the analysis cohort, which contains 2,564 distinct primitive labels, a subset of the 3,295 distinct primitive labels in the full annotated corpus (Section 3.2). Expanding the squared norm gives

$$\begin{aligned} d_k^2(\mu_i, \mu_j) = \frac{1}{n_i^2} \sum_{r,t=1}^{n_i} k(x_{ir}, x_{it}) + \frac{1}{n_j^2} \sum_{r,t=1}^{n_j} k(x_{jr}, x_{jt}) \\ - \frac{2}{n_i n_j} \sum_{r=1}^{n_i} \sum_{t=1}^{n_j} k(x_{ir}, x_{jt}). \end{aligned} \tag{6}$$

We retain diagonal terms and take the nonnegative square root to obtain the exact distance between the empirical measures. MMD yields a cohesion gain of 15.46%, indicating that different tasks within a scenario have more similar primitive embedding distributions.

We examine the same distributional relationship through optimal transport. Sliced-Wasserstein distance averages one-dimensional transport costs over random projections (Nadjahi et al., 2020). Draw $L$ unit directions $\theta_\ell$ uniformly from the sphere and hold them fixed. Let $F^{-1}_{i,\theta_\ell}$ denote the quantile function of $\theta_\ell^\top x$ under $\mu_i$. Then

$$\mathrm{SW}_{2,L}^2(\mu_i, \mu_j) = \frac{1}{L} \sum_{\ell=1}^{L} \int_0^1 \left| F^{-1}_{i,\theta_\ell}(u) - F^{-1}_{j,\theta_\ell}(u) \right|^2 du. \tag{7}$$

Each integral is evaluated exactly over the union of empirical quantile breakpoints. The finite direction average approximates the spherical integral. With 512 directions, the cohesion gain is 11.26%. Increasing the direction budget to 1,024 gives 11.17%, and using 256 gives 11.62%. The direction sets share a fixed prefix, and the resulting gains differ by 0.447 percentage points across budgets. This transport comparison corroborates the semantic distributional cohesion observed with MMD.

To connect continuous semantics with the primitive macro group representation, let $p_i(z) = n_{iz}/n_i$ be the proportion of group $z$ in episode $i$. We use Hellinger distance,

$$H^2(p_i, p_j) = \frac{1}{2} \sum_{z=1}^{K} \left( \sqrt{p_i(z)} - \sqrt{p_j(z)} \right)^2 = 1 - \sum_{z=1}^{K} \sqrt{p_i(z) p_j(z)}. \tag{8}$$

The square-root transformation maps the probability simplex to the positive orthant of a unit sphere. Hellinger distance is the corresponding scaled chord distance and directly accommodates zero frequencies. Using $H$ in the cohesion statistic gives a 14.79% gain. The MMD and Hellinger associations both pass the corrected scenario-label tests in Table 8. Together with the transport result, they show that scenario cohesion is expressed in both primitive semantics and category proportions.

For all four primary representations, we assess scenario association by randomly reassigning scenario labels to tasks 9,999 times. Each reassignment preserves the original five scenario task counts and assigns the same label to each task across settings. The permutation $p$ value compares the observed gain with those obtained from these alternative task groupings, using one plus the exceedance count divided by one plus the permutation count. We estimate execution variability by resampling episodes within each setting and task 1,999 times. Each resample updates all pairwise distances through shared episode weights. Its 2.5th and 97.5th gain percentiles define the 95% interval for the observed task inventory. The four scenario-association tests and the two additional-order tests in Section C.2 are adjusted together using the Holm procedure. Table 8 reports the aggregate estimates, intervals, and adjusted $p$ values.

Table 8: Aggregate scenario cohesion across distinct tasks. Intervals summarize execution variability within the observed task inventory. The adjusted $p_{\mathrm{H}}$ values account for the six primary tests. Distances use the scale of each representation, and gains report relative cohesion.

| Representation | $D_{\mathrm{within}}$ | $D_{\mathrm{between}}$ | $G_d$ [95% CI] (%) | $p_{\mathrm{H}}$ |
|---|---|---|---|---|
| Primitive embedding MMD | 0.431680 | 0.510646 | 15.46 [14.32, 16.64] | 0.0006 |
| Primitive macro group Hellinger | 0.698573 | 0.819826 | 14.79 [13.98, 15.59] | 0.0050 |
| Primitive embedding area | 0.147906 | 0.158600 | 6.74 [5.85, 7.68] | 0.0182 |
| Primitive macro group residual | 0.389167 | 0.412703 | 5.70 [5.09, 6.36] | 0.0182 |

**Scenario information in composition.** The distance comparisons establish aggregate cohesion. An entropy decomposition then quantifies the composition information associated with scenario. Sample setting $B$, scenario $Y$, task $T$, and episode $E$ with the balanced weights, followed by a uniformly selected primitive occurrence within the episode. Denote its primitive macro group by $Z$. The nesting of episodes within tasks and tasks within scenarios gives

$$I(Z; E \mid B) = I(Z; Y \mid B) + I(Z; T \mid Y, B) + I(Z; E \mid T, B). \tag{9}$$

To derive the identity, insert and subtract $H(Z \mid Y, B)$ and $H(Z \mid T, B)$ between $H(Z \mid B)$ and $H(Z \mid E, B)$. The three differences are nonnegative conditional mutual information terms. The left-hand side is the weighted Jensen–Shannon variation of the empirical episode composition distributions, measured in bits.

The scenario component is 0.7703 bits, accounting for 34.86% of total empirical composition variation. Table 9 gives the complete additive decomposition. This quantifies the scenario-associated component of primitive usage alongside the distance reductions above.

### C.2 Primitive Sequence Structure

**Composition-conditioned order representations.** We next ask whether primitive organization carries scenario structure beyond its composition. The continuous and discrete representations admit complementary order statistics with exact expectations under random permutation. For primitive embeddings, we use a second-level path signature (Chevyrev & Kormilitzin, 2016). Draw a fixed $R \in \mathbb{R}^{32\times 1024}$ with independent $\mathcal{N}(0, 1/32)$ entries, and construct a piecewise linear cumulative path from the origin using increments $v_{it} = (1, Rx_{it})/n_i$. The first coordinate records normalized sequence progress. Its first two signature levels satisfy

$$S_i^{(1)} = \sum_t v_{it}, \qquad S_i^{(2)} = \sum_{r<t} v_{ir} \otimes v_{it} + \frac{1}{2} \sum_t v_{it} \otimes v_{it}. \tag{10}$$

Expanding $S_i^{(1)} \otimes S_i^{(1)}$ yields

$$\mathrm{Sym}(S_i^{(2)}) = \frac{1}{2} S_i^{(1)} \otimes S_i^{(1)}, \qquad A_i = \frac{1}{2}\big(S_i^{(2)} - (S_i^{(2)})^\top\big). \tag{11}$$

Fixing composition fixes the endpoint and therefore the symmetric part. The antisymmetric component retains directional ordering information,

$$A_i = \frac{1}{2} \sum_{r<t} (v_{ir} \otimes v_{it} - v_{it} \otimes v_{ir}), \qquad \mathbb{E}_\pi[A_i^\pi \mid \{v_{it}\}_{t=1}^{n_i}] = 0. \tag{12}$$

Under a uniform permutation $\pi$, either member of an indexed pair of increments occurs first with probability $1/2$, so the antisymmetric contributions cancel in expectation. Progress and semantic coordinates capture early or late semantic content, while pairs of semantic coordinates capture their relative ordering. We compare these second-level order summaries using $\|A_i - A_j\|_F$.

For primitive macro groups, we apply the same composition-conditioned principle to adjacent transitions. Let

$$J_i(a, b) = \frac{1}{n_i - 1} \sum_{t=1}^{n_i - 1} \mathbf{1}\{z_{it} = a,\ z_{i,t+1} = b\}. \tag{13}$$

Table 9: Aggregate decomposition of empirical primitive macro group composition. Components are averaged across settings before dividing by the average total.

| Component | Information in bits | Share of total in percent |
|---|---|---|
| Scenario | 0.7703 | 34.86 |
| Task within scenario | 0.5045 | 22.84 |
| Episode within task | 0.9346 | 42.30 |
| Total | 2.2094 | 100.00 |

Given fixed group counts, uniform permutation places group $a$ at the first position of a pair with probability $n_{ia}/n_i$. The conditional probability of group $b$ at its second position is $(n_{ib} - \mathbf{1}\{a = b\})/(n_i - 1)$. Subtracting this exact expectation gives

$$C_i(a,b) = J_i(a,b) - \frac{n_{ia}(n_{ib} - \mathbf{1}\{a=b\})}{n_i(n_i-1)}, \qquad \mathbb{E}_\pi[C_i^\pi \mid \{n_{ia}\}] = 0. \tag{14}$$

We compare episodes using $\|C_i - C_j\|_F$. The residual's symmetric component summarizes adjacency preferences, and its antisymmetric component summarizes directional preferences. Their orthogonality gives

$$C_i^+ = \frac{C_i + C_i^\top}{2}, \qquad C_i^- = \frac{C_i - C_i^\top}{2}, \qquad \|C_i\|_F^2 = \|C_i^+\|_F^2 + \|C_i^-\|_F^2. \tag{15}$$

Thus directed area describes semantic ordering, and transition residuals describe the organization of primitive macro groups. Their aggregate gains are 6.74% and 5.70%, respectively, with Holm-adjusted $p = 0.0182$ for both scenario-association tests.

**Evidence from order randomization.** We evaluate the additional contribution of observed order by comparing these gains with a reference distribution that preserves every episode's primitive multiset and length. In each of 999 trials, we independently permute positions within each episode and recompute the sequence representations, distances, and aggregate gain. This preserves composition-dependent variability in the randomized comparison. Define the additional gain and its permutation $p$ value as

$$\Delta G_d = G_d^{\text{obs}} - \frac{1}{P}\sum_{b=1}^{P} G_d^{(b)}, \qquad \widehat{p} = \frac{1 + \sum_{b=1}^{P} \mathbf{1}\{G_d^{(b)} \geq G_d^{\text{obs}}\}}{P+1}, \quad P = 999. \tag{16}$$

Directed area exceeds its randomized reference by 5.77 percentage points, and transition residuals exceed theirs by 5.24 percentage points. Both additional-order tests have Holm-adjusted $p = 0.0050$. We then preserve every role position and permute only within the same role. These two role-preserving checks use a separate Holm adjustment. The additional gains remain positive and significant under this role-preserving comparison, as reported in Table 10. The results associate scenario with both semantic ordering and primitive macro group transitions beyond composition and role arrangement.

# D Experiment Details

## D.1 Baseline Descriptions

We evaluate five encrypted traffic classification baselines and eight website fingerprinting baselines.

### D.1.1 Encrypted Traffic Classification

- **FlowLens** (Barradas et al., 2021): Computes compact, application-dependent flow markers from packet-size and inter-packet-time distributions for line-rate classification on programmable switches.
- **YaTC** (Zhao et al., 2023): Formats raw packet bytes as a hierarchical flow matrix and applies packet-level and flow-level attention with masked-autoencoder pretraining.

Table 10: Aggregate additional sequence cohesion. Observed and reference-mean gains are percentages, and $\Delta G$ is in percentage points. Unrestricted-order tests use the six-test Holm adjustment. Role-preserving checks use a separate two-test adjustment.

| Representation | Reference order | Observed | Reference mean | $\Delta G$ | $p_{\mathrm{H}}$ |
|---|---|---|---|---|---|
| Directed area | Unrestricted | 6.74 | 0.97 | 5.77 | 0.0050 |
| Transition residual | Unrestricted | 5.70 | 0.46 | 5.24 | 0.0050 |
| Directed area | Role-preserving | 6.74 | 3.26 | 3.48 | 0.0020 |
| Transition residual | Role-preserving | 5.70 | 1.42 | 4.28 | 0.0020 |

- **NetMamba** (Wang et al., 2024): Applies a Mamba state-space model to capture long-range sequential traffic dependencies in linear time.
- **TFusion** (Fu et al., 2025): Fuses per-packet sequential features with per-flow and per-host statistical features before training lightweight downstream classifiers.
- **TrafficFormer** (Zhou et al., 2025): Represents traffic as packet or burst tokens and uses a Transformer encoder to learn contextual relationships.

#### D.1.2 Website Fingerprinting

- **Deep Fingerprinting (DF)** (Sirinam et al., 2018): A deep CNN that classifies packet-direction sequences by capturing local and hierarchical patterns.
- **AWF** (Rimmer et al., 2018): An early deep-learning baseline extracting discriminative representations directly from packet-level traffic sequences.
- **Tik-Tok** (Rahman et al., 2020): Leverages inter-packet timing patterns alongside packet ordering and direction to capture temporal characteristics.
- **Var-CNN** (Bhat et al., 2019): Uses parallel convolutional branches to model complementary views such as direction and timing for improved data efficiency.
- **NetCLR** (Bahramali et al., 2023): Employs contrastive learning and trace augmentation to learn representations that are robust to network variations.
- **TMWF** (Jin et al., 2023): Treats multi-tab recognition as ordered-set prediction and uses single-tab queries to extract website features from different trace positions.
- **ARES** (Deng et al., 2025): Formulates multi-tab fingerprinting as multi-label classification, extracts local patterns from multi-level traffic aggregation features, and models their correlations with improved self-attention.
- **BAPM** (Guan et al., 2021): Uses block-level attention to profile mixed traces for multi-tab website fingerprinting.

### D.2 Experiment Configuration

**Data splits.** All benchmarks share fixed episode splits randomly assigned within each task, and behavior primitive segments inherit their source episode's split. The episode test set contains 209 episodes: 189 benign and 20 malicious, with scenario counts of 35, 72, 60, 22, and 20 in the order listed in Section 3.1. The corrected primitive split counts are 35,400 training, 2,529 validation, and 2,101 test segments before oversampling; the reported primitive evaluation covers 2,101 test segments.

Table 11 summarizes the configurations used for risk, scenario, and primitive identification. Risk and scenario identification use two and five classes, respectively, with predictions evaluated at the episode level. Primitive identification uses 47 macro groups and is evaluated at the behavior primitive segment level.

Table 11: Main experiment configurations. R, S, and P denote risk, scenario, and primitive identification. LR and WD denote learning rate and weight decay. DIR denotes packet directions, DT denotes signed relative timestamps, and DT2 denotes separate direction and interarrival-time channels. Lengths refer to model inputs. RF denotes random forest; $n$ and $d$ denote its number of trees and maximum depth.

| Baseline | Task | Input / length | Batch | LR | Optimizer | Epochs | Additional settings |
|---|---|---|---|---|---|---|---|
| FlowLens | R, S | 4 flow statistics | – | – | RF | – | $n = 100$, unrestricted depth |
| YaTC | R, S | $40 \times 40$ bytes | 256 | $2.00 \times 10^{-3}$ | AdamW | 200 | WD=0.05; warmup=20 epochs |
| NetMamba | R | $40 \times 40$ bytes | 128 | $1.335 \times 10^{-3}$ | AdamW | 40 | WD=0.05; warmup=20 epochs |
| NetMamba | S | $40 \times 40$ bytes | 3072 | $1.068 \times 10^{-2}$ | AdamW | 40 | WD=0.05; warmup=20 epochs |
| TFusion | R | Segment length 200 | 200 | $2.259 \times 10^{-5}$ | Adam | 15 | WD=$3.293 \times 10^{-5}$; RF: $n = 200$, $d = 10$ |
| TFusion | S | Segment length 100 | – | – | RF | – | Frozen encoder; $n = 20$, $d = 50$ |
| TrafficFormer | R | 256 tokens | 32 | $2.033 \times 10^{-5}$ | AdamW | 4 | Dropout=0.455; warmup=5.18% |
| TrafficFormer | S | 320 tokens | 96 | $4.543 \times 10^{-5}$ | AdamW | 4 | Dropout=0.5; warmup=10% |
| DF | R, S, P | DIR / 5,000 | 128 | $2 \times 10^{-3}$ | Adamax | 30 | WD=0 |
| AWF | R, S, P | DIR / 3,000 | 256 | $8 \times 10^{-4}$ | RMSprop | 30 | WD=0 |
| TikTok | R, S, P | DT / 5,000 | 128 | $2 \times 10^{-3}$ | Adamax | 30 | WD=0 |
| VarCNN | R, S, P | DT2 / 5,000 | 50 | $10^{-3}$ | Adam | 30 | WD=0 |
| NetCLR | R, S, P | DIR / 5,000 | 256 | $3 \times 10^{-4}$ | Adam | 30 | WD=0 |
| TMWF | R, S, P | DIR / 30,720 | 80 | $5 \times 10^{-4}$ | Adam | 30 | WD=0 |
| ARES | R | MTAF / $8 \times 8000$ | 128 | $5 \times 10^{-4}$ | AdamW | 40 | WD=0.01; 20 ms bins |
| ARES | S, P | MTAF / $8 \times 8000$ | 512 | $2 \times 10^{-3}$ | AdamW | 30 | WD=0.01; 20 ms bins |
| BAPM | R, S, P | DIR / 8,500 | 128 | $5 \times 10^{-4}$ | Adam | 30 | WD=0 |

For YaTC and NetMamba, LR is the base absolute learning rate before warmup and decay. TFusion's risk batch size and epoch count apply to encoder training. TrafficFormer uses AdamW for risk identification; its four epochs indicate the training budget. Dashes denote inapplicable settings. Numerical hyperparameters are rounded where necessary.

The eight WF baselines use packet sequences, truncated or zero-padded to the lengths in Table 11. ARES converts the first 10,000 packets into eight-channel temporal features using 20 ms bins. TMWF uses a model input length of 30,720, although its risk inputs retain only the first 10,000 packets before padding. We train the WF models with cross-entropy loss and fixed learning rates. All checkpoints are selected by validation accuracy. For primitive identification, training classes containing fewer than 200 segments are oversampled to 200 by random sampling with replacement within each class; validation and test sets retain their original distributions.

FlowLens operates on packet count, total bytes, median interarrival time, and flow duration. YaTC and NetMamba are fine-tuned from pretrained checkpoints, with weight decay 0.05, layer-wise learning-rate decay 0.75, and label smoothing 0.1. TFusion's risk encoder is trained contrastively on benign flows, followed by a random forest with balanced class weights and a minimum leaf size of five. For scenario identification, TFusion freezes an encoder previously trained on benign ANT flows and trains only the five-class random forest. TrafficFormer uses task-specific sequence lengths, learning rates, and dropout rates, as listed in the table.

For episode-level evaluation of encrypted traffic classification baselines, we select the aggregation strategy with the highest validation-set accuracy among `mean_soft`, `majority_vote`, `confidence_weighted_mean`, and `max_confidence`. For scenario identification, flow-level class probabilities are averaged within each episode before taking the argmax. Risk identification uses the same aggregation for FlowLens and YaTC, and majority voting for NetMamba, TFusion, and TrafficFormer.