CYBERCLEAR: A BENCHMARK FOR LLM AGENT SYSTEMS ON APT ATTACK CHAIN PROVENANCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents have demonstrated promising capabilities in cybersecurity tasks, yet their ability to reconstruct complete Advanced Persistent Threat (APT) attack campaigns from complex security logs remains largely unexplored. Existing cybersecurity benchmarks for agents mainly focus on vulnerability discovery, exploitation, and security analysis tasks, leaving the evaluation of attack chain provenance under realistic security logs insufficiently studied. To address this gap, we introduce CyberClear, a benchmark for evaluating LLM agents and advanced agent systems on APT attack chain provenance from long-context security logs. CyberClear covers both single-step attacks and multi-stage attack chains, requiring agents to identify attack evidence, infer attack progression, and generate provenance graphs containing entities, causal relationships, MITRE ATT&CK techniques, and forensic evidence. To enable comprehensive evaluation, we develop an evaluation method tailored to APT attack chain provenance. Unlike conventional text similarity metrics that focus on surface-level matching, our evaluation examines whether reconstructed graphs preserve the semantics of attack chains across single-step behavior correctness, multi-step behavior identification, temporal and causal consistency, entity and relationship fidelity, and overall attack narrative consistency. Advanced multi-agent systems powered by state-of-the-art LLMs still struggle on CyberClear, motivating us to propose CyberProvenance, an agent cyber harness designed for multi-agents that augments LLM agents with evidence accumulation, execution-based validation, and feedback-guided refinement mechanisms for reliable attack-chain provenance. Extensive evaluations on CyberClear demonstrate the effectiveness of CyberProvenance in improving evidence reasoning, execution-grounded validation, and complete APT attack chain reconstruction. Anonymous Benchmark [1]

1 INTRODUCTION

Large language model (LLM) agents are taking on increasingly complex tasks through multi-step reasoning, tool use, and interaction with external environments (Yao et al., 2022; Schick et al., 2023; Wang et al., 2025). Their capabilities have extended to cybersecurity, including vulnerability discovery, exploitation, and security analysis (Deng et al., 2023; Fang et al., 2024). As agents assume a greater role in security work, comprehensive benchmarks are needed to evaluate their capabilities under realistic conditions.

Existing cybersecurity benchmarks evaluate agents through classic capture-the-flag (CTF) challenges (Shao et al., 2025; Zhang et al., 2025a) and vulnerabilities drawn from real software projects (Carlini et al., 2025; Zhu et al., 2025; Zhang et al., 2026; Lee et al., 2025; Wang et al., 2026). These benchmarks test whether agents can solve security challenges, discover or reproduce vulnerabilities, and repair vulnerable code. However, these evaluations do not directly assess agents' ability to reason about attack progression and reconstruct complete attack chains.

Advanced persistent threat (APT) attack chain provenance addresses this task by correlating security evidence to recover how an attack campaign unfolded (Li et al., 2021; Zipperle et al., 2022; Zhang et al., 2025b). This task presents three main challenges: (i) identifying relevant evidence

[1] The benchmark is partially available and will be fully released.

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.

from long and noisy security logs; (ii) establishing relationships among processes, files, users, and network activities; and (iii) inferring dependencies and temporal progression across multiple attack stages (Phan & Bauschert, 2026; Zhang et al., 2025c). However, existing evaluations largely focus on attack detection or investigation guided by prior attack clues, and lack a unified setting for assessing whether agents can independently discover dispersed evidence and reconstruct complete attack chains without prior attack knowledge.

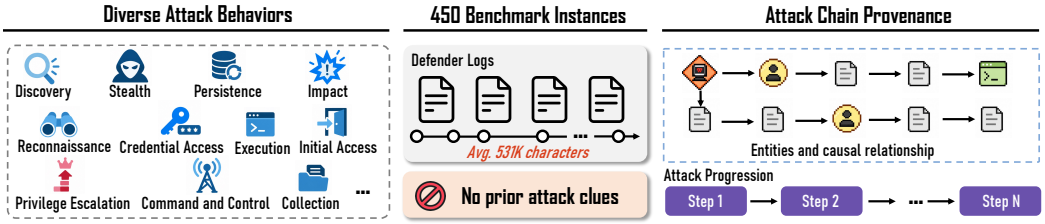


Figure 1: Overview of CyberClear. Given long-context defender logs without prior attack clues, agents are required to identify attack evidence, infer attack progression, and reconstruct complete APT attack chains across multiple attack stages.

To address this gap, we introduce CyberClear, a benchmark for evaluating LLM agents and advanced agent systems on APT attack chain provenance. As illustrated in Figure 1, CyberClear evaluates whether agents can reconstruct complete APT attack chains from long-context defender observations without prior attack clues. CyberClear contains 450 instances covering both single-step attacks and multi-stage attack chains, with each instance containing an average of 531K characters of security logs. Agents are required to identify attack evidence, infer attack progression, and generate provenance graphs containing attack entities, causal relationships, MITRE ATT&CK techniques, and forensic evidence. These elements identify the involved entities, executed actions, and attack strategies, enabling the complete reconstruction of APT attack chains. Since provenance graphs describing the same attack may differ in representation, we develop a DOT-based LLM judge to evaluate the semantic quality of reconstructed attack chains. The proposed evaluation method assesses generated graphs from five perspectives: single-step behavior correctness, multi-step behavior identification, temporal and causal consistency, entity and relationship fidelity, and overall attack narrative consistency, while ignoring irrelevant differences in graph identifiers and visual styles.

Our evaluation reveals that current agents still face significant challenges in long-context evidence reasoning, causal attack progression inference, and complete APT attack chain reconstruction. To address these challenges, we propose CyberProvenance, an agent cyber harness for improving the reliability of LLM agents in reconstructing attack chains from long-context security logs in CyberClear. The harness accumulates evidence across logs, validates predicted attack behaviors in isolated execution environments, and refines inconsistent steps using execution feedback and log evidence. We evaluate CyberProvenance alongside representative agent frameworks on CyberClear using the proposed metrics. Experimental results demonstrate that CyberProvenance effectively performs continuous validation of attack behaviors and reconstructs complete attack chains.

Our main contributions are as follows:

- We introduce CyberClear, a comprehensive benchmark for evaluating LLM agents and advanced agent systems on APT attack chain provenance from long-context defender logs without supplied attack clues.
- We develop a semantic evaluation method that assesses reconstructed provenance graphs from multiple perspectives, including single-step behavior correctness, multi-step behavior identification, temporal and causal consistency, entity and relationship fidelity, and overall attack consistency.
- We propose CyberProvenance, an agent cyber harness designed for multi-agents, which augments LLM agents with evidence accumulation, execution-based validation, and feedback-guided refinement.
- We conduct extensive evaluations of representative agent frameworks and LLM backbones on CyberClear, showing that the primary limitations of current agents for APT analysis lie

not merely in long context reasoning, but in iterative refinement, evidence consistency, and the effective use of cybersecurity specific verification tools.

2 Related Work

APT Attack Chain Provenance. APT attack chain provenance aims to reconstruct complete attack campaigns by tracing and correlating suspicious activities from large volumes of security logs (Li et al., 2021; Zipperle et al., 2022; Zhang et al., 2025b). Unlike traditional intrusion detection (Du et al., 2017; Mirsky et al., 2018; Pendlebury et al., 2019) that only identifies isolated malicious events, attack chain provenance focuses on discovering the relationships among entities, behaviors, and attack stages to recover the evolution of multi-step attacks (Phan & Bauschert, 2026; Zhang et al., 2025c). Early methods such as SLEUTH (Hossain et al., 2017), HOLMES (Milajerdi et al., 2019), and CONAN (Xiong et al., 2020) rely on predefined expert rules or TTP mappings to identify suspicious activities, but their limited rule coverage often leads to a large number of false positives. Subsequent graph learning-based approaches (Li et al., 2023b; Rehman et al., 2024; Cheng et al., 2024) leverage provenance graphs to capture complex dependencies among system entities for improved detection, but provide either insufficient contextual information or overly coarse-grained results for attack investigation. With the rapid development of LLMs, recent studies have increasingly applied LLM-based agents to attack provenance tasks (Mukherjee & Kantarcioglu, 2025; Yan et al., 2026; Zhao et al., 2026; Ock et al., 2026), enabling more flexible reasoning over security logs and attack behaviors. However, these approaches either identify only entity relationships associated with attack indicators (Mukherjee & Kantarcioglu, 2025; Ock et al., 2026) or rely on predefined attack clues during evaluation (Yan et al., 2026; Zhao et al., 2026), making them difficult to apply to realistic APT attack chain provenance scenarios where the complete attack process must be reconstructed from security logs.

Table 1: Comparison with existing cybersecurity benchmarks for Agents.

Benchmark	Scope	Instance	Attack Behaviors	ATT&CK	Logs	Attack Chain
NYU CTF Bench (Shao et al., 2025)	CTF	200	6	✗	✗	✗
Cybench (Zhang et al., 2025a)	CTF	40	6	✗	✗	✗
AutoAdvExBench (Carlini et al., 2025)	CTF + Real-world	75	2	✗	✗	✗
CVE-Bench (Zhu et al., 2025)	Real-world	40	8	✗	✗	✗
BountyBench (Zhang et al., 2026)	Real-world	40	36	✗	✗	✗
SEC-bench (Lee et al., 2025)	Real-world	200	16	✗	✗	✗
CyberGym (Wang et al., 2026)	Real-world	1,507	28	✗	✗	✗
DiagChain-Bench (Liu et al., 2026)	Real-world	69	N/R	✗	✓	✓
CyberClear	**Real-world**	**450**	**89**	✓	✓	✓

Cybersecurity Benchmarks for Agents. The increasing adoption of agents in cybersecurity has motivated the development of benchmarks for evaluating their autonomous security capabilities (Lin et al., 2026; Sanz-Gómez et al., 2025; Carlini et al., 2025). As shown in Table 1, existing benchmarks mainly focus on offensive security tasks, including CTF-based problem solving and real-world vulnerability assessment, such as Cybench (Zhang et al., 2025a), CVE-Bench (Zhu et al., 2025), BountyBench (Zhang et al., 2026), and CyberGym (Wang et al., 2026). While these benchmarks provide realistic environments for evaluating agent capabilities in vulnerability discovery, exploitation, and remediation, they primarily focus on software vulnerability-related tasks, overlooking whether agents can analyze attack evidence from security logs and reason over provenance information to reconstruct complete attack chains. To address this gap, we introduce a dedicated benchmark, CyberClear, that better emulates realistic APT provenance scenarios featuring long-context security logs and complex multi-stage attacks. These characteristics make CyberClear substantially more challenging for existing agents, as evidenced by our evaluation results in Section 5.

3 CyberClear Benchmark

3.1 Task Formulation

We formulate CyberClear as an APT attack chain provenance task that requires agents to reconstruct complete attack processes from large-scale security logs. Given a collection of defender logs $\boldsymbol{L} = \{L_1, L_2, \ldots, L_n\}$ collected during an attack campaign, the agent performs provenance analysis without access to prior attack clues. The agent is required to identify attack-related entities, infer

their relationships, and reconstruct the complete attack process from dispersed log evidence. The final output is a provenance graph $\boldsymbol{G} = (E, R)$, where E represents attack-related entities such as processes, accounts, files, services, and network resources, while R captures their interactions and dependencies. Beyond entity relationships, the graph organizes the reconstructed attack process into an attack narrative timeline, where each step contains its attack behavior, corresponding MITRE ATT&CK techniques (Jiang et al., 2025), and supporting forensic evidence.

3.2 Benchmark Construction

Figure 2 presents an overview of our four-stage data construction pipeline, including attack evidence extraction, provenance graph generation, automatic verification, and human expert review. Each sample consists of defender logs collected from a continuous attack campaign and a provenance graph that represents relationships among attack entities and the evolution of multi-stage attacks.

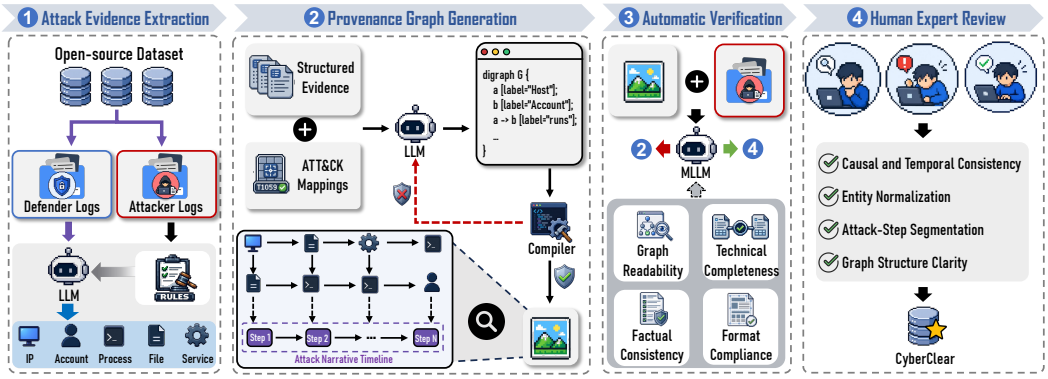


Figure 2: The construction pipeline of CyberClear.

Stage I: Attack Evidence Extraction. We collect raw attack logs from publicly available APT datasets, including PROVCON (Yusof et al., 2025) and CAM-LDS (Landauer et al., 2026), which are constructed from real-world attack scenarios. While designed to support APT behavior analysis and provenance research, these datasets primarily focus on attack detection and security log analysis. We design attack evidence extraction rules based on attacker logs. For each sample, we provide Claude Opus 4.8 (Anthropic, 2026) with extraction rules to extract attack evidence, including key attack entities, temporal events, entity interactions, step-level attack descriptions, and corresponding MITRE ATT&CK techniques. The extracted attack evidence is further organized into structured representations for subsequent processing.

Stage II: Provenance Graph Generation. We employ Claude Opus 4.8 to generate Graphviz DOT code for each sample based on the extracted attack evidence and MITRE ATT&CK annotations. To ensure consistent entity representation in generated DOT graphs, we define unified graph representations that assign predefined node formats to different entity types, including network entities, accounts, processes, files, and services. The generated DOT code is compiled into visual provenance graphs, and failed compilations are returned to the model for iterative correction. As shown in Figure 2 (2), the resulting provenance graphs preserve entity relationships, and summarize the evolution of multi-stage attacks through attack timelines.

Stage III: Automatic Verification. The generated graphs only require evidence consistency verification, so we employ GPT-4o (Hurst et al., 2024) to automatically verify them using attacker logs as the ground truth. The verification process consists of four sequential checks: (1) **Graph readability**, ensuring that the generated graphs are clear, complete, and visually understandable; (2) **Technical completeness**, verifying that all required ATT&CK techniques are correctly included; (3) **Factual consistency**, checking whether the reconstructed entities, relationships, and attack timeline are consistent with the actual attack behaviors; and (4) **Format compliance**, ensuring that the generated graphs follow predefined visualization and representation requirements. If any verification criterion fails, the feedback generated by GPT-4o is sent back to step 2 for graph regeneration.

Stage IV: Human Expert Review. To further ensure dataset reliability, security experts manually review the provenance graphs that pass automatic verification. The experts evaluate the generated graphs based on predefined quality criteria and filter out low quality samples from open-source datasets, including incomplete attack traces, insufficient evidence coverage, and inconsistent provenance information. Only samples that satisfy both automatic verification and expert review are included in CyberClear.

3.3 Data Analysis

Dataset Scale. CyberClear contains 450 APT attack chain provenance samples with approximately 2.1GB of data, including 318 easy instances, 122 medium instances, and 10 hard instances categorized by attack complexity. Table 2 summarizes the dataset statistics, including sample distribution, attack complexity, input log volume, and log length. The easy subset consists entirely of single-step attacks, while medium and hard subsets contain multi-step attack chains requiring models to reason over dependencies among multiple attack stages. Compared with easy and medium instances, hard samples involve substantially larger input log volumes, with an average of 83.2 log files per sample and longer contexts containing approximately 7.80×10^5 characters, providing a more challenging setting for evaluating APT attack chain provenance capabilities.

Table 2: Statistics of the CyberClear.

Statistic	Easy	Medium	Hard	Overall
Total Samples				
Sample Count	318	122	10	450
Attack Complexity				
Single-step Attack	318	0	0	318
Multi-step Attack	0	122	10	132
Input Log Volume (count)				
Minimum	1	1	79	–
Maximum	12	24	92	–
Average	3.26	5.99	83.2	5.78
Input Log Length (characters)				
Minimum	336	1167	1,222	–
Maximum	5.18e6	5.44e6	5.76e7	–
Average	7.07e5	6.69e5	7.80e5	5.31e5

Attack Behaviors. Figure 3 shows the distribution of attack behaviors in CyberClear. The benchmark covers 89 distinct attack behaviors spanning Credential Access, Defense Evasion, Persistence, Execution, Impact, and other attack categories. Discovery and Reconnaissance are the two most frequent categories, accounting for 24.00% and 15.56% of all instances, respectively. Among them, Composite Attacks are constructed by combining multiple attack behaviors, accounting for 6.44% of all instances. This diverse distribution provides broad coverage of APT behaviors across different attack stages and enables comprehensive evaluation of attack chain provenance capabilities.

3.4 Evaluation Metrics

Code-Level Evaluation. The generated DOT code is evaluated from both textual similarity and functional correctness perspectives. BLEU and ROUGE-L are used to measure the similarity between generated and reference DOT code, capturing token-level overlap and sequence-level consistency, respectively. CodeBLEU is further adopted to evaluate structural consistency by considering code syntax and semantic information beyond surface-level matching. Pass@1 is used to assess whether the generated DOT code can be successfully compiled, reflecting practical generation reliability.

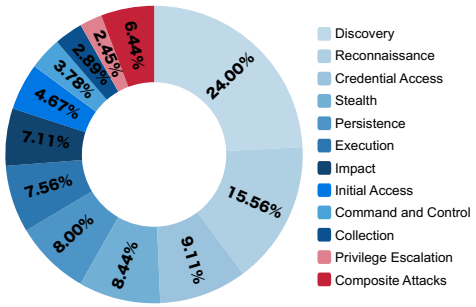


Figure 3: Distribution of attack behaviors.

Semantic Evaluation via LLM-as-a-Judge. To complement lexical matching and code structure metrics, we develop a DOT-based LLM judge to evaluate the semantic quality of reconstructed attack chains. The judge directly compares reference and predicted DOT code while ignoring irrelevant differences in node identifiers, statement order, graph layouts, and visual styles. We use GLM-5.2 (Zeng et al., 2026) as the judge model to perform evidence-based semantic comparison between reference and predicted provenance graphs. The semantic evaluation framework assesses reconstructed provenance graphs from five perspectives. Each dimension is scored from 0 to 1, and the reported scores are averaged over all samples. Detailed illustrations are in Appendix C.

- **Single-Step Judge.** Evaluates whether individual attack behaviors are correctly reconstructed by comparing the actor, core action, target, and overall attack meaning.

- **Loose Judge.** Measures whether multi-step attack behaviors are correctly identified without considering temporal ordering among attack steps.
- **Strict Judge.** Evaluates whether the reconstructed attack chain preserves the temporal progression and causal dependencies among different attack stages.
- **Detail Judge.** Measures the consistency of graph-level elements, including entities, directed relationships, and attributes between reference and predicted provenance graphs.
- **Scenario Judge.** Assesses whether the reconstructed graph conveys the same overall attack process, target, method, progression, and outcome as the reference attack campaign.

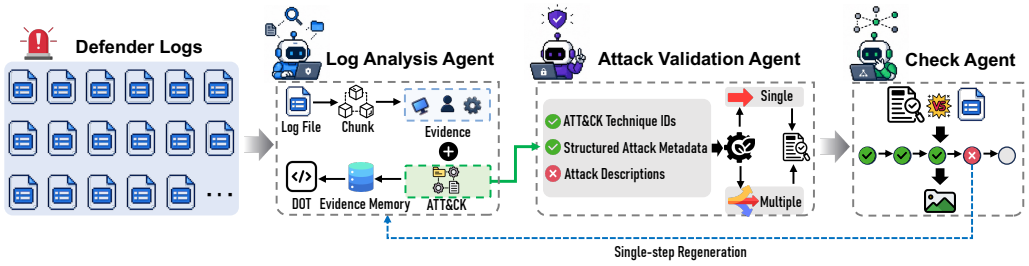


Figure 4: Overview of CyberProvenance, an agent cyber harness for reliable attack-chain provenance. Designed for multi-agent collaboration, the harness assigns specialized capabilities to different agents, enabling evidence accumulation, execution-based validation, and feedback-guided refinement.

4 METHODOLOGY

We present CyberProvenance, an agent cyber harness designed for multi-agent collaboration to improve the reliability of LLM-based agents. The harness assigns specialized capabilities to different agent roles, including Log Analysis Agent, Attack Validation Agent, and Check Agent, which collaboratively perform evidence accumulation, execution-based validation, and iterative provenance graph refinement, as illustrated in Figure 4.

Log Analysis Agent. To identify and accumulate attack evidence from large volumes of logs and establish an initial provenance structure, the Log Analysis Agent is responsible for evidence accumulation in the CyberProvenance harness. It receives defender security logs $\boldsymbol{L} = \{L_1, \ldots, L_n\}$ and progressively analyzes logs to identify attack related evidence, including temporal events, processes, commands, network activities, and potential MITRE ATT&CK techniques. To handle long context logs, each log file L_i is divided into multiple chunks $\{C_i^1, \ldots, C_i^{K_i}\}$, and the extracted evidence from each chunk is continuously accumulated into an Evidence Memory $\boldsymbol{M}$. Based on the accumulated evidence, the agent performs global reasoning to generate an initial provenance hypothesis:

$$\boldsymbol{H}_0 = \mathcal{F}(\boldsymbol{M}) = (\boldsymbol{A}, \boldsymbol{S}, \boldsymbol{T}, \boldsymbol{G}_0),$$

where $\boldsymbol{A}$ denotes the attack scenario, $\boldsymbol{S}$ represents attack stages and step dependencies, $\boldsymbol{T}$ denotes ATT&CK technique mappings, and $\boldsymbol{G}_0$ represents the reconstructed provenance graph. The reconstructed provenance graph $\boldsymbol{G_0}$ is finally serialized into structured DOT code, providing an explicit graph representation for subsequent attack validation.

Attack Validation Agent. The Attack Validation Agent provides execution-grounded validation by verifying the executability and reliability of reconstructed attack steps through real environment reproduction. This reduces the risk of relying on plausible but unverified attack inferences. Given the predicted ATT&CK techniques $\boldsymbol{T} = \{\tau_1, \ldots, \tau_m\}$ and structured attack steps generated by the Log Analysis Agent, the agent retrieves the corresponding execution specifications for each technique:

$$q_i = \mathcal{R}(\tau_i) = (p_i, x_i, r_i, d_i),$$

where p_i, x_i, r_i, and d_i represent the required platform, executor, privilege, and dependency information of technique τ_i, respectively. Using these specifications, the agent selects appropriate execution strategies, where independent techniques are reproduced using Atomic Red Team (Red Canary) and dependency aware multi step attack chains are orchestrated through CALDERA (Apache Software Foundation) in isolated environments built with Ludus (Bad Sector Labs). The generated execution traces collected by Sysmon (Microsoft), Linux Audit (Linux Audit Project), and Zeek (Zeek

Project) are used to produce structured validation reports, indicating whether each reconstructed attack step can be successfully reproduced.

Check Agent. The Check Agent closes the refinement loop of the harness by integrating the reconstructed provenance graph and validation feedback to assess reliability through graph structures, attack-step dependencies, and ATT&CK technique mappings. When all attack steps are successfully reproduced and consistent with the collected evidence, the current provenance graph is preserved as the final result. Otherwise, unreliable steps are refined while preserving validated nodes, and the updated hypothesis is revalidated to form an iterative prediction, validation, and refinement loop.

Table 3: Performance comparison on CyberClear across agent frameworks and LLM backbones using code-level and semantic metrics. Blue cells indicate the best result for each metric.

Methods	**Code-Level Metrics**				**Semantic Metrics**				
	BLEU	ROUGE-L	CodeBLEU	Pass@1	Single	Loose	Strict	Detail	Scenario
OpenClaw (Single)	0.3317	0.4863	0.3795	0.8222	0.6794	0.6006	0.4387	0.4051	0.4805
	0.3556	0.5117	0.3868	0.7200	0.7812	0.7125	0.5490	0.4898	0.5823
	0.3692	0.5297	0.3887	0.8467	0.7726	0.7214	0.5008	0.4536	0.5862
OpenClaw (Multiple)	0.4183	0.5654	0.5648	0.9200	0.5513	0.4749	0.3580	0.3301	0.3679
	0.4422	0.5908	0.5721	0.8178	0.7160	0.6629	0.5034	0.4090	0.5847
	0.4516	0.5848	0.5643	0.9333	0.7136	0.6684	0.5178	0.4053	0.5821
AgentVerse	0.3664	0.5446	0.6074	0.9400	0.5080	0.4337	0.3273	0.3039	0.3269
	0.3903	0.5700	0.6147	0.8378	0.7304	0.6743	0.5150	0.4248	0.5886
	0.4015	0.5942	0.6027	0.9422	0.7113	0.6897	0.5011	0.4179	0.6135
LLM-Debate	0.4089	0.5278	0.6502	0.8644	0.6135	0.5344	0.4089	0.3758	0.4334
	0.4328	0.5532	0.6575	0.7622	0.7622	0.6853	0.5368	0.4581	0.6061
	0.4413	0.5789	0.6494	0.8844	0.7436	0.7082	0.5216	0.4704	0.6047
MultiPersona	0.3775	0.5570	0.4217	0.9289	0.6843	0.6033	0.4418	0.4037	0.4842
	0.4014	0.5824	0.4290	0.8267	0.7800	0.7107	0.5519	0.4825	0.5961
	0.4136	0.5314	0.4247	0.9422	0.7131	0.7198	0.5289	0.5002	0.5235
LLM-Blender	0.4196	0.5495	0.5761	0.8733	0.5632	0.4869	0.3660	0.3384	0.3789
	0.4435	0.5749	0.5834	0.7711	0.7216	0.6668	0.5077	0.4151	0.5864
	0.4519	0.5976	0.6112	0.8956	0.7317	0.6761	0.5248	0.4465	0.5969
DyLAN	0.3497	0.5241	0.5099	0.8844	0.8454	0.7514	0.5421	0.4769	0.6223
	0.3808	0.5528	0.5491	0.8667	0.8511	**0.7705**	0.6224	0.5433	0.6545
	0.3884	0.5258	0.5375	0.8711	0.8350	0.7502	0.5592	0.5214	0.6771
MacNet	0.3633	0.4728	0.5326	0.9267	0.6186	0.5590	0.3928	0.3977	0.4241
	0.3630	0.4784	0.5422	0.7889	0.7708	0.7129	0.5226	0.5090	0.5061
	0.3389	0.4515	0.5086	0.8088	0.7375	0.6809	0.4758	0.5070	0.5022
CAMEL	0.3577	0.5647	0.5629	0.9222	0.3754	0.3048	0.2371	0.2217	0.2043
	0.4057	0.5869	0.5790	0.7978	0.6710	0.6322	0.4673	0.3618	0.5656
	0.3864	0.5781	0.4201	0.9377	0.5371	0.4557	0.2955	0.3025	0.4117
AgentScope-V2	0.4921	0.6477	0.6805	0.8622	0.8412	0.7373	0.6291	0.5485	0.7021
	0.4977	0.6523	0.6854	0.7889	0.8343	0.6786	0.5864	0.5169	0.6936
	0.5035	0.6334	**0.7240**	0.9866	0.8525	0.6597	0.6106	0.5710	0.7115
CyberProvenance (Ours)	0.4971	0.6873	0.6867	**0.9933**	0.8627	0.7325	0.6340	0.5981	0.6710
	0.4810	0.6827	0.7052	0.8911	0.8764	0.7027	**0.6477**	0.5776	0.7087
	0.4952	**0.6968**	0.7016	0.9333	**0.8897**	0.7438	0.6337	**0.6124**	**0.7263**

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

LLMs. We evaluate DeepSeek-V4-Flash (Xu et al., 2026), Qwen3.8-Max (Qwen Team, 2026), and GPT-5.6-Terra (OpenAI, 2026) in the main experiments. These models exhibit different reasoning and generation characteristics, allowing us to examine the performance of agent frameworks across different LLM backbones. Additional results with GPT-4o-mini (Hurst et al., 2024) and Gemini-3.5-Flash-Lite (Google DeepMind, 2026) are reported in Appendix A.1.

Methods. We compare CyberProvenance against two categories of baselines: (1) **Single-Agent Methods** including OpenClaw (Single-Agent) (OpenClaw); (2) **Multi-Agent Systems** including OpenClaw (Multi-Agent), AgentVerse (Chen et al., 2024), LLM-Debate (Du et al., 2023), MultiPersona (Wang et al., 2024), LLM-Blender (Jiang et al., 2023), DyLAN (Liu et al., 2024), MacNet (Qian et al., 2025), CAMEL (Li et al., 2023a), and AgentScope-V2 (Dawei Gao, 2024; 2025). Additional details are provided in Appendix B.1.

5.2 MAIN RESULTS

Code-Level Metrics. Table 3 shows that existing agent frameworks can generate valid provenance graphs, but their performance varies markedly across methods and LLM backbones, indicating that stable attack-graph reconstruction remains difficult. AgentScope-V2 achieves the highest BLEU and CodeBLEU scores of 0.5035 and 0.7240, while CyberProvenance obtains the best ROUGE-L score of 0.6968 and Pass@1 score of 0.9933. Compared with general-purpose agent frameworks, CyberProvenance maintains consistently strong code-level performance across multiple metrics. These results suggest that iterative refinement and cybersecurity-specific validation tools contribute to more reliable provenance graph reconstruction.

Semantic Metrics. The challenge is more evident under semantic metrics, where most existing frameworks struggle to preserve complete attack progression, entity dependencies, and fine-grained evidence. Although DyLAN achieves the best Loose score of 0.7705, its advantage does not extend consistently to stricter semantic criteria. CyberProvenance achieves the highest Single, Strict, Detail, and Scenario scores of 0.8897, 0.6477, 0.6124, and 0.7263, respectively. The remaining gap on Strict and Detail further indicates that complete APT attack-chain reconstruction is still challenging for current agent systems.

5.3 PERFORMANCE ANALYSIS ACROSS DIFFICULTY LEVELS

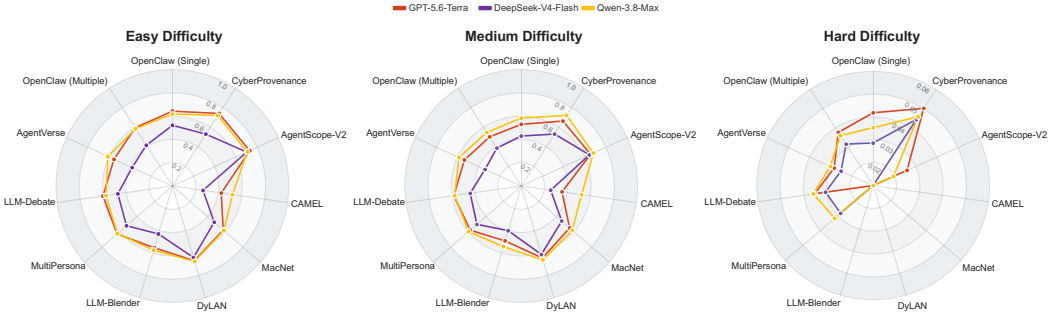


Figure 5: Average Semantic Metrics scores of different agent frameworks across the Easy, Medium, and Hard subsets of CyberClear.

We evaluate different agent frameworks across the Easy, Medium, and Hard subsets of CyberClear using the average Semantic Metrics score over Single, Loose, Strict, Detail, and Scenario. As shown in Figure 5, all frameworks exhibit consistent performance degradation as the number of input log files increases rapidly. This trend indicates that larger log volumes make it increasingly difficult to identify relevant evidence and maintain coherent attack progression during provenance. These results demonstrate that CyberClear effectively captures the increasing difficulty of APT attack chain reconstruction across different log scales.

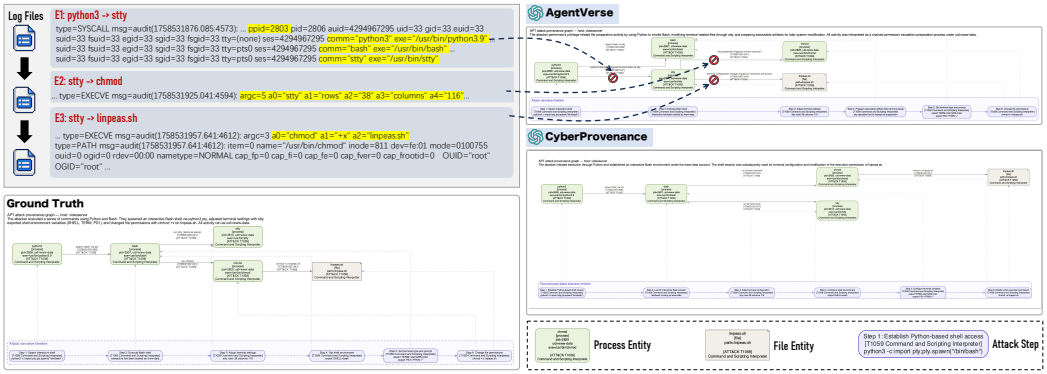


Figure 6: Case study comparing AgentVerse and CyberProvenance on attack provenance using GPT-5.6-Terra. CyberProvenance corrects unsupported causal relations by aligning the reconstructed graph with execution evidence.

5.4 Performance Analysis Across Attack Stage Levels

Attack stage complexity exposes a clear performance gap in current agent frameworks. As shown in Figure 7, we report the average Semantic Metrics score across Single, Loose, Strict, Detail, and Scenario, where nearly all methods perform substantially worse on multi-stage attacks than on single-stage attacks across the three LLM backbones. The decline is especially pronounced for CAMEL, MacNet, and LLM-Debate, whereas CyberProvenance remains more robust and consistently achieves stronger multi-stage performance. These results highlight the importance of iterative validation and evidence-guided refinement for maintaining coherent attack chain reconstruction as attack complexity increases.

	GPT-5.6-Terra		DeepSeek-V4-Flash		Qwen-3.8-Max	
	Single	Multi	Single	Multi	Single	Multi
OpenClaw (Single)	0.60	0.42	0.58	0.40	0.57	0.45
OpenClaw (Multiple)	0.63	0.56	0.62	0.53	0.64	0.56
AgentVerse	**0.68**	**0.68**	**0.66**	**0.66**	**0.70**	**0.70**
LLM-Debate	**0.74**	0.54	**0.69**	0.59	**0.71**	0.52
MultiPersona	0.64	0.45	0.62	0.42	0.64	0.46
LLM-Blender	**0.72**	0.57	**0.67**	0.57	**0.70**	0.61
DyLAN	**0.84**	0.64	**0.85**	0.61	**0.85**	**0.66**
MacNet	**0.77**	0.55	0.61	0.46	**0.74**	0.58
CAMEL	**0.65**	0.35	0.63	0.26	**0.67**	0.52
AgentScope-V2	**0.85**	**0.66**	**0.84**	0.64	**0.84**	**0.68**
CyberProvenance	**0.87**	**0.73**	**0.84**	**0.69**	**0.85**	**0.73**

0.0–0.2 0.2–0.4 0.4–0.6 0.6–0.8 0.8–1.0

Figure 7: Average Semantic Metrics scores across single-stage and multi-stage attacks.

5.5 Case Study

Figure 6 presents a representative example using GPT-5.6-Terra, where the ground-truth graph shows that `python3` launches `bash`, which subsequently invokes both `stty` and `chmod`, while `chmod` modifies `linpeas.sh`. AgentVerse incorrectly associates `python3` directly with `stty`, although the audit log shows that `stty` is spawned by the Bash process rather than Python. It further introduces dependencies from `stty` to `chmod` and `linpeas.sh`, even though `chmod` is independently invoked by Bash and the permission change on `linpeas.sh` is directly caused by `chmod +x`. In contrast, CyberProvenance removes these unsupported dependencies and reconstructs the causal structure more consistently with the ground truth. More cases in the Appendix E.

6 Conclusion

In this work, we introduce CyberClear, a benchmark for evaluating LLM agents on APT attack chain provenance from long-context security logs. CyberClear covers both single-step and multi-stage attacks and provides a semantic evaluation method that assesses attack behaviors, temporal and causal consistency, entity relationships, and overall attack narrative consistency. Extensive evaluations show that even advanced agent systems still face substantial challenges in evidence reasoning and complete attack chain reconstruction. To address these limitations, we propose CyberProvenance, demonstrates the effectiveness of an agent cyber harness that combines evidence accumulation, execution-based validation, and feedback-guided refinement for reliable APT attack chain reconstruction. We hope CyberClear and CyberProvenance will provide a useful foundation for developing more capable and trustworthy agents for complex cybersecurity analysis.

AI USE STATEMENT

In this work, we used generative AI tools to assist with refining the research methodology and experimental design, implementing and debugging parts of the experimental code, and interpreting intermediate experimental results. We also used generative AI tools for literature search and summarization, language polishing, improving the organization and readability of the manuscript, and assisting with the preparation of figures, tables, and captions. All AI-assisted code and experimental procedures were manually reviewed and tested by the authors, and the reported results were checked against the corresponding execution outputs and evaluation scripts. Literature-related suggestions were verified against the original papers, and all AI-assisted text, figures, and technical claims were reviewed and revised by the authors before inclusion. We take full responsibility for the final content of this work, including all text, claims, code, experiments, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies defensive cybersecurity analysis and APT attack provenance. CyberClear is constructed from publicly available security datasets and does not involve human subjects or private user data collection. All execution-based validation is conducted only in authorized and isolated environments, with private network ranges and safeguards that prevent destructive or unauthorized actions against external systems. While the studied techniques are inherently dual-use, the benchmark and framework are intended for defensive analysis, provenance, and evaluation of security agents, and we will clearly document the intended use and safety constraints of the released artifacts.

REPRODUCIBILITY STATEMENT

We have taken several steps to support the reproducibility of this work. The main paper describes the benchmark construction, evaluation protocol, and overall experimental setup, while the appendix provides the complete configuration details for all evaluated agent frameworks. Additional implementation details, data processing procedures, prompts, and evaluation methods are included in the appendix. These materials are intended to provide the information necessary to reproduce the reported experiments and results.

REFERENCES

Anthropic. System card: Claude opus 4.8, 2026. URL https://www-cdn.anthropic.com/0b4915911bb0d19eca5b5ee635c80fef830a37ea.pdf.

Apache Software Foundation. Apache caldera. https://github.com/apache/caldera. GitHub repository.

Bad Sector Labs. Ludus. https://github.com/badsectorlabs/ludus. GitHub repository.

Nicholas Carlini, Javier Rando, Edoardo Debenedetti, Milad Nasr, and Florian Tramèr. Autoadvexbench: Benchmarking autonomous exploitation of adversarial example defenses. *arXiv preprint arXiv:2503.01811*, 2025.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, volume 2024, pp. 20094–20136, 2024.

Zijun Cheng, Qiujian Lv, Jinyuan Liang, Yan Wang, Degang Sun, Thomas Pasquier, and Xueyuan Han. Kairos: Practical intrusion detection and investigation using whole-system provenance. In *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 3533–3551. IEEE, 2024.

Xuchen Pan Weirui Kuang Zhijian Ma Bingchen Qian Fei Wei Wenhao Zhang Yuexiang Xie Daoyuan Chen Liuyi Yao Hongyi Peng Zeyu Zhang Lin Zhu Chen Cheng Hongzhu Shi Yaliang Li Bolin Ding Jingren Zhou Dawei Gao, Zitao Li. Agentscope: A flexible yet robust multi-agent platform. *CoRR*, abs/2402.14034, 2024.

Yuexiang Xie Weirui Kuang Liuyi Yao Bingchen Qian Zhijian Ma Yue Cui Haohao Luo Shen Li Lu Yi Yi Yu Shiqi He Zhiling Luo Wenmeng Zhou Zhicheng Zhang Xuguang He Ziqian Chen Weikai Liao Farruh Isakulovich Kushnazarov Yaliang Li Bolin Ding Jingren Zhou Dawei Gao, Zitao Li. Agentscope 1.0: A developer-centric framework for building agentic applications. *CoRR*, abs/2508.16279, 2025.

Gelei Deng, Yi Liu, Víctor Mayoral-Vilches, Peng Liu, Yuekang Li, Yuan Xu, Tianwei Zhang, Yang Liu, Martin Pinzger, and Stefan Rass. Pentestgpt: An llm-empowered automatic penetration testing tool. *arXiv preprint arXiv:2308.06782*, 2023.

Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pp. 1285–1298, 2017.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.

Richard Fang, Rohan Bindu, Akul Gupta, and Daniel Kang. Llm agents can autonomously exploit one-day vulnerabilities. *arXiv preprint arXiv:2404.08144*, 2024.

Google DeepMind. Gemini 3.5 flash model card, 2026. URL `https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-5-Flash-Model-Card.pdf`.

Md Nahid Hossain, Sadegh M Milajerdi, Junao Wang, Birhanu Eshete, Rigel Gjomemo, R Sekar, Scott Stoller, and VN Venkatakrishnan. {SLEUTH}: Real-time attack scenario reconstruction from {COTS} audit data. In *26th USENIX security symposium (USENIX Security 17)*, pp. 487–504, 2017.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14165–14178, 2023.

Yuning Jiang, Qiaoran Meng, Feiyang Shang, Nay Oo, Le Thi Hong Minh, Hoon Wei Lim, and Biplab Sikdar. Mitre att&ck applications in cybersecurity and the way forward. *arXiv preprint arXiv:2502.10825*, 2025.

Max Landauer, Wolfgang Hotwagner, Thorina Boenke, Florian Skopik, and Markus Wurzenberger. Cam-lds: Cyber attack manifestations for automatic interpretation of system logs and security alerts. *arXiv preprint arXiv:2603.04186*, 2026.

Hwiwon Lee, Ziqi Zhang, Hanxiao Lu, and Lingming Zhang. Sec-bench: Automated benchmarking of llm agents on real-world software security tasks, 2025. URL `https://arxiv.org/abs/2506.11791`.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023a.

Shaofei Li, Feng Dong, Xusheng Xiao, Haoyu Wang, Fei Shao, Jiedong Chen, Yao Guo, Xiangqun Chen, and Ding Li. Nodlink: An online system for fine-grained apt attack detection and investigation. *arXiv preprint arXiv:2311.02331*, 2023b.

Zhenyuan Li, Qi Alfred Chen, Runqing Yang, Yan Chen, and Wei Ruan. Threat detection and investigation with system-level provenance graphs: A survey. *Computers & Security*, 106:102282, 2021.

Justin Lin, Eliot Jones, Donovan Jasper, Ethan Ho, Anna Wu, Arnold Yang, Neil Perry, Andy Zou, Matt Fredrikson, Zico Kolter, et al. Comparing ai agents to cybersecurity professionals in real-world penetration testing. In *International Conference on Learning Representations*, volume 2026, pp. 1872–1902, 2026.

Linux Audit Project. Linux audit. https://github.com/linux-audit/audit-userspace. GitHub repository.

Xuyang Liu, Yibin Han, Zhenwei Zhang, Kai Chang, Zhiwei Xu, Tian Qiu, Weixian Deng, Jiabao Gao, Xiaolin Peng, Hai Wan, and Xibin Zhao. Diagchain: A diagnostic benchmark for evaluating llm agents on evidence-grounded attack chain reconstruction, 2026. URL https://arxiv.org/abs/2608.03591.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic llm-powered agent network for task-oriented agent collaboration, 2024. URL https://arxiv.org/abs/2310.02170.

Microsoft. Sysmon: System monitor. https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon.

Sadegh M Milajerdi, Rigel Gjomemo, Birhanu Eshete, Ramachandran Sekar, and VN Venkatakrishnan. Holmes: real-time apt detection through correlation of suspicious information flows. In *2019 IEEE symposium on security and privacy (SP)*, pp. 1137–1152. IEEE, 2019.

Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: An ensemble of autoencoders for online network intrusion detection. *arXiv preprint arXiv:1802.09089*, 2018.

Kunal Mukherjee and Murat Kantarcioglu. Llm-driven provenance forensics for threat investigation and detection. *arXiv preprint arXiv:2508.21323*, 2025.

Jiwon Ock, Jungsik Lee, and Changhee Choi. Distributed multi-llm provenance graph construction for scalable correlation of apt logs. In *2026 IEEE 46th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pp. 13–16. IEEE, 2026.

OpenAI. Gpt-5.6 system card. https://deploymentsafety.openai.com/gpt-5-6/gpt-5-6.pdf, 2026. OpenAI Deployment Safety Hub.

OpenClaw. OpenClaw: Personal ai assistant. https://github.com/openclaw/openclaw. Software repository.

Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. {TESSERACT}: Eliminating experimental bias in malware classification across space and time. In *28th USENIX security symposium (USENIX Security 19)*, pp. 729–746, 2019.

Trung V Phan and Thomas Bauschert. Learning the apt kill chain: Temporal reasoning over provenance data for attack stage estimation. In *ICC 2026-IEEE International Conference on Communications*, pp. 1–6. IEEE, 2026.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, et al. Scaling large language model-based multi-agent collaboration. In *International Conference on Learning Representations*, volume 2025, pp. 41488–41505, 2025.

Qwen Team. Qwen3.8-max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8.

Red Canary. Atomic red team. https://github.com/redcanaryco/atomic-red-team. GitHub repository.

Mati Ur Rehman, Hadi Ahmadi, and Wajih Ul Hassan. Flash: A comprehensive approach to intrusion detection via provenance graph representation learning. In *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 3552–3570. IEEE, 2024.

María Sanz-Gómez, Víctor Mayoral-Vilches, Francesco Balassone, Luis Javier Navarrete-Lozano, Cristóbal RJ Chavez, and Maite del Mundo de Torres. Cybersecurity ai benchmark (caibench): a meta-benchmark for evaluating cybersecurity ai agents. *arXiv preprint arXiv:2510.24317*, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Minghao Shao, Sofija Jancheska, Meet Udeshi, Brendan Dolan-Gavitt, Haoran Xi, Kimberly Milner, Boyuan Chen, Max Yin, Siddharth Garg, Prashanth Krishnamurthy, Farshad Khorrami, Ramesh Karri, and Muhammad Shafique. Nyu ctf bench: A scalable open-source benchmark dataset for evaluating llms in offensive security, 2025. URL https://arxiv.org/abs/2406.05590.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents. In *International Conference on Learning Representations*, volume 2025, pp. 65882–65919, 2025.

Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 257–279, 2024.

Zhun Wang, Tianneng Shi, Jingxuan He, Matthew Cai, Jialin Zhang, and Dawn Song. Cybergym: Evaluating ai agents' real-world cybersecurity capabilities at scale. In *International Conference on Learning Representations*, volume 2026, pp. 123341–123386, 2026.

Chunlin Xiong, Tiantian Zhu, Weihao Dong, Linqi Ruan, Runqing Yang, Yueqiang Cheng, Yan Chen, Shuai Cheng, and Xutong Chen. Conan: A practical real-time apt detection system with high accuracy and efficiency. *IEEE Transactions on Dependable and Secure Computing*, 19(1): 551–565, 2020.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Wenhao Yan, Ning An, Linxu Li, Bingsheng Bi, Bo Jiang, Zhigang Lu, Baoxu Liu, Junrong Liu, and Cong Dong. Provagent: Threat detection based on identity-behavior binding and multi-agent collaborative attack investigation. *arXiv preprint arXiv:2603.09358*, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Anis Yusof, Shaofei Li, Arshdeep Singh Kawatra, Ding Li, Ee-Chien Chang, and Zhenkai Liang. From observations to insights: Constructing effective cyberattack provenance with provcon. In *Proceedings of the Workshop on SOC Operations and Construction (WOSOC)*, 2025.

Zeek Project. Zeek: The zeek network security monitor. https://github.com/zeek/zeek. GitHub repository.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Andy Zhang, Joey Ji, Celeste Menders, Riya Dulepet, Thomas Qin, Ron Wang, Junrong Wu, Kyleen Liao, Jiliang Li, Jinghan Hu, et al. Bountybench: Dollar impact of ai agent attackers and defenders on real-world cybersecurity systems. *Advances in Neural Information Processing Systems*, 38, 2026.

Andy K Zhang, Neil Perry, Riya Dulepet, Joey Ji, Celeste Menders, Justin W Lin, Eliot Jones, Gashon Hussein, Samantha Liu, Donovan Julian Jasper, Pura Peetathawatchai, Ari Glenn, Vikram Sivashankar, Daniel Zamoshchin, Leo Glikbarg, Derek Askaryar, Haoxiang Yang, Aolin Zhang, Rishi Alluri, Nathan Tran, Rinnara Sangpisit, Kenny O Oseleononmen, Dan Boneh, Daniel E. Ho, and Percy Liang. Cybench: A framework for evaluating cybersecurity capabilities and risks

of language models. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=tc90LV0yRL.

Bo Zhang, Yansong Gao, Boyu Kuang, Changlong Yu, Anmin Fu, and Willy Susilo. A survey on advanced persistent threat detection: A unified framework, challenges, and countermeasures. *ACM Computing Surveys*, 57(3):62, 2025b.

Zhao Zhang, Senlin Luo, Yingdan Guan, and Limin Pan. Mpca: Constructing the apts provenance graphs through multi-perspective confidence and association. *Information and Software Technology*, 180:107670, 2025c. ISSN 0950-5849. doi: https://doi.org/10.1016/j.infsof.2025.107670. URL https://www.sciencedirect.com/science/article/pii/S0950584925000096.

Guangze Zhao, Yongzheng Zhang, Weilin Gai, Hongri Liu, Yuliang Wei, and Bailing Wang. Hunteragent: Neuro-symbolic attack trace reconstruction under anti-forensics. *arXiv preprint arXiv:2605.29269*, 2026.

Yuxuan Zhu, Antony Kellermann, Dylan Bowman, Philip Li, Akul Gupta, Adarsh Danda, Richard Fang, Conner Jensen, Eric Ihli, Jason Benn, et al. Cve-bench: A benchmark for ai agents' ability to exploit real-world web application vulnerabilities. *arXiv preprint arXiv:2503.17332*, 2025.

Michael Zipperle, Florian Gottwalt, Elizabeth Chang, and Tharam Dillon. Provenance-based intrusion detection systems: A survey. *ACM Computing Surveys*, 55(7):1–36, 2022.

APPENDIX OVERVIEW

This appendix provides supplementary experiments, analyses, and case studies, organized as follows:

- **Appendix A: Additional Experiments.** We report additional evaluations with different LLM backbones, conduct ablation studies, analyze consistency with human judgments, and investigate performance across difficulty and attack-stage levels.
- **Appendix B: Experimental Details.** We provide detailed descriptions of baseline configurations, validation environments, and experimental settings.
- **Appendix C: Details of Semantic Evaluation via LLM-as-a-Judge.** We describe the scoring criteria, multi-dimensional evaluation protocol, and prompts used for semantic assessment.
- **Appendix D: Attack Behavior Taxonomy and Boundaries.** We present the complete attack behavior taxonomy, discuss validation boundaries, and summarize future extensions of CyberClear.
- **Appendix E: More Cases.** We provide additional qualitative examples of attack-chain provenance reconstruction and error analysis.

A ADDITIONAL EXPERIMENTS

A.1 EVALUATION WITH ADDITIONAL LLM BACKBONES

Table 4 reports additional results with GPT-4o-mini and Gemini-3.5-Flash-Lite to examine the robustness of different agent frameworks across additional LLM backbones. The results show substantial performance variation across backbones, indicating that provenance is sensitive to the underlying model capability. With Gemini-3.5-Flash-Lite, CyberProvenance achieves the strongest performance across all five Semantic Metrics metrics while maintaining competitive code-level similarity. In contrast, performance with GPT-4o-mini is generally lower, particularly on Strict, Detail, and Scenario, highlighting the difficulty of reconstructing complete attack chains with weaker backbones. These results further show that both the agent framework and the underlying LLM contribute substantially to reliable APT attack chain provenance.

A.2 ABLATION STUDIES

To quantify the contribution of each harness component, we remove Evidence Memory, Execution Validation, and Feedback Refinement individually while keeping all other settings unchanged. As shown in Table 5, removing Execution Validation causes the largest degradation, reducing the average Semantic Metrics score from 0.7212 to 0.6472, with especially large drops in Strict, Detail, and Scenario. Removing Evidence Memory yields a comparable decline to 0.6516 and notably reduces Strict and Detail by 0.0759 and 0.0888, respectively, showing the importance of maintaining consistent evidence across long-context logs. Removing Feedback Refinement leads to a smaller but consistent decrease, with the average semantic score falling to 0.6897 and all evaluation metrics degrading relative to the full model. These results demonstrate that Evidence Memory and Execution Validation provide the main gains, while Feedback Refinement further improves reconstruction quality through iterative correction.

A.3 CONSISTENCY WITH HUMAN JUDGMENTS

To evaluate the reliability of our semantic evaluation method, we compare the model-based scores with human judgments. Two human experts independently assess the generated provenance graphs, and each expert repeats the evaluation three times; the final human score is obtained by averaging all judgments. We use the GPT-5.6-Terra evaluation results for this analysis and measure the correlation between automatic scores and human assessments. As shown in Figure 8, the scores produced by our evaluator exhibit strong consistency with human judgments across all five dimensions. The Pearson correlation coefficients reach 0.9425, 0.9351, 0.9445, 0.9426, and 0.9376 for Single, Loose, Strict, Detail, and Scenario, respectively. These results indicate that our evaluation method can reliably approximate human assessments while providing a scalable alternative for evaluating attack-chain provenance reconstruction.

Table 4: Additional results of representative agent frameworks on CyberClear with different LLM backbones.

Methods		Code-Level Metrics				Semantic Metrics				
		BLEU	ROUGE-L	CodeBLEU	Pass@1	Single	Loose	Strict	Detail	Scenario
OpenClaw (Single)		0.3617	0.5236	0.3019	0.8113	0.4509	0.3588	0.2253	0.2602	0.3209
		0.3428	0.4975	0.3659	0.8267	0.6521	0.5748	0.4146	0.3917	0.4615
OpenClaw (Multiple)		0.4483	0.6027	0.4872	0.9085	0.5694	0.4592	0.2982	0.3279	0.4293
		0.4237	0.5718	0.5526	0.9178	0.5894	0.5127	0.3812	0.3526	0.4098
AgentVerse		0.3964	0.5819	0.5298	0.9289	0.6383	0.5314	0.3510	0.3650	0.4869
		0.3749	0.5521	0.5963	0.9378	0.5486	0.4725	0.3497	0.3264	0.3618
LLM-Debate		0.4389	0.5651	0.5726	0.8533	0.7407	0.6388	0.4284	0.4178	0.5727
		0.4146	0.5387	0.6385	0.8711	0.6418	0.5589	0.4236	0.3895	0.4587
MultiPersona		0.4075	0.5943	0.3441	0.9178	0.4299	0.3581	0.2263	0.2476	0.2945
		0.3893	0.5685	0.4148	0.9267	0.6635	0.5864	0.4281	0.3972	0.4716
LLM-Blender		0.4496	0.5868	0.4985	0.8622	0.5911	0.4816	0.3147	0.3397	0.4475
		0.4261	0.5584	0.5689	0.8800	0.5962	0.5187	0.3864	0.3615	0.4219
DyLAN		0.4362	0.5762	0.6144	0.8622	0.7863	0.6842	0.4583	0.4366	0.6127
		0.4584	0.5783	0.6643	0.8844	0.7583	0.6273	0.4258	0.4538	0.6029
MacNet		0.3908	0.5559	0.4066	0.8467	0.4413	0.2981	0.1763	0.2564	0.3343
		0.3933	0.5578	0.5444	0.8466	0.5742	0.4674	0.3567	0.4523	0.4196
CAMEL		0.3902	0.5873	0.3384	0.9222	0.4264	0.3596	0.2278	0.2454	0.2894
		0.3887	0.5842	0.4219	0.9222	0.4753	0.3606	0.2563	0.2505	0.3009
AgentScope-V2		0.4642	0.6073	0.6513	0.9867	0.6646	0.5702	0.3934	0.4035	0.4998
		0.5035	0.6435	0.7311	0.8622	0.7095	0.6889	0.4785	0.5557	0.6278
CyberProvenance (Ours)		0.4871	0.6246	0.6673	0.9822	0.6748	0.5792	0.3974	0.4046	0.5098
		0.4896	0.6738	0.6941	0.9911	0.8485	0.7186	0.6179	0.5843	0.6897

Table 5: Ablation study of CyberProvenance on CyberClear using GPT-5.6-Terra. Red values indicate performance drops relative to the full model.

Variants	Code-Level Metrics				Semantic Metrics				
	BLEU	ROUGE-L	CodeBLEU	Pass@1	Single	Loose	Strict	Detail	Scenario
CyberProvenance	**0.4952**	**0.6968**	**0.7016**	**0.9333**	**0.8897**	**0.7438**	**0.6337**	**0.6124**	**0.7263**
w/o Evidence Memory	0.4510 ↓0.0442	0.6423 ↓0.0545	0.6554 ↓0.0462	0.9044 ↓0.0289	0.8421 ↓0.0476	0.6845 ↓0.0593	0.5578 ↓0.0759	0.5236 ↓0.0888	0.6502 ↓0.0761
w/o Execution Validation	0.4598 ↓0.0354	0.6492 ↓0.0476	0.6615 ↓0.0401	0.8978 ↓0.0355	0.8470 ↓0.0427	0.6780 ↓0.0658	0.5485 ↓0.0852	0.5200 ↓0.0924	0.6425 ↓0.0838
w/o Feedback Refinement	0.4825 ↓0.0127	0.6811 ↓0.0157	0.6889 ↓0.0127	0.9178 ↓0.0155	0.8734 ↓0.0163	0.7169 ↓0.0269	0.5983 ↓0.0354	0.5716 ↓0.0408	0.6885 ↓0.0378

A.4 Additional Performance Analysis Across Difficulty Levels

The additional results examine how code-level similarity varies with task difficulty across different agent frameworks and LLM backbones. BLEU, ROUGE-L, and CodeBLEU remain relatively stable for most methods on easy and medium instances, but decrease substantially on hard instances. As shown in Figure 9, the decline becomes more pronounced as the complexity of attack-chain reconstruction increases. Hard instances also produce larger performance gaps among different agent frameworks, indicating greater differences in their ability to preserve the structure of the target

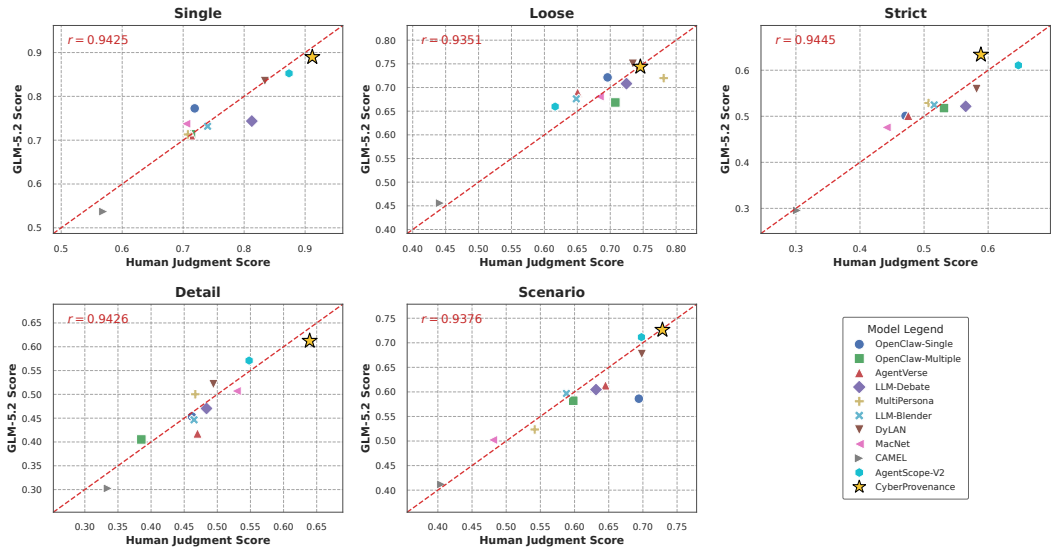


Figure 8: Consistency between GPT-5.6-Terra-based evaluation and human judgments. Two human experts independently evaluate generated provenance graphs, with each evaluation repeated three times and averaged.

provenance graphs. This trend is consistently observed across the three LLM backbones, although the extent of degradation varies across models. These results further show that increasing task difficulty challenges both the semantic reconstruction of attack chains and the structural consistency of the generated graphs.

A.5 Additional Performance Analysis Across Attack Stage Levels

Across the three code-level metrics, agent frameworks exhibit clear differences between single-stage and multi-stage attack-chain reconstruction. Most methods show relatively stable BLEU and ROUGE-L scores across the two settings, while larger variations can be observed in CodeBLEU. As shown in Figure 10, AgentScope-V2 and CyberProvenance maintain consistently high scores across different LLM backbones, indicating stronger structural fidelity in the generated provenance graphs. In contrast, several general-purpose agent frameworks achieve noticeably lower similarity, especially when reconstructing more complex multi-stage attack chains. The comparison across GPT-5.6-Terra, DeepSeek-V4-Flash, and Qwen-3.8-Max further shows that the same agent framework can exhibit different levels of robustness depending on the underlying LLM. These results suggest that reliable provenance graph generation depends not only on the backbone model, but also on whether the agent framework provides effective support for iterative refinement, evidence consistency, and cybersecurity-oriented validation.

B Experimental Details

B.1 Baselines

For all baselines, we set the temperature to 0.2 and use top-$p = 1.0$. The remaining configuration details for each baseline are provided below:

OpenClaw (Single) (OpenClaw). We configure OpenClaw with a single LLM agent for security log analysis, attack chain reconstruction, and DOT generation, using Web Search when needed and allowing up to three rounds of self verification and refinement. We utilize the official implementation available at `https://github.com/openclaw/openclaw`.

OpenClaw (Multiple). We build three collaborative OpenClaw agents for attack evidence extraction, DOT code generation, and output verification, respectively. Web Search is restricted to ATT&CK identifier and name verification, and the system performs up to three rounds of gener-

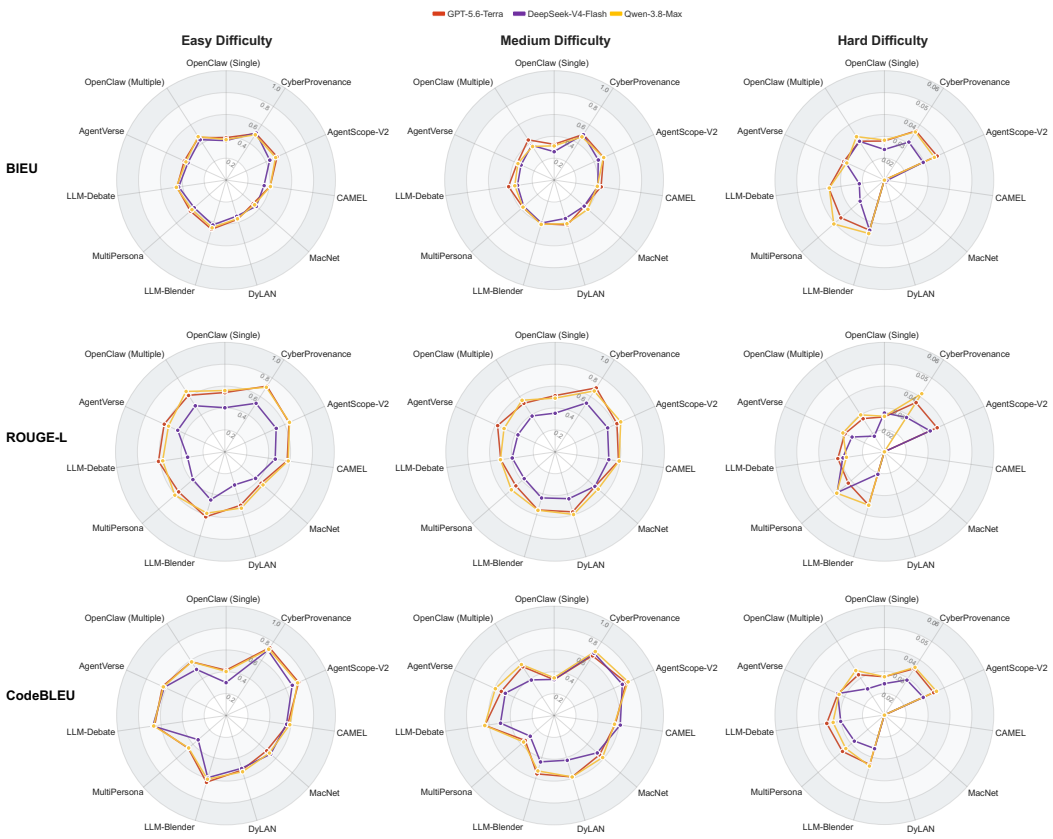


Figure 9: Code-Level similarity across difficulty levels using BLEU, ROUGE-L, and CodeBLEU. Columns correspond to easy, medium, and hard instances, and each radar chart compares agent frameworks under three LLM backbones.

BIEU

	GPT-5.6-Terra		DeepSeek-V4-Flash		Qwen-3.8-Max	
	Single	Multi	Single	Multi	Single	Multi
OpenClaw (Single)	0.36	0.40	0.33	0.35	0.34	0.39
OpenClaw (Multiple)	0.45	0.45	0.42	0.42	0.44	0.45
AgentVerse	0.39	0.42	0.36	0.38	0.38	0.41
LLM-Debate	0.44	0.45	0.41	0.42	0.43	0.44
MultiPersona	0.41	0.43	0.37	0.39	0.39	0.42
LLM-Blender	0.45	0.45	0.42	0.42	0.44	0.45
DyLAN	0.37	0.43	0.34	0.37	0.37	0.42
MacNet	0.33	0.37	0.37	0.36	0.34	0.41
CAMEL	0.41	0.43	0.35	0.39	0.41	0.40
AgentScope-V2	0.51	0.48	0.44	0.44	0.50	0.50
CyberProvenance	0.50	0.48	0.50	0.49	0.48	0.48

ROUGE-L

	GPT-5.6-Terra		DeepSeek-V4-Flash		Qwen-3.8-Max	
	Single	Multi	Single	Multi	Single	Multi
OpenClaw (Single)	0.52	0.55	0.48	0.50	0.50	0.55
OpenClaw (Multiple)	0.58	0.59	0.56	0.58	0.58	0.61
AgentVerse	0.59	0.59	0.54	0.56	0.56	0.59
LLM-Debate	0.58	0.58	0.52	0.54	0.54	0.58
MultiPersona	0.52	0.55	0.55	0.57	0.57	0.60
LLM-Blender	0.60	0.59	0.55	0.56	0.57	0.60
DyLAN	0.51	0.57	0.52	0.55	0.54	0.60
MacNet	0.44	0.48	0.47	0.48	0.46	0.52
CAMEL	0.58	0.58	0.57	0.58	0.58	0.59
AgentScope-V2	0.64	0.61	0.62	0.63	0.65	**0.66**
CyberProvenance	**0.71**	**0.67**	**0.68**	**0.70**	**0.68**	**0.69**

CodeBLEU

	GPT-5.6-Terra		DeepSeek-V4-Flash		Qwen-3.8-Max	
	Single	Multi	Single	Multi	Single	Multi
OpenClaw (Single)	0.37	0.43	0.38	0.38	0.37	0.42
OpenClaw (Multiple)	0.56	0.59	0.56	0.57	0.56	0.59
AgentVerse	0.60	0.61	0.60	0.62	0.61	0.63
LLM-Debate	0.65	0.65	0.64	**0.67**	**0.65**	**0.67**
MultiPersona	0.41	0.47	0.42	0.42	0.42	0.46
LLM-Blender	0.61	0.62	0.57	0.59	0.58	0.60
DyLAN	0.52	0.59	0.50	0.53	0.54	0.59
MacNet	0.49	0.55	0.54	0.52	0.52	0.59
CAMEL	0.57	0.57	0.56	0.59	0.59	0.56
AgentScope-V2	**0.73**	**0.71**	**0.67**	**0.69**	**0.72**	**0.74**
CyberProvenance	**0.71**	**0.69**	**0.68**	**0.70**	**0.70**	**0.72**

0.0–0.2 0.2–0.4 0.4–0.6 0.6–0.8 0.8–1.0

Figure 10: Code-Level similarity across single-stage and multi-stage instances using BLEU, ROUGE-L, and CodeBLEU. Columns correspond to three LLM backbones, and each heatmap compares agent frameworks under single-stage and multi-stage settings.

ation, review, and refinement. We use the official code released at `https://github.com/openclaw/openclaw`.

AgentVerse (Chen et al., 2024). We configure AgentVerse with four expert agents that collaborate under a predefined protocol for log analysis and attack chain reconstruction. The agents jointly analyze evidence, propose candidate reconstructions, and consolidate the final DOT graph. The source code is available at `https://github.com/OpenBMB/AgentVerse/`.

LLM-Debate (Du et al., 2023). We configure LLM-Debate with three agents and two debate rounds. Each agent independently generates an attack-chain candidate, then revises it after reviewing the other agents' outputs, with the final results consolidated into a single reconstruction. We use the official implementation available at `https://github.com/composable-models/llm_multiagent_debate`.

MultiPersona (Wang et al., 2024). In our experiments, MultiPersona uses a single LLM instance to simulate three task-specific personas for attack-chain reconstruction. The personas interact over multiple rounds to analyze attack evidence and collaboratively refine the reconstructed chain. The experiments are conducted using the implementation provided at `https://github.com/MikeWangWZHL/Solo-Performance-Prompting`.

LLM-Blender (Jiang et al., 2023). For LLM-Blender, three generator LLMs produce 11 candidate reconstructions, which are pairwise ranked by PairRanker. The top three candidates ($K = 3$) are retained and fused by GenFuser to produce the final DOT graph. Our experiments build on the implementation released at `https://github.com/yuchenlin/LLM-Blender`.

DyLAN (Liu et al., 2024). We enable DyLAN's dynamic agent selection, multi-round interaction, and early stopping for attack-chain reconstruction. The corresponding implementation is obtained from `https://github.com/SALT-NLP/DyLAN`.

MacNet (Qian et al., 2025). In the MacNet setting, agents are organized as a directed acyclic collaboration network and interact following the topological order to derive the final reconstruction. We conduct the experiments with the MacNet implementation from `https://github.com/OpenBMB/ChatDev/tree/macnet`.

CAMEL (Li et al., 2023a). For CAMEL, we assign predefined roles to the participating agents and allow them to collaboratively reconstruct the attack chain through autonomous role-playing dialogue. The experimental setup is implemented with the codebase at `https://github.com/camel-ai/camel`.

AgentScope-V2 (Dawei Gao, 2024; 2025). AgentScope-V2 is configured with role-specialized agents that exchange intermediate evidence and reconstruction results through structured messages, allowing later agents to refine earlier outputs before producing the final graph. We implement this baseline using the AgentScope repository at `https://github.com/agentscope-ai/agentscope`.

B.2 Attack Validation Environment

For environment-based validation, we select execution procedures using the predicted ATT&CK Technique identifiers, while obtaining platform and runtime settings from predefined environment configurations. The configurations of the execution and telemetry tools are provided below:

Atomic Red Team (Red Canary). We use Atomic Red Team to reproduce individual attack steps. For each predicted Technique identifier, we retrieve candidate tests from a predefined catalog and select compatible implementations according to their supported platforms, execution requirements, and dependencies. Execution results and the corresponding telemetry are retained for subsequent comparison with the observed evidence. We use the official implementation available at `https://github.com/redcanaryco/atomic-red-team`.

CALDERA (Apache Software Foundation). We use CALDERA to orchestrate multi-step attack chains with execution dependencies. Predicted Technique identifiers are mapped to predefined abilities, whose requirements and parsers specify execution prerequisites and extract facts from procedure outputs. The execution agents and planner are specified by the experiment configuration. Operation records and execution traces are retained to assess step completion and examine dependencies between steps. We use the official implementation available at `https://github.com/apache/caldera`.

Ludus (Bad Sector Labs). We use Ludus with Proxmox to provision isolated virtual machines from predefined environment templates. The range configuration specifies the operating systems, network topology, services, and telemetry components required by the selected procedures. Testing mode is configured to block communication outside the experimental range, and snapshots provide

reproducible starting states for validation runs. The official documentation is available at `https://docs.ludus.cloud/`.

Sysmon (Microsoft). We deploy Sysmon on virtual machines to collect host activity under a predefined event-filtering configuration. The collected events cover process creation, network connections, and file activity, providing host-level observations for comparison with the reconstructed attack steps. We use the official distribution available at `https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon`.

Linux Audit (Linux Audit Project). We deploy Linux Audit on virtual machines and configure audit rules for the system calls and file operations relevant to the selected procedures. The audit daemon collects the resulting records, including timestamps, process identifiers, user identifiers, and operation outcomes where available. These records provide execution evidence for validating attack steps on Linux hosts. We use the official implementation available at `https://github.com/linux-audit/audit-userspace`.

Zeek (Zeek Project). We use Zeek to monitor traffic visible on the experimental range's designated capture interface. Connection records and protocol-specific logs, including DNS and HTTP records where applicable, complement the host telemetry collected by Sysmon and Linux Audit. The resulting network observations are associated with validation runs and incorporated into the structured validation reports. We use the official implementation available at `https://github.com/zeek/zeek`.

C Details of Semantic Evaluation via LLM-as-a-Judge

C.1 Scoring Criteria

We evaluate each reference attack step by aligning it with the predicted content and assessing the degree of semantic agreement. Based on this comparison, each step is categorized as **Correct**, **Minor Partial**, **Moderate Partial**, **Major Partial**, **Incorrect**, or **Missing**. We then assign an integer raw score in $[0, 100]$ according to the step-level scoring criteria below and normalize it to $[0, 1]$ by dividing by 100.

- **(80,100] (Correct):** The reference attack event is accurately reconstructed. Its actor, core action, target, and overall attack meaning are preserved. A slightly lower score within this range may be assigned when only limited non-critical details are omitted or harmlessly imprecise.
- **(60,80] (Minor Partial):** The same event and its core behavior are clearly identifiable, but a secondary behavioral qualifier, execution method, local outcome, or causal explanation is incomplete. These differences do not alter the identity, attack stage, or principal meaning of the event.
- **(40,60] (Moderate Partial):** The same event remains identifiable, but at least one core semantic component of the actor, action, target, or outcome is missing, incorrect, or ambiguous. Consequently, the attack process is incomplete, although it has not changed into a different event.
- **(20,40] (Major Partial):** The prediction is substantially incomplete and preserves only a broad core behavior, with most important information missing or inaccurate.
- **(0,20] (Incorrect):** The predicted event substantively conflicts with the reference event or exhibits only a very weak semantic correspondence.
- **0 (Missing):** The prediction contains no behavior with a meaningful semantic correspondence to the reference attack event.

C.2 Multi-dimensional Semantic Assessment

In this section, we provide detailed descriptions of our multi-dimensional semantic evaluation method. Given a reference provenance graph G and a predicted provenance graph $\hat{G}$, the judge directly compares their DOT code, treating the reference graph as authoritative. Differences in node identifiers, DOT statement order, graph layout, colors, shapes, and other purely visual properties are ignored, while entities and attributes, such as commands, file paths, IP addresses, timestamps, PIDs, services, and ATT&CK identifiers, are evaluated when they carry forensic significance. We

use GLM-5.2 as the judge model with the temperature set to 0, and require each judgment to provide supporting evidence from both the reference and predicted graphs whenever available.

Single-Step Judge. The Single-Step Judge evaluates the semantic agreement between the reference event and the predicted content for single-step samples. It directly assigns a real-valued score in the range $[0, 1]$ based on whether the predicted event preserves the core action, target, and overall meaning of the reference event.

Loose Judge. The Loose Judge evaluates multi-step samples by independently measuring the semantic coverage of each reference attack step without considering temporal order. The sample-level score is computed as the mean of all step-level scores. A predicted step may cover multiple reference steps only when it contains distinct sub-actions corresponding to these steps, and the same predicted sub-action cannot be reused to match multiple reference steps. Unsupported predicted actions cannot match reference steps and do not contribute to the score.

Strict Judge. The Strict Judge evaluates whether the predicted attack chain preserves the temporal progression of the reference chain. Let K denote the number of reference steps in the longest consecutively correct prefix, and let N denote the total number of reference steps. The prefix is counted in reference-step units rather than predicted-step positions, allowing one predicted step to cover multiple consecutive reference steps when its distinct sub-actions appear in the correct semantic order.

The prefix terminates at the first partial or incorrect reference event, or when an unsupported attack action is inserted before the reference chain is completed. If all reference steps are correctly covered, U denotes the number of unsupported attack actions appended after the completed chain. The Strict Judge score is defined as:

$$S_{\text{strict}} = \begin{cases} \dfrac{K}{N}, & K < N, \\ \dfrac{2N}{2N + U}, & K = N. \end{cases} \tag{1}$$

A completely correct chain without unsupported appended actions receives a score of 1. Unsupported entities, relations, and attributes are evaluated separately by the Detail Judge.

Detail Judge. The Detail Judge evaluates the fidelity of graph-level details, including entities, directed relations, and semantic attributes. Purely visual properties, such as node colors, shapes, and layout information, are ignored. Reference and predicted elements are aligned one-to-one within each category.

A fully correct match receives a weight of 1.0, while partial matches receive weights of 0.75, 0.50, or 0.25 according to the preserved semantic information. Incorrect matches, uncovered reference elements, and unsupported predicted elements receive zero weight.

For each category $c \in \{\text{ent}, \text{rel}, \text{attr}\}$, we compute:

$$F_c = \frac{2TP_c^w}{N_c^{\text{ref}} + N_c^{\text{pred}}}, \qquad S_{\text{detail}} = \sum_c w_c F_c, \tag{2}$$

where TP_c^w denotes the weighted semantic matches, and N_c^{ref} and N_c^{pred} denote the numbers of reference and predicted elements in category c. We set $w_{\text{ent}} = 0.35$, $w_{\text{rel}} = 0.45$, and $w_{\text{attr}} = 0.20$. The final Detail-Judge score is normalized to $[0, 1]$.

Scenario Judge. The Scenario Judge evaluates the overall consistency between the reference and predicted attack descriptions extracted from DOT files. It assigns a real-valued score in the range $[0, 1]$ according to whether the two descriptions convey the same attack process, target, method, progression, and outcome. The judge allows synonymous expressions, semantic paraphrases, and harmless generalizations. However, contradictions, unsupported core attack stages, and fabricated attack objectives reduce the final score.

C.3 LLM-AS-A-JUDGE PROMPTS

System Prompt

You are an evidence−grounded and semantically tolerant cybersecurity evaluation judge specializing in APT attack provenance.
Return exactly one valid JSON object that follows the requested schema. Use only evidence encoded in the supplied reference and predicted DOT graphs.
Do not use markdown fences or add text outside the JSON object. Before responding, verify that all quotes, backslashes, newlines, commands, paths, and evidence excerpts inside JSON strings are correctly escaped.

Figure 11: System prompt defining the evaluator's role, evidence constraints, and JSON-only response requirements.

General Evaluation Protocol

You are evaluating an APT provenance graph encoded in Graphviz DOT. Compare REFERENCE_DOT with PREDICTED_DOT.
REFERENCE_DOT is authoritative.

I. GENERAL EVALUATION PRINCIPLES

1. Judge semantic equivalence rather than lexical overlap.
2. Ignore harmless differences in node IDs, DOT statement order, layout, rank direction, font, color, shape, and other purely visual attributes.
3. Do not reward additional cybersecurity knowledge merely because it is plausible. Ignore harmless paraphrases or explanatory wording that adds no factual claim. A substantive predicted action, participating entity, relation, objective, timestamp, or ATT&CK technique that is not supported by the reference is an error.
4. For Single−Step, Loose, and Strict, judge the semantic attack event rather than exact forensic identifiers. If the semantic subject/actor role, core action, object/target role, and overall meaning agree, allow synonyms, paraphrases, active/passive rewrites, and reasonable generalization. An omitted or generalized ATT&CK ID, PID, timestamp, exact IP address, command, path, process/file/service name, or other non−behavioral detail does not by itself prevent a step−level correct judgment. Detail Judge audits those entities, relations, and attributes separately.
5. Equivalent timezone expressions are acceptable only when they denote the same event and preserve temporal order.
6. Quote or concisely identify evidence from both DOT graphs for every judgment. Prefer step numbers, node IDs, edge endpoints, labels, commands, paths, timestamps, and ATT&CK IDs.
7. Do not infer an unstated fact merely because it would be plausible in a real attack.

II. EVALUATION ROUTING

Extract the numbered attack−narrative steps from REFERENCE_DOT and let N be the number of reference steps.

If N = 1, use evaluation_mode = single_step and evaluate: Single−Step Judge, Detail Judge, and Scenario Judge.
If N >= 2, use evaluation_mode = multi_step and evaluate: Loose Judge, Strict Judge, Detail Judge, and Scenario Judge.

Timeline step boxes are evaluated by the step metrics. Do not count step boxes again as system entities under Detail Judge.

Step−level judges evaluate attack−step coverage and temporal organization from the narrative timeline. They must not separately inventory or score system nodes, provenance edges, edge attributes, or ATT&CK annotations; Detail Judge owns that structural audit. A step is covered when its semantic subject/actor role, core action, object/target role, and overall event meaning agree. These roles need

General Evaluation Protocol (continued)

not use the same words or exact identifiers. This semantic check is not a second edge−level score.

III. COMMON 0−−100 SCORING PROTOCOL

81−−100: Faithful
61−−80: Mostly Faithful
41−−60: Partially Faithful
21−−40: Weakly Aligned
0−−20: Minimally Aligned or Failed

For Strict and Detail, these bands only interpret the computed score. For Single−Step, every Loose reference step, and Scenario Judge, they guide direct score selection and are refined in Sections VI, VII, and X.
Reserve 100 for complete semantic equivalence under the content evaluated by the metric. Use 0 when no meaningful semantic correspondence exists.

IV. SEMANTIC ALIGNMENT AND FALSE−POSITIVE ACCOUNTING

Strict−specific simplification: a reference behavior in the wrong chronological order, the absence of a corresponding next predicted sub−action, and a prediction that ends early are all incorrect. Do not use separate missing or out_of_order labels in strict.step_checks. Use unsupported_inserted only when an additional reference−unsupported attack action is inserted into the predicted narrative before the reference chain has been completed.

Prediction−only label:

unsupported: A substantive predicted action, participating entity, relation, objective, timestamp, or ATT&CK technique cannot be supported by or matched to any reference content. A named entity is substantive when the prediction asserts that it participates in the attack. Harmless paraphrases, connective wording, legends, and formatting that introduce no new factual claim are not unsupported.

Correct reference coverage and unsupported prediction must be evaluated independently.
Do not invalidate a correctly aligned reference item merely because other prediction content is fabricated.
Single−Step and Loose score only their reference steps and do not count prediction−side false positives .
Unsupported content is audited separately by Detail, Scenario, and unsupported_content.
Scenario does not count units or compute precision/recall; it considers unsupported content's semantic impact in its holistic score.

For Detail Judge, use weighted semantic precision, recall, and F1.

Within the category being scored, let:
− N_ref be the number of unique substantive reference semantic units;
− N_pred be the number of unique substantive predicted semantic units;
− TP_w be the weighted semantic true−positive count. A correct one−to−one alignment contributes 1.0. A partial alignment must additionally receive a partial_severity and credit: minor partial = 0.75, moderate partial = 0.50, and major partial = 0.25. Incorrect, missing, and unsupported_inserted contribute 0.

Then compute:

semantic_recall = TP_w / N_ref
semantic_precision = TP_w / N_pred
semantic_f1 = 2 ∗ semantic_precision ∗ semantic_recall/ (semantic_precision + semantic_recall)

The equivalent count form is:

semantic_f1 = 2 ∗ TP_w / (N_ref + N_pred)

General Evaluation Protocol (continued)

raw_score = 100 ∗ semantic_f1
final_score = raw_score

Each reference unit may align to at most one predicted unit, and each predicted unit may align to at most one reference unit. A compound prediction step may cover multiple reference steps only after being decomposed into distinct sub−actions. Do not create many−to−one credit from the same sub−action.

For Detail rows, identify a matched predicted unit canonically with the pair prediction_match and prediction_evidence. Within one Detail category, do not reuse the same normalized pair for two different reference items. If the same display name occurs in different predicted nodes or edges, include enough node/edge
evidence to distinguish the two units.

Partial−severity calibration:

Minor Partial (credit 0.75): the core fact is faithful and only a limited investigation−relevant qualifier is omitted, generalized, or locally inaccurate;
Moderate Partial (credit 0.50): the same event or item is clearly recognizable, but one essential participant, object, method, result, or relation detail is missing or inaccurate;
Major Partial (credit 0.25): only the central behavior or role remains recognizable and most distinguishing information is absent, but the match is still stronger than keywords or category overlap.

For correct, incorrect, missing, and unsupported_inserted, set partial_severity to null and credit to 1.0, 0, 0, and 0 respectively. Do not use a partial credit merely to express uncertainty; select the credit from the semantic evidence.

V. ORDER AND COMPOUND−STEP RULES

One predicted step may match multiple reference steps when it contains multiple distinguishable attack sub−actions. Segmentation differences are not errors.

Order−of−mention convention:
1. Explicit temporal evidence has highest priority, including before, after, first, then, subsequently, timestamps, numbered actions, and directed temporal arrows.
2. If no explicit temporal evidence exists, textual order of mention is chronological order. Thus A and B means A followed by B, and A, B, and C means A followed by B followed by C.
3. Actions on separate lines are ordered from top to bottom.
4. Explicit temporal meaning overrides surface mention order. For example, B occurred after A means A followed by B.
5. If the text explicitly says actions are simultaneous, do not impose a sequential order from mention order alone.
6. These rules apply to attack actions, not lists of entities or attributes. Files A and B does not describe two ordered attack actions.

Loose Judge uses these semantic correspondences and ignores their order.
Strict Judge uses ordered sub−action alignment rather than fixed Step−number alignment.
A compound predicted step may satisfy several consecutive reference steps when it contains distinct sub−actions in the same order as the reference. Several Strict rows may therefore share one prediction_step_index, but each must cite a different prediction_subaction. The same sub−action cannot receive credit twice, and a compound prediction cannot skip an unmatched intermediate reference step.

Figure 12: General evaluation protocol specifying shared principles for evidence-grounded semantic comparison of reference and predicted provenance graphs.

Single-Step and Loose Judge Prompt

SINGLE−STEP JUDGE

1.Apply only when N = 1. Compare the single reference attack step with the predicted attack content. Judge whether the semantic subject/actor role, core action, object/target role, and overall attack meaning agree.

2.Allow synonymous or generalized expression and ignore ATT&CK annotations and exact forensic identifiers at this step level. Supporting evidence may occur in one compound prediction step, but it must clearly express the reference behavior.

3.Assign exactly one reference−alignment label from Section IV: correct, partial, incorrect, or missing. Apply the entity boundaries already incorporated into those definitions when the action matches but an entity is omitted, generalized, or contradictory.

4.Assign one direct integer step_score from 0 to 100 to the only reference step. Use exactly the same semantic
anchors as Loose Judge

The judgment, partial_severity, and step_score must agree. The Single−Step score is exactly this step score:

single_step_judge_score = step_score

Single−Step has no temporal−order requirement and does not use N_pred, precision, recall, or F1. Extra prediction−only actions, entities, relations, and attributes do not reduce this reference−step score; record them in unsupported_content and evaluate them under Detail and Scenario. If no meaningful predicted event corresponds to the reference step, use judgment=missing and step_score=0.

LOOSE JUDGE

Apply only when N >= 2. Evaluate unordered semantic coverage of reference attack steps. Do not penalize reordering.
For every reference step, locate a semantically corresponding predicted sub−action if one exists.

Assign exactly one reference−alignment label from Section IV to every reference step: correct, partial, incorrect, or missing.
Apply the entity boundaries already incorporated into those definitions when the action matches but an entity is omitted, generalized, or contradictory.

A single predicted step may correctly cover several reference steps through distinct sub−actions. Under Loose Judge, those reference steps need not be consecutive and their predicted order is not penalized. Never use one predicted sub−action to satisfy two different reference steps.

Score every reference step independently with one integer step_score from 0 to 100. Judge whether the matched predicted sub−action preserves the reference step's semantic subject/actor role, core action , object/target role, behavioral direction, and overall event meaning. Allow synonymous or generalized wording.
Do not lower the step score merely for omitted ATT&CK annotations or exact forensic identifiers that do not change the event. Use these anchors:

81−−100, correct: The same semantic event is reconstructed: its actor/subject role, core action, object/target role, and overall meaning agree. Use 100 when this behavioral meaning is fully preserved. Use 81−−99 for harmless semantic imprecision, not for missing ATT&CK IDs or exact forensic identifiers alone.
61−−80, minor partial: The same event and its core behavior are clear, but a limited investigation−relevant component is omitted, generalized, or locally inaccurate.
41−−60, moderate partial: The same event remains identifiable, but one essential participant, object, action component, result, or causal detail is missing or wrong.
21−−40, major partial: Only the broad core event is preserved; most distinguishing information is missing or inaccurate, although the match is stronger than isolated keywords or entity overlap.

Single-Step and Loose Judge Prompt (continued)

1−−20, incorrect or minimally aligned: A candidate sub−action has only a weak event−level connection and materially conflicts with the reference action, actor/object, direction, causal meaning, or outcome.
0, missing or wholly incompatible: No meaningful predicted event corresponds to the reference step. Keyword, entity−name, or ATT&CK−label overlap alone receives 0.

The categorical judgment, partial_severity, and step_score must agree with these ranges. Extra prediction−only actions do not lower a matched reference step's Loose score and do not enter a denominator; record them in unsupported_content for the other applicable judges.

Let s_j be the step_score for reference step j and N be the total number of reference steps. Compute the sample score without integer rounding:

loose_step_score_sum = sum(s_j), j = 1,...,N
loose_judge_score = loose_step_score_sum / N

At dataset level, let M be the number of multi−step samples and first calculate the sample mean above for each sample i. Then compute the macro−average:

dataset_loose_score = (1 / M) ∗ sum(loose_judge_score_i), i = 1,...,M

Every sample therefore has equal weight regardless of its number of reference steps.
Do not pool all steps across the dataset, and do not round per−sample means before computing the dataset macro−average.

Figure 13: Single-Step and Loose Judge prompt for evaluating individual attack steps and unordered semantic coverage of multi-step attack chains.

Strict Judge Prompt

Apply only when N >= 2. Evaluate the consecutive correct prefix using ordered sub−action alignment. Maintain a reference cursor and a prediction cursor. Decompose compound predicted steps into distinguishable ordered sub−actions.
One predicted step may advance the reference cursor through multiple consecutive reference steps, provided those sub−actions are semantically correct and appear in the required order.

A reference step is strict−correct when:

1.its semantic subject/actor role matches;
2.its core action/predicate matches;
3.its semantic object/target role and overall attack meaning are preserved;
4.it occurs at the next required chronological position; and
5.no unsupported substantive narrative−timeline action appears before it or between matched actions.

The semantic roles need not use the same words or exact identifiers. Synonyms, paraphrases, active/passive rewrites, and reasonable generalization are strict−correct. ATT&CK annotations and exact PIDs, timestamps, IPs, commands, paths, or entity names do not affect Strict unless they change which attack event is described.

Assign each Strict comparison exactly one label: correct, partial, incorrect, or unsupported_inserted.

– Use incorrect when the next predicted sub−action has the wrong behavior, actor/object, or attack meaning; when the expected behavior appears in the wrong chronological order; when no corresponding predicted sub−action exists; or when the prediction has ended early.
– Use unsupported_inserted only when the next predicted sub−action is an additional substantive attack action unsupported by the reference before the reference chain is complete. Also record that action in unsupported_content with strict_effect=terminates_prefix and one canonical step_actions unit.

Strict Judge Prompt (continued)

Let K be the number of consecutive correct reference steps from the beginning. Start with K = 0 and increase K once for each consecutively correct reference step. Distinct ordered sub−actions from one compound predicted step may increase K multiple times. Stop immediately at the first partial, incorrect, or unsupported_inserted result. Later correct content may be audited but must not increase K or restore the prefix.

If the predicted timeline ends early, label the first unmatched reference step incorrect, set prediction_step_index to null, and explain prediction ended early in reason. If a reference behavior exists but appears in the wrong chronological order, label it incorrect and explain the order mismatch in reason. An extra entity or provenance edge that appears only in the graph structure and is absent from the narrative timeline is audited by Detail Judge and does not by itself terminate the strict prefix.

Strict score computation is deterministic:
If $K < N$:
strict_raw_score = $100 * K / N$

If K = N but U unsupported substantive attack actions are appended after the complete correct reference timeline, combine ordered recall = 1 with ordered precision = N / (N + U) using F1:

strict_raw_score = $100 * 2 * N / (2 * N + U)$

U excludes harmless restatements, legends, entity details, and visual annotations. Round half up:

strict_judge_score = strict_raw_score

Use 100 only when K = N and U = 0. Use 0 when the first chronological comparison already fails.

Required consistency examples:

N=2: K=0 −> 0; K=1 −> 50; K=2 −> 100.
N=3: K=0 −> 0; K=1 −> 33; K=2 −> 67; K=3 −> 100.
N=4: K=0 −> 0; K=1 −> 25; K=2 −> 50; K=3 −> 75; K=4 −> 100.

Interpretive bands, which must not replace the formula:

81−−100: faithful or nearly complete consecutive timeline;
61−−80: long correct prefix, but failure before completion;
41−−60: meaningful intermediate prefix, often around half;
21−−40: short initial attack fragment;
0−−20: failure at the beginning or negligible correct prefix.

Figure 14: Strict Judge prompt for evaluating the consecutive correct prefix of an attack chain under ordered sub-action alignment.

Detail Judge Prompt

Inventory and evaluate reference−to−prediction coverage in three categories. Every inventory item must be atomic enough for one clear judgment. Avoid double−counting the same fact.

Before writing any Detail rows, make a one−to−one allocation separately for entities, relations, and attributes: assign each predicted unit to only its single best−matching reference unit.
Write that matched reference row first. After a predicted unit has been allocated, never use it for another reference row.
For every remaining reference item, either use a different real predicted unit or write judgment=missing with an empty prediction_match and empty prediction_evidence. A missing row must never cite a prediction.

Detail Judge Prompt (continued)

1. Entities: Semantic graph nodes representing attack participants or objects, including accounts/users, processes, tools, commands represented as explicit objects, files/artifacts, hosts/IPs, services, sockets, network objects, and other system entities. Identify an entity from its semantic label and role, such as [user], [process], [tool], [file], [network], or [service], rather than from color, shape, or fill style, because generated graphs may use different visual conventions.
2. Relations: Important directed provenance relations between system entities. Direction and action semantics both matter. A line is an edge, not an entity. An edge between two system entities that expresses an attack action such as executed, created, read, downloaded, accessed, or connected_to remains a provenance relation whether rendered as a solid, dashed, or dotted line.
3. Attributes: Investigation−relevant PID, timestamp/order, executable, command, host/IP, file path, service name, evidence/log source, and ATT&CK ID explicitly encoded in the reference.

Do not classify the following as system entities or provenance relations:

− timeline step boxes, graph titles, top−level scenario descriptions, legends, cluster boundaries, and layout anchors;
− a dotted step−to−entity edge, which is an evidence−alignment or traceability link and may be used to locate supporting evidence but is not a provenance relation;
− a dashed step−to−step edge, which is a narrative or temporal−order edge used only by Strict Judge;
− a dashed cluster boundary or an edge that exists only to control layout.

Visual style alone never determines edge type. Classify an edge from the semantic types of its endpoints and its label. A solid edge is usually a provenance relation only when it connects system entities and expresses a system or attack behavior.

PIDs, timestamps, ATT&CK IDs, executable paths, commands, and host/IP values embedded inside another entity node are normally attributes of that entity, not separate entities. A command, tool, host, or IP is a separate entity only when the DOT explicitly represents it as a semantic participant node.

The entities, relations, and attributes arrays must perform bidirectional inventory. First list every unique substantive reference item and align it one−to−one to at most one predicted item using correct, partial, incorrect, or missing. Then list every remaining unique substantive predicted entity, relation, or attribute as an unsupported false positive in unsupported_content. A fabricated attack event may legitimately contribute separate false positives to the entity and relation categories because these are distinct structural units; within each category, however, deduplicate repeated or paraphrased instances of the same semantic item.

Use the correct, partial, incorrect, and missing definitions from Section IV for every reference detail; do not create a separate Detail−specific meaning for these labels.

Treat a reversed or contradictory relation aligned to a reference relation as incorrect rather than partial when it changes the attack meaning. Record an independent fabricated relation in unsupported_content and include it in the relation N_pred count, thereby lowering relation precision and F1.
Do not apply an additional discretionary deduction or score cap. For small graphs, use the same atomic inventory and formulas; do not invent extra nodes to enlarge the denominator or ignore a substantive item merely because the graph is small.

Compute one weighted semantic F1 for each category c in {entities, relations, attributes}:

TP_w_c = sum of correct and partial credits in category c
recall_c = TP_w_c / N_ref_c
precision_c = TP_w_c / N_pred_c
f1_c = 2 ∗ TP_w_c / (N_ref_c + N_pred_c)

Use default category weights entities=0.35, relations=0.45, and attributes=0.20. Relations receive the largest weight because attack provenance depends primarily on recovering directed behavioral dependencies, not merely naming nodes. If a category has N_ref_c=N_pred_c=0, mark it inactive and renormalize the weights of the remaining active categories to sum to 1.

Detail Judge Prompt (continued)

detail_f1 = sum(active_weight_c ∗ f1_c)
detail_raw_score = 100 ∗ detail_f1
detail_judge_score = detail_raw_score

Do not round category F1 values, active weights, detail_f1, or
detail_raw_score before normalization and aggregation.
If all categories are inactive, set Detail score to 0 and flag the sample for schema review, because a provenance graph is expected to contain at least one detail unit.

Figure 15: Detail Judge prompt for evaluating entity, relation, and attribute coverage through one-to-one reference–prediction matching.

Scenario Judge Prompt

Compare only REFERENCE_SCENARIO with PREDICTED_SCENARIO.
Judge whether the two descriptions communicate the same overall attack scenario. Ignore wording, sentence order, formatting, and harmless differences in detail.
Do not infer unstated intent. An unsupported attack stage, objective, or ATT&CK technique is a substantive error.

Do not decompose the descriptions into atomic claims, do not count fields, and do not calculate precision, recall, or F1.
Read each description as one complete semantic unit and directly assign an integer score from 0 to 100. Consider, only when explicitly stated and relevant, whether they preserve the same core attack behavior or intent, principal target or object, important method/tool, high−level progression, and outcome. These are interpretive cues, not mandatory fields; never penalize a description for omitting a field that the reference itself does not state.

Use the following holistic rubric:

81−−100 (Equivalent or Nearly Equivalent): Both descriptions communicate the same attack scenario. Use 100 only when their substantive meanings are fully equivalent and the prediction adds no material unsupported meaning. Scores 81−−99 allow only minor generalization, omission, or harmless extra detail.
61−−80 (Mostly Equivalent): The same overall scenario and principal attack process are clearly recognizable, but one or more important specifics are omitted, generalized, or locally inaccurate. There is no contradiction that changes the central attack story.
41−−60 (Partially Equivalent): A substantial portion of the scenario matches and the central behavior may remain recognizable, but a major stage, target, method, objective, progression, or outcome is missing, uncertain, or wrong.
21−−40 (Weakly Related): The descriptions share only a broad attack category, isolated behavior, or limited fragment, while most of the scenario meaning is absent or different.
0−−20 (Minimally Related or Failed): Scores 0−−20 indicate that only a trivial semantic fragment overlaps and the descriptions largely communicate different scenarios. Use 0 when the predicted scenario is absent, unrelated, or fundamentally contradicts the reference scenario.

Unsupported additions are judged by their effect on the overall meaning, not by a fixed deduction. A peripheral plausible detail may have little effect; a fabricated core stage, target, objective, or outcome must reduce the score substantially. Do not award unsupported content merely because it is plausible cybersecurity knowledge.

Return a concise qualitative judgment together with matched overall meaning, important omissions or generalizations, contradictions, unsupported additions, evidence from both descriptions, and the reason for the selected score. If the reference scenario is absent, use judgment=unscorable, score 0, and request schema review.

Figure 16: Scenario Judge prompt for holistic evaluation of semantic agreement between reference and predicted attack scenarios.

Required JSON Output Schema

Return JSON only. Do not return an overall average; the caller computes the arithmetic mean of applicable metric scores.

For evaluation_mode = single_step, return exactly these top−level fields:

```
{
 sample: <sample_name>,
 evaluation_mode: single_step,
 scores: {
  single_step_judge_score: 0,
  detail_judge_score: 0,
  scenario_judge_score: 0
 },
 score_calculations: {
  single_step_judge: {
   step_score: 0,
   final_score: 0
  },
  detail_judge: {
   entities: {
    reference_unit_count: 0,
    predicted_unit_count: 0,
    weighted_true_positive: 0.0,
    recall: 0.0,
    precision: 0.0,
    f1: 0.0,
    active_weight: 0.0
   },
   relations: {
    reference_unit_count: 0,
    predicted_unit_count: 0,
    weighted_true_positive: 0.0,
    recall: 0.0,
    precision: 0.0,
    f1: 0.0,
    active_weight: 0.0
   },
   attributes: {
    reference_unit_count: 0,
    predicted_unit_count: 0,
    weighted_true_positive: 0.0,
    recall: 0.0,
    precision: 0.0,
    f1: 0.0,
    active_weight: 0.0
   },
   detail_f1: 0.0,
   raw_score: 0.0,
   final_score: 0
  },
  scenario_judge: {
   scoring_method: holistic_llm_judgment,
   selected_band: 0−−25,
   final_score: 0
  }
 },
 score_reasons: {
```

Required JSON Output Schema (continued)

```
  single_step_judge: ...,
  detail_judge: ...,
  scenario_judge: ...
 },
 unsupported_content: [],
 reference_timeline: [
  {
   index: 1,
   summary: ...,
   evidence: ...
  }
 ],
 prediction_timeline: [
  {
   index: 1,
   summary: ...,
   evidence: ...
  }
 ],
 single_step_match: {
  gold_step_index: 1,
  prediction_step_index: 1,
  prediction_subaction: ...,
  judgment: correct,
  partial_severity: null,
  credit: 1.0,
  step_score: 100,
  gold_evidence: ...,
  prediction_evidence: ...,
  reason: ...
 },
 detail: {
  entities: [],
  relations: [],
  attributes: []
 },
 scenario: {
  judgment: equivalent,
  reference_summary: ...,
  prediction_summary: ...,
  matched_meaning: [],
  missing_or_generalized: [],
  contradictions: [],
  unsupported_additions: [],
  reference_evidence: ...,
  prediction_evidence: ...,
  reason: ...
 }
}
```

For evaluation_mode = multi_step, return the common top−level fields plus loose_step_matches and strict.
The scores object contains loose_judge_score, strict_judge_score, detail_judge_score, and scenario_judge_score.
The score_calculations object contains loose_judge, detail_judge, and scenario_judge.
The strict object contains reference_step_count, prediction_step_count, consecutive_correct_prefix_count, first_failure_reference_index, first_failure_prediction_index, first_failure_type, unsupported_suffix_step_count, strict_prefix_ratio, strict_raw_score, strict_judge_score, and step_checks.

Required JSON Output Schema (continued)

SAMPLE_NAME: __SAMPLE_NAME__

REFERENCE_SCENARIO: __REFERENCE_SCENARIO__

PREDICTED_SCENARIO: __PREDICTED_SCENARIO__

REFERENCE_DOT: __REFERENCE_DOT__

PREDICTED_DOT: __PREDICTED_DOT__

Figure 17: Required JSON output schema for reporting applicable metric scores, score calculations, and supporting evaluation details.

D Attack Behavior Taxonomy and Boundaries

D.1 Complete List of Attack Behaviors

Table 6 presents the complete attack behaviors in CyberClear. We normalize the annotated attack behaviors according to their MITRE ATT&CK techniques and merge instances with identical technique sets. This process yields 89 distinct attack behaviors from 450 instances, including 46 single-technique behaviors and 43 composite behaviors. Composite behaviors capture attacks involving multiple ATT&CK techniques and thus reflect more complex attacker activity patterns.

We further evaluate the attack-behavior coverage of different agent frameworks on CyberClear. For each behavior, we compute the compilation success rate over all associated instances and regard a generated provenance graph as successful when its DOT code can be compiled correctly. As shown in Table 7, a behavior is considered supported when at least 80% of its instances are successfully compiled. This analysis provides a finer-grained view of model capability by showing which attack behaviors can be handled reliably beyond the aggregate Pass@1 score.

D.2 Method Boundaries

We ground the concrete validation implementation in the observed log evidence, while using ATT&CK Techniques to specify the corresponding attack behavior categories. Because the same Technique may admit multiple implementations across operating systems, tools, and versions, the selected implementation must be compatible with the execution context and evidence reflected in the logs. We obtain infrastructure-dependent runtime parameters from predefined configurations and validate inter-step dependencies exclusively using facts and evidence generated during the current execution. After each local refinement, we replay the attack chain from the initial snapshot to prevent previously generated facts from affecting subsequent dependency verification. We conduct all validation in authorized private environments with external network access blocked, predefined target scopes, and non-destructive execution constraints.

D.3 Future Work

Future work will explore adaptive cyber harnesses that dynamically optimize evidence retrieval, validation strategies, tool selection, and agent collaboration based on accumulated experience. Meanwhile, we plan to extend the scale and scope of CyberClear by incorporating more APT campaigns, ATT&CK behaviors, and longer multi-stage attack chains. We will broaden the benchmark to cover more diverse environments, including host, network, cloud, and container settings. We also aim to include more challenging observations with noisy logs, missing evidence, and cross-source inconsistencies. In addition, we will enrich the benchmark with finer-grained annotations for attack-step dependencies and evidence alignment. Finally, we will evaluate emerging agent frameworks and foundation models to provide a more comprehensive understanding of their capabilities in complete attack-chain reconstruction.

E MORE CASES

This section presents additional cases from CyberClear.

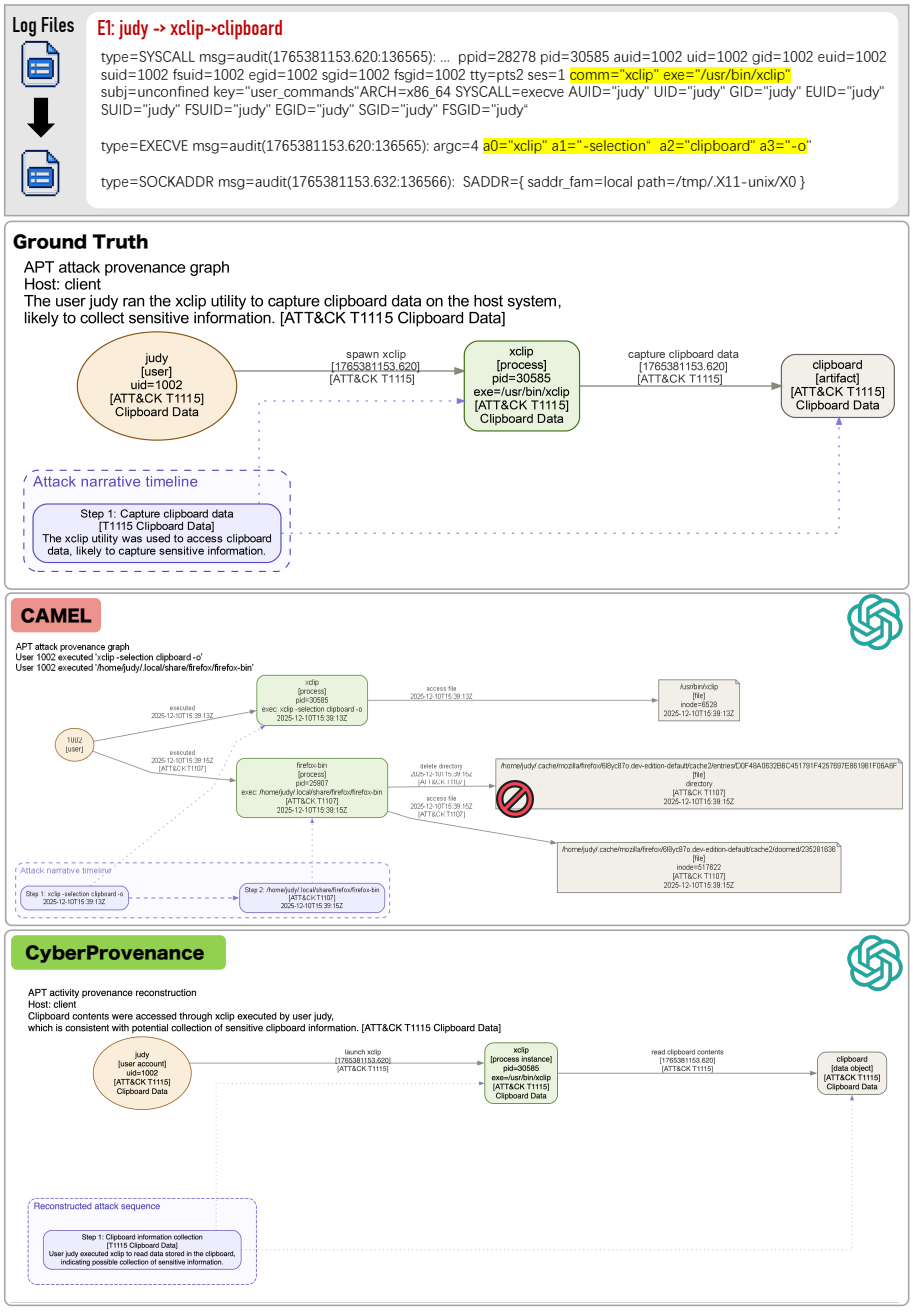


Figure 18: Case study of attack-chain provenance. LLM-Debate incorrectly omits the `bash` → `stty` dependency and its terminal-configuration step, despite explicit audit evidence highlighted in yellow: `comm="stty"`, `exe="/usr/bin/stty"`, and `stty rows 38 columns 116`. CyberProvenance correctly aligns this evidence with the corresponding entity relation and attack step, producing a provenance graph consistent with the observed execution.

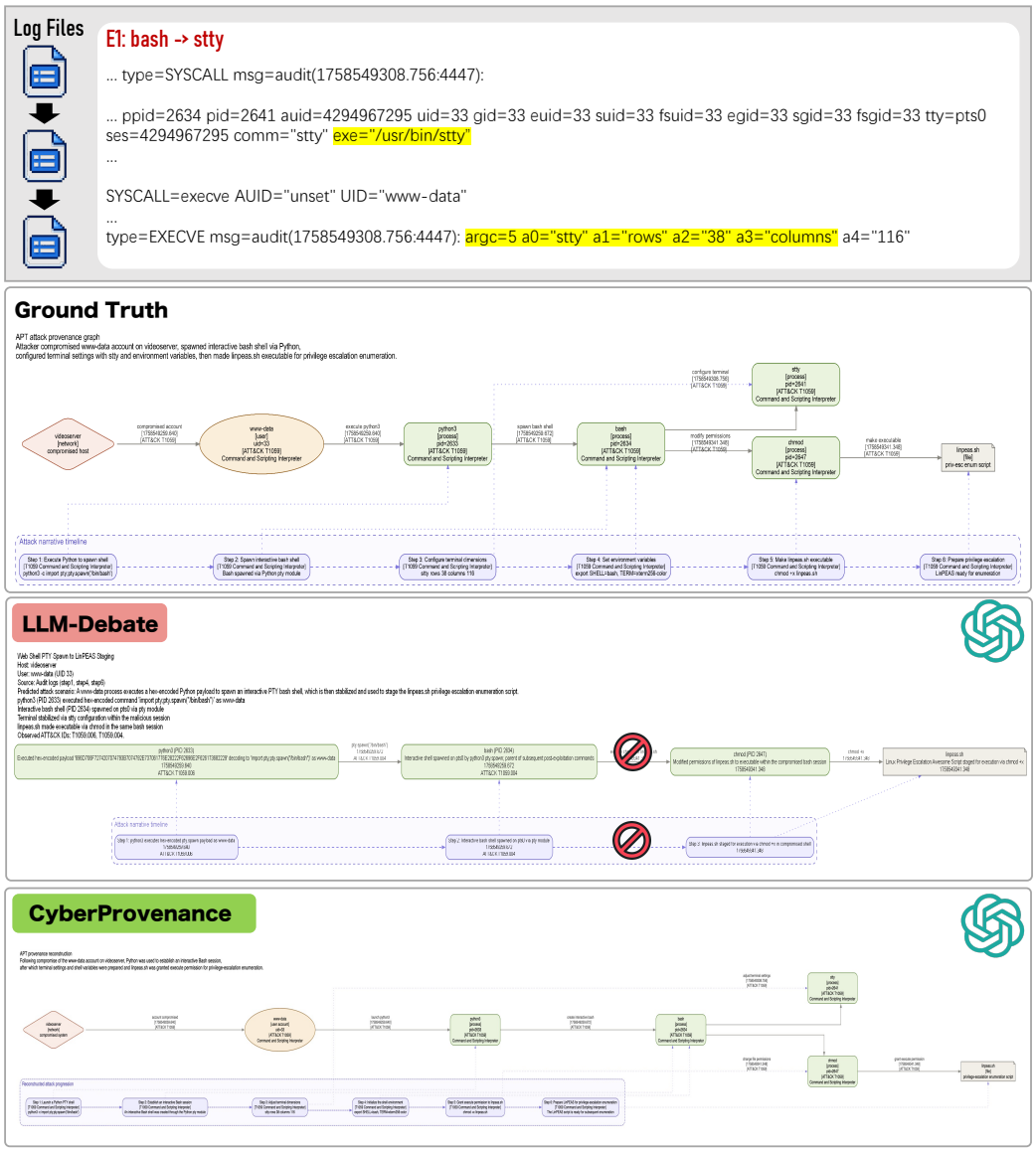


Figure 19: Case study comparing provenance results. LLM-Debate omits the `bash` → `stty` dependency, although the yellow-highlighted audit evidence explicitly records `/usr/bin/stty` and `stty rows 38 columns 116`. CyberProvenance correctly aligns this evidence with the corresponding entity relation and attack step.

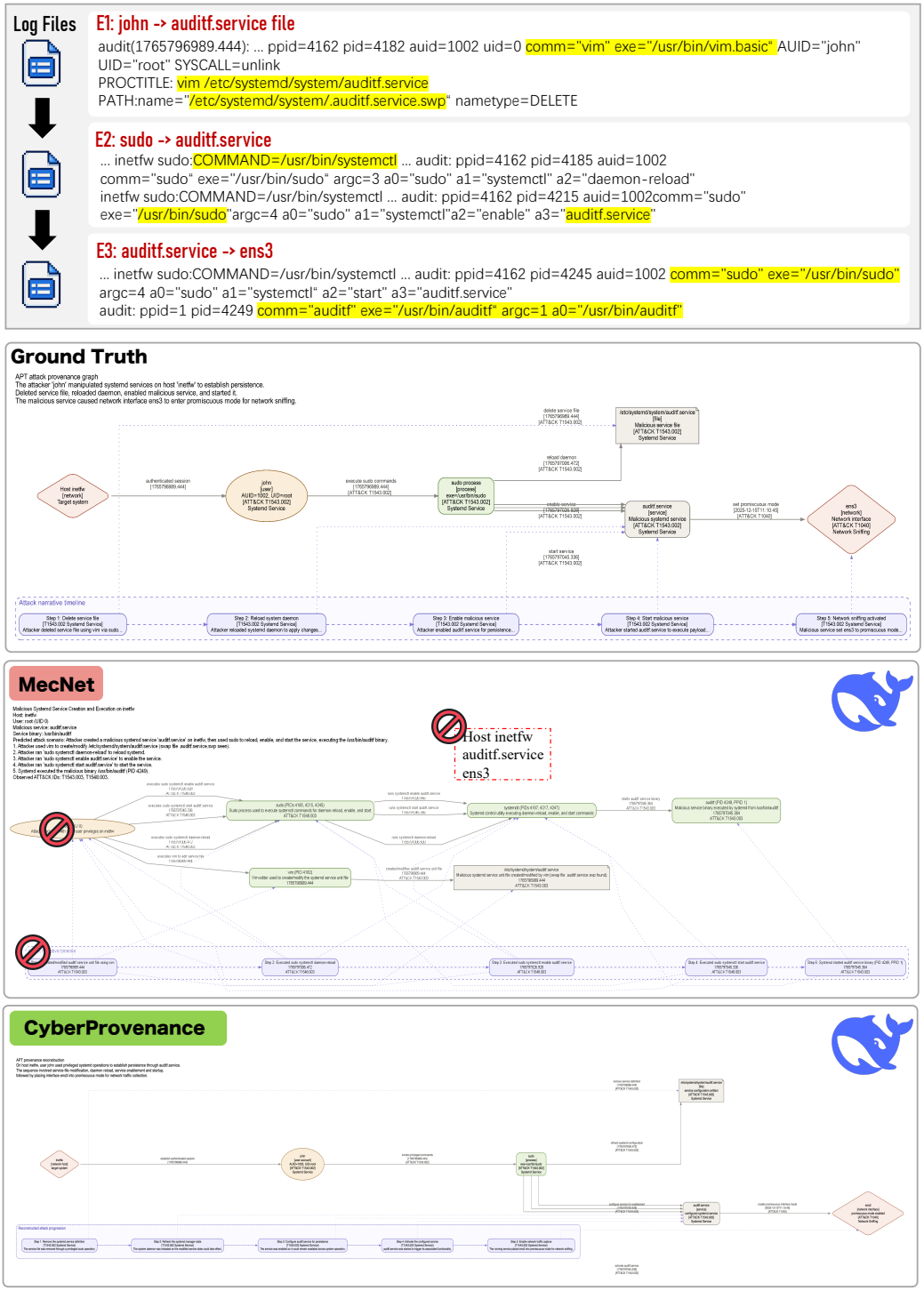


Figure 20: Case study of attack-chain provenance. MacNet fails to preserve three key evidence-supported relations, including the modification of `auditf.service`, its privileged activation through `sudo`, and the subsequent execution of the malicious service, while also omitting important host, service, and network entities. As highlighted in yellow, the logs explicitly record `vim /etc/systemd/system/auditf.service`, `systemctl enable auditf.service`, and the execution of `/usr/bin/auditf`. CyberProvenance retains these dependencies and reconstructs a more complete attack progression consistent with the observed audit evidence.

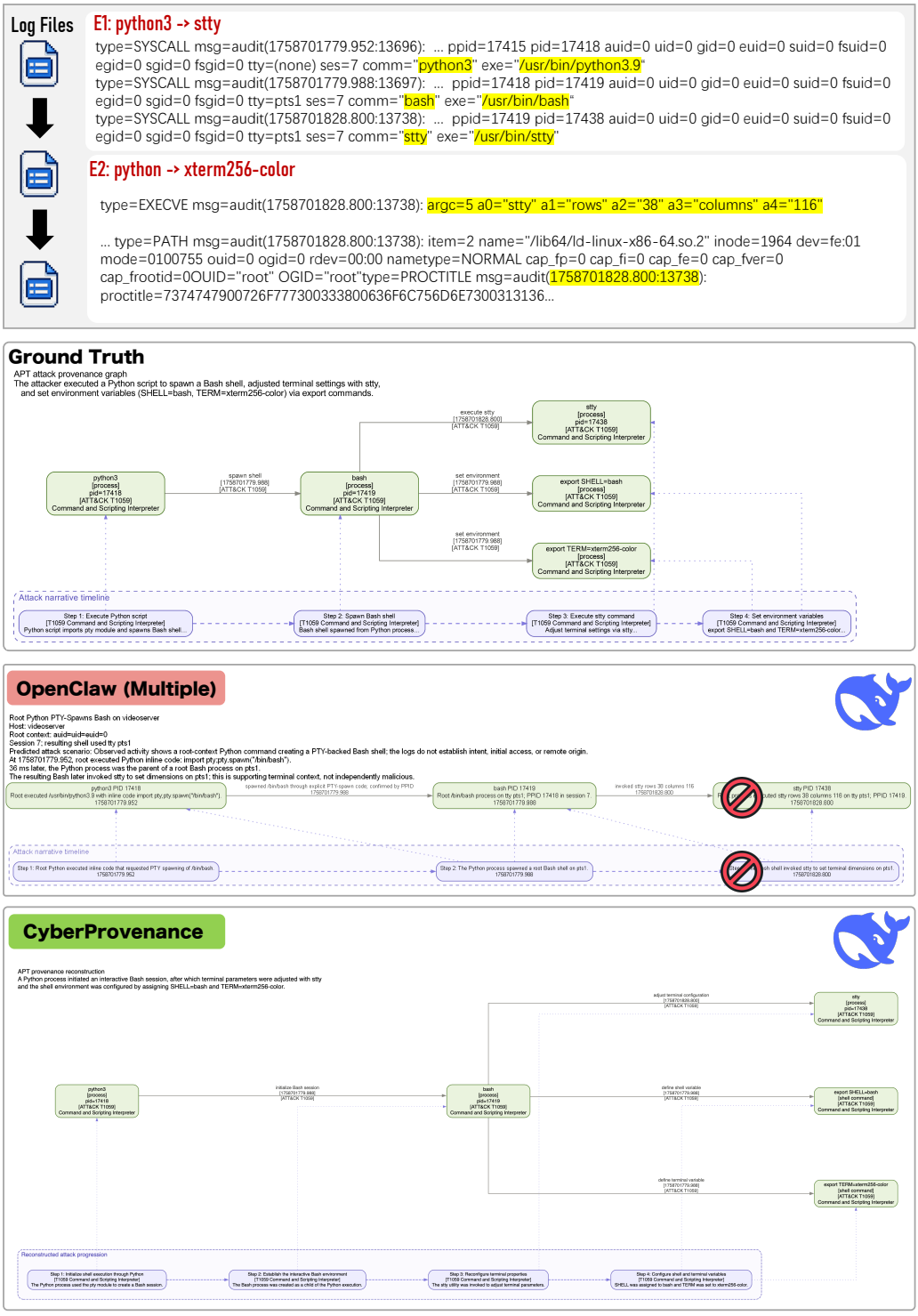


Figure 21: Case study of attack-chain provenance. OpenClaw fails to faithfully preserve the process dependencies associated with terminal configuration. As highlighted in yellow, the logs explicitly record `python3` spawning `bash`, followed by the execution of `/usr/bin/stty` with `rows 38 columns 116`. CyberProvenance correctly reconstructs these evidence-supported process relations and the corresponding attack-step progression.

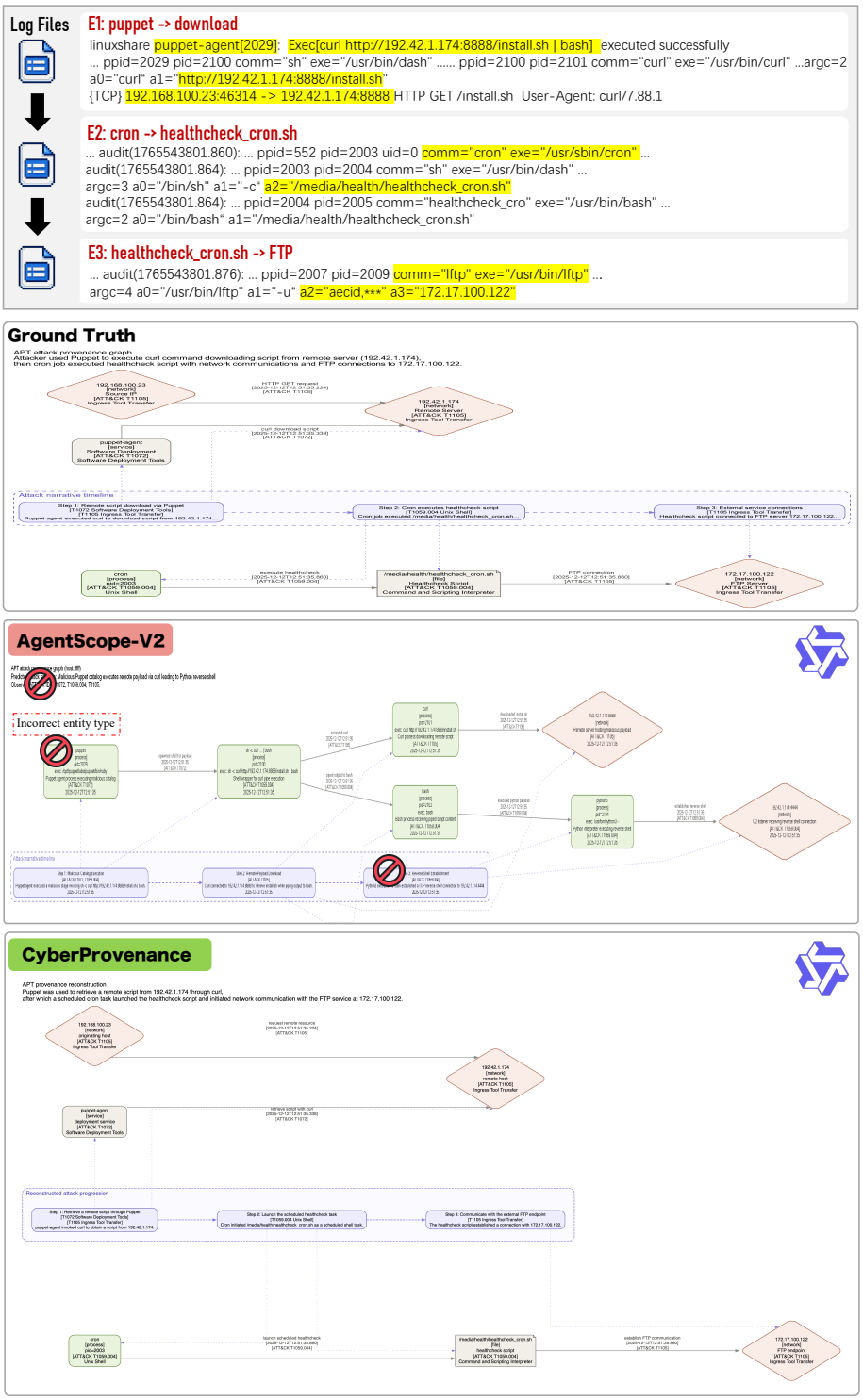


Figure 22: Case study of attack-chain provenance. AgentScope-V2 misclassifies a key entity and fails to faithfully align the reconstructed steps with the observed execution chain. As highlighted in yellow, the logs explicitly record the Puppet-triggered `curl` download, the subsequent execution of `healthcheck_cron.sh` by `cron`, and the final `lftp` connection to `172.17.100.122`. CyberProvenance correctly reconstructs these evidence-supported entities and attack-step dependencies.

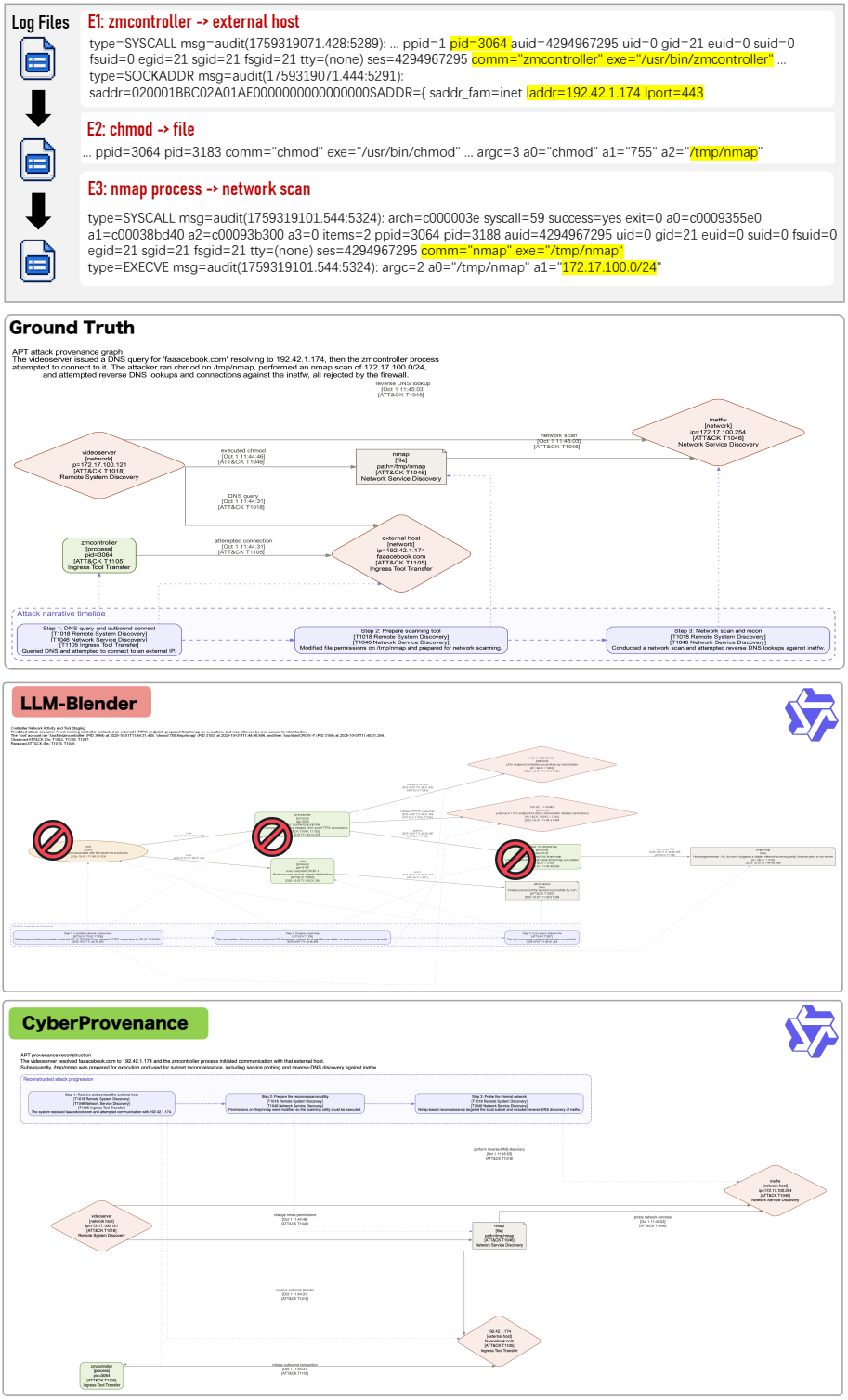


Figure 23: Case study of attack-chain provenance. LLM-Blender misses several key evidence-supported relations and fails to faithfully preserve the reconnaissance chain. As highlighted in yellow, the logs explicitly record the `zmcontroller` connection to `192.42.1.174:443`, the `chmod 755 /tmp/nmap` operation, and the subsequent `nmap` scan of `172.17.100.0/24`. CyberProvenance correctly reconstructs these observed entities, relations, and attack-step dependencies.

Table 6: All attack behaviors in CyberClear. The 450 instances yield 89 distinct technique sets: 46 single-technique and 43 composite behaviors.

Attack behavior	ATT&CK ID(s)
B01 Abuse Elevation Control Mechanism: Sudo and Sudo Caching	T1548.003
B02 Account Access Removal	T1531
B03 Account Discovery: Local Account	T1087.001
B04 Active Scanning: Vulnerability Scanning	T1595.002
B05 Active Scanning: Wordlist Scanning	T1595.003
B06 Application Window Discovery	T1010
B07 Boot or Logon Autostart Execution	T1547
B08 Clipboard Data	T1115
B09 Command and Scripting Interpreter	T1059
B10 Command and Scripting Interpreter: Unix Shell	T1059.004
B11 Create or Modify System Process: Systemd Service	T1543.002
B12 Data Destruction	T1485
B13 Data from Information Repositories	T1213
B14 Data from Local System	T1005
B15 Data from Network Shared Drive	T1039
B16 Exfiltration Over C2 Channel	T1041
B17 Exploitation for Privilege Escalation	T1068
B18 File and Directory Discovery	T1083
B19 File and Directory Permissions Modification: Linux and Mac Permissions	T1222.002
B20 Gather Victim Host Information: Software	T1592.002
B21 Hide Artifacts: Hidden Files and Directories	T1564.001
B22 Indicator Removal: Clear Command History	T1070.003
B23 Indicator Removal: File Deletion	T1070.004
B24 Ingress Tool Transfer	T1105
B25 Inhibit System Recovery	T1490
B26 Network Boundary Bridging	T1599
B27 Network Sniffing	T1040
B28 OS Credential Dumping	T1003
B29 OS Credential Dumping: /etc/passwd and /etc/shadow	T1003.008
B30 Password Policy Discovery	T1201
B31 Peripheral Device Discovery	T1120
B32 Remote Services	T1021
B33 Remote Services: SSH	T1021.004
B34 Remote System Discovery	T1018
B35 Scheduled Task/Job	T1053
B36 Scheduled Task/Job: Cron	T1053.003
B37 Service Stop	T1489
B38 Software Deployment Tools	T1072
B39 System Information Discovery	T1082
B40 System Network Configuration Discovery	T1016
B41 System Network Connections Discovery	T1049
B42 System Owner/User Discovery	T1033
B43 System Time Discovery	T1124
B44 Use Alternate Authentication Material: Application Access Token	T1550.001
B45 Valid Accounts: Local Accounts	T1078.003
B46 Virtualization/Sandbox Evasion: System Checks	T1497.001
B47 OS Credential Dumping: /etc/passwd and /etc/shadow + Exfiltration Over C2 Channel	T1003.008 +T1041
B48 Masquerading: Match Legitimate Resource Name or Location + Traffic Signaling: Port Knocking	T1036.005 +T1205.001
B49 Masquerading: Match Legitimate Resource Name or Location + Hide Artifacts: Hidden Files and Directories	T1036.005 +T1564.001
B50 Network Sniffing + Steal Application Access Token	T1040 +T1528
B51 Command and Scripting Interpreter + Hijack Execution Flow	T1059 +T1574
B52 Command and Scripting Interpreter: Unix Shell + Remote Access Tools	T1059.004 +T1219
B53 Command and Scripting Interpreter: Python + Exploit Public-Facing Application	T1059.006 +T1190
B54 Data Staged: Local Data Staging + Archive Collected Data: Archive via Utility	T1074.001 +T1560.001
B55 Account Manipulation: SSH Authorized Keys + Ingress Tool Transfer	T1098.004 +T1105
B56 Ingress Tool Transfer + Create Account: Local Account	T1105 +T1136.001
B57 Ingress Tool Transfer + Data Encrypted for Impact	T1105 +T1486
B58 Ingress Tool Transfer + Modify Authentication Process: Pluggable Authentication Modules	T1105 +T1556.003
B59 Data Encrypted for Impact + Data Manipulation: Stored Data Manipulation	T1486 +T1565.001
B60 Event Triggered Execution + Hijack Execution Flow: Path Interception by Search Order Hijacking	T1546 +T1574.008
B61 Boot or Logon Autostart Execution + System Services	T1547 +T1569
B62 Gather Victim Network Information: DNS + Gather Victim Org Information	T1590.002 +T1591
B63 Gather Victim Host Information: Software + Active Scanning	T1592.002 +T1595
B64 Rootkit + Hide Artifacts + Hijack Execution Flow	T1014 +T1564 +T1574
B65 Remote System Discovery + Network Service Discovery + Ingress Tool Transfer	T1018 +T1046 +T1105
B66 Masquerading: Match Legitimate Resource Name or Location + Application Layer Protocol: Web Protocols + Traffic Signaling: Port Knocking	T1036.005 +T1071.001 +T1205.001
B67 Scheduled Task/Job: Cron + Application Layer Protocol: Web Protocols + Ingress Tool Transfer	T1053.003 +T1071.001 +T1105
B68 Command and Scripting Interpreter: Unix Shell + Software Deployment Tools + Ingress Tool Transfer	T1059.004 +T1072 +T1105
B69 Command and Scripting Interpreter: Unix Shell + Non-Application Layer Protocol + Exploit Public-Facing Application	T1059.004 +T1095 +T1190
B70 Valid Accounts: Domain Accounts + Brute Force: Password Guessing + External Remote Services	T1078.002 +T1110.001 +T1133
B71 Valid Accounts: Local Accounts + Ingress Tool Transfer + Traffic Signaling: Port Knocking	T1078.003 +T1105 +T1205.001
B72 Account Manipulation: SSH Authorized Keys + Ingress Tool Transfer + System Binary Proxy Execution	T1098.004 +T1105 +T1218
B73 Ingress Tool Transfer + Create Account: Local Account + System Binary Proxy Execution	T1105 +T1136.001 +T1218
B74 Ingress Tool Transfer + User Execution: Malicious Link + Phishing: Spearphishing Voice	T1105 +T1204.001 +T1566.004
B75 Ingress Tool Transfer + System Binary Proxy Execution + Modify Authentication Process: Pluggable Authentication Modules	T1105 +T1218 +T1556.003
B76 Ingress Tool Transfer + Remote Access Tools + Hide Artifacts: Hidden Files and Directories	T1105 +T1219 +T1564.001
B77 Remote Services: SMB/Windows Admin Shares + Exfiltration Over C2 Channel + Command and Scripting Interpreter: PowerShell + Application Layer Protocol: Web Protocols	T1021.002 +T1041 +T1059.001 +T1071.001
B78 Scheduled Task/Job: Cron + Exploitation of Remote Services + Implant Internal Image + Deploy Container	T1053.003 +T1210 +T1525 +T1610
B79 Input Capture: Keylogging + Input Capture: Credential API Hooking + Clipboard Data + Software Extensions	T1056.001 +T1056.004 +T1115 +T1176
B80 Exfiltration Over C2 Channel + Command and Scripting Interpreter: PowerShell + Indicator Removal: File Deletion + Phishing: Spearphishing Attachment + Acquire Infrastructure: Server	T1041 +T1059.001 +T1070.004 +T1566.001 +T1583.004
B81 Exfiltration Over C2 Channel + Exploit Public-Facing Application + Create or Modify System Process: Windows Service + Phishing: Spearphishing Attachment + Acquire Infrastructure: Server	T1041 +T1190 +T1543.003 +T1566.001 +T1583.004
B82 OS Credential Dumping: LSASS Memory + OS Credential Dumping: Cached Domain Credentials + Input Capture: Keylogging + Email Collection: Remote Email Collection + Create or Modify System Process: Windows Service + Phishing: Spearphishing Attachment	T1003.001 +T1003.005 +T1056.001 +T1114.002 +T1543.003 +T1566.001
B83 OS Credential Dumping: LSASS Memory + OS Credential Dumping: Cached Domain Credentials + Input Capture: Keylogging + Command and Scripting Interpreter + Email Collection: Local Email Collection + Email Collection: Remote Email Collection + Phishing: Spearphishing Attachment	T1003.001 +T1003.005 +T1056.001 +T1059 +T1114.001 +T1114.002 +T1566.001
B84 OS Credential Dumping: LSASS Memory + OS Credential Dumping: Cached Domain Credentials + Input Capture: Keylogging + Email Collection: Local Email Collection + Email Collection: Remote Email Collection + Create or Modify System Process: Windows Service + Phishing: Spearphishing Attachment	T1003.001 +T1003.005 +T1056.001 +T1114.001 +T1114.002 +T1543.003 +T1566.001
B85 OS Credential Dumping: LSASS Memory + OS Credential Dumping: Cached Domain Credentials + Remote Services + Input Capture: Keylogging + Email Collection: Local Email Collection + Email Collection: Remote Email Collection + Create or Modify System Process: Windows Service + Phishing: Spearphishing Attachment	T1003.001 +T1003.005 +T1021 +T1056.001 +T1114.001 +T1114.002 +T1543.003 +T1566.001
B86 Remote Services: Windows Remote Management + Exfiltration Over C2 Channel + Command and Scripting Interpreter: PowerShell + Web Service: Bidirectional Communication + Ingress Tool Transfer + Exfiltration Over Web Service: Exfiltration to Cloud Storage + Lateral Tool Transfer + Acquire Infrastructure: Server	T1021.006 +T1041 +T1059.001 +T1102.002 +T1105 +T1567.002 +T1570 +T1583.004
B87 OS Credential Dumping: LSASS Memory + OS Credential Dumping: Cached Domain Credentials + Remote Services + Input Capture: Keylogging + Command and Scripting Interpreter + Application Layer Protocol: Web Protocols + Email Collection: Local Email Collection + Email Collection: Remote Email Collection + Phishing: Spearphishing Attachment	T1003.001 +T1003.005 +T1021 +T1056.001 +T1059 +T1071.001 +T1114.001 +T1114.002 +T1566.001
B88 Exfiltration Over C2 Channel + Command and Scripting Interpreter: PowerShell + Application Layer Protocol: Web Protocols + Account Discovery: Local Account + Drive-by Compromise + System Binary Proxy Execution: Mshta + System Binary Proxy Execution: Msiexec + Create or Modify System Process: Windows Service + Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder + Acquire Infrastructure: Web Services	T1041 +T1059.001 +T1071.001 +T1087.001 +T1189 +T1218.005 +T1218.007 +T1543.003 +T1547.001 +T1583.006
B89 System Service Discovery + System Network Configuration Discovery + System Owner/User Discovery + System Network Connections Discovery + Process Discovery + Permission Groups Discovery + System Information Discovery + File and Directory Discovery + Account Discovery + Password Policy Discovery + Software Discovery + System Location Discovery + Group Policy Discovery	T1007 +T1016 +T1033 +T1049 +T1057 +T1069 +T1082 +T1083 +T1087 +T1201 +T1518 +T1614 +T1615

Table 7: Attack behavior coverage of different agent frameworks under GPT-5.6-Terra. A checkmark indicates that the framework successfully handles the corresponding attack behavior, while a cross indicates failure.

Behavior	OpenClaw (Single)	OpenClaw (Multiple)	AgentVerse	LLM-Debate	MultiPersona	LLM-Blender	DyLAN	MacNet	CAMEL	AgentScope-V2	CyberProvenance
B01	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B02	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B03	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B04	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B05	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B06	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B07	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B08	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B09	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B10	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B12	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B13	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B14	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B15	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B16	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B17	✗	✓	✓	✗	✓	✗	✗	✗	✓	✓	✓
B18	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B19	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B22	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B23	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B24	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B25	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
B26	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B27	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B28	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✓
B29	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B30	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B31	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B32	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B34	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B35	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B36	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B37	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B38	✗	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓
B39	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B41	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B42	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B43	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B44	✗	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓
B45	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B46	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B47	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
B48	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✓
B49	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
B50	✗	✓	✗	✗	✓	✓	✗	✗	✓	✓	✓
B51	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓
B52	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
B53	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓
B54	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗
B55	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
B56	✗	✓	✗	✗	✗	✓	✗	✓	✓	✓	✓
B57	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓
B58	✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✓
B59	✗	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓
B60	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓
B61	✓	✗	✓	✓	✗	✓	✓	✓	✓	✗	✓
B62	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✓
B63	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓
B64	✗	✓	✗	✗	✓	✗	✗	✗	✗	✓	✓
B65	✓	✗	✗	✗	✓	✓	✓	✓	✓	✗	✓
B66	✗	✓	✓	✗	✓	✓	✗	✗	✗	✓	✗
B67	✓	✗	✓	✓	✓	✓	✓	✓	✗	✓	✓
B68	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓
B69	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓
B70	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✓
B71	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓
B72	✓	✓	✓	✗	✗	✓	✓	✓	✗	✓	✓
B73	✗	✗	✗	✗	✓	✓	✗	✗	✗	✓	✗
B74	✗	✓	✓	✗	✗	✓	✓	✓	✗	✓	✓
B75	✗	✓	✓	✓	✗	✓	✓	✗	✓	✓	✓
B76	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓
B77	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
B78	✗	✓	✗	✗	✗	✓	✓	✓	✗	✓	✓
B79	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗
B80	✗	✓	✓	✗	✗	✓	✗	✗	✗	✗	✓
B81	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗	✗
B82	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
B83	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
B84	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
B85	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
B86	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
B87	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
B88	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
B89	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗