From Assistance to Autonomy: An Empirical Study of AI Use in a Live Capture-the-Flag (CTF) Competition

Tingxuan Tang[1], Nicolas Janis[1], Kalyn Asher Montague[1], Kevin Eykholt[2], Dhilung Kirat[2], Youngja Park[2], Jiyong Jang[2], Adwait Nadkarni[1], Yue Xiao[1] *

[1]William & Mary, [2]IBM Research

Abstract

Capture-the-Flag (CTF) competitions are increasingly becoming a testbed for evaluating AI capabilities at solving security tasks, due to their controlled environments and objective success criteria. Existing evaluations have focused on how successful AI is at solving individual CTF challenges in isolation from human CTF players. As AI usage increases in both academic and industrial settings, it is equally likely that human CTF players may collaborate with AI agents to solve CTF challenges. This possibility exposes a key knowledge gap: *how do human players perceive AI CTF assistance; when assistance is provided, in what ways do they collaborate and is it effective with respect to human performance; how do humans assisted by AI compare to the performance of fully autonomous AI agents on the same set of challenges*. We address this gap with the first empirical study of AI assistance in a live, onsite CTF. In a study with 41 participants (out of the total 95 that participated in the CTF), we qualitatively study (i) how participants' perception, trust, and expectations shift before versus after hands-on AI use, and (ii) how participants collaborate with an instrumented AI assistant. Moreover, we also (iii) benchmark four autonomous CTF agents on the same fresh challenge set to compare outcomes with human teams and analyze agent trajectories. We find that, for human players, AI literacy and domain knowledge are complementary competencies, and both have irreplaceable advantages. Proficient and efficient use of AI amplifies professional skills. Importantly, although advanced autonomous agents showed outstanding performance, human-in-the-loop is the winning paradigm where AI accelerates exploration while humans provide targeted guidance and verification. We conclude with implications for the future design of CTF competitions and for building effective human-in-the-loop AI systems for security.

*Corresponding author.

1 Introduction

Large language models (LLMs) are improving at an exceptional pace and are increasingly used for security tasks, from vulnerability discovery [1–11] and exploit generation [12–15] to patch development [16–21]. Capture The Flag (CTF) competitions have emerged as a widely used benchmark to evaluate the capabilities of LLM agents in security-related challenges [22, 23]. CTF challenges are designed to simulate realistic vulnerability scenarios that require reconnaissance, hypothesis formation, iterative testing, and exploitation. CTFs also define clear success criteria (i.e., submitting the flag) and can be hosted in controlled environments for reproducible evaluation.

Recent work on AI for CTF has largely focused on *AI capability-centric* evaluations, including benchmark construction [23–25], AI agent framework design [26–30], and training pipelines that improve end-to-end performance [31]. While these efforts quantify what AI systems can do in isolation, they offer limited insight into the potential of *human-AI collaboration*. Such collaboration is necessary as full automation of security workflows remains challenging. Security-relevant tasks usually require contextual judgment and iterative verification. Over-delegating security tasks to AI introduces practical risks, including hallucinated conclusions [32], unintended tool misuse [33], wasted analyst time, and computational resources. In this paper, we study how participants perceive and leverage AI in a security context and the effectiveness of this collaboration. We address these questions through an in-person CTF competition, which provides a measurable and ethically controlled environment for observing human-AI interaction under realistic time pressure while solving security challenges. Specifically, we study the following three research questions:

RQ*$_1$*: What expectations do participants have for AI, specifically in a CTF scenario? How do perception, trust, and expectation shift? How does participants' AI exper-

tise impact their CTF scores?

• *User survey analysis of common perception (RQ_1).* We performed a pre-survey and post-survey with **41** CTF participants to measure participant expectations and reflections. We find that participants' expected AI effectiveness and willingness to use AI in future competitions decreased after hands-on use ($\mathcal{F}_1$). Participants attributed this shift to recurring model failures, including flawed reasoning, hallucinations, and non-working code ($\mathcal{F}_2$), which in turn reduced trust in AI outputs, especially among participants with higher CTF domain knowledge. Finally, we observe that AI expertise can partially compensate for limited CTF domain knowledge: participants with low CTF expertise but high AI expertise achieved competitive scores, whereas participants with intermediate CTF expertise but novice AI expertise solved few challenges ($\mathcal{F}_3$), underscoring effective AI use as a key determinant of performance.

RQ_2: How do humans actually collaborate with an AI assistant during a live CTF? What distinguishes effective collaboration from ineffective collaboration? Do collaboration strategies shift in a competition environment?

• *Qualitative analysis of Human-AI interactions (RQ_2)* To understand how CTF players actually collaborate with an AI assistant during a live competition, we develop and deploy an instrumented assistant, `CTFriend`, for participant to interact with and qualitatively analyze **2,299** chat messages to identify emergent interaction patterns, effective strategies, common failure modes, and how human-AI *leadership* (i.e., who drives the next step) evolves over time. We find that participants' stated intention to work *cooperatively* with the assistant often collapses under real-time competitive pressure: many teams shift toward end-to-end delegation, letting the AI take on entire tasks rather than using it for incremental support ($\mathcal{F}_6$). Whether this delegation helps depends strongly on expertise. High-expertise users tend to employ higher-quality prompts with richer technical details and more effective prompt engineering strategies to achieve better outcomes while novices used low-quality prompts, leading to errors and task failures ($\mathcal{F}_7$). Lower-expertise users were susceptible to low-risk, low-success rate strategies and adopted `"answer shopping"` through of repetitive prompting ($\mathcal{F}_8$). This divergence is not inevitable. We observed that some participants with zero or limited CTF knowledge were able to use the AI as a constructive learning scaffold that allowed them to rapidly acquire missing domain concepts and solve challenges they would otherwise be unable to approach ($\mathcal{F}_9$).

RQ_3: How do autonomous CTF agents compare to humans on the same set of challenges? Can autonomous agents outperform humans, and what are the limits?

• *Empirical comparison between Human and Autonomous AI agents (RQ_3)* We evaluate four autonomous agent frameworks paired with three Claude-family models (12 configurations total) on the same fresh challenge set used in the live competition, and compare their performance against human teams. We find that advanced agents can outperform most human teams while requiring only $\sim$1/5 of the cumulative runtime, with the strongest configuration reaching `4900` points (second among the top-10 human teams) at a cost of $96.32 in API usage ($\mathcal{F}_{11}$). Performance varies sharply across agent designs: agents that combine long-horizon planning with robust interactive tool support consistently perform best, whereas restricted tool wrappers and fixed tool sets correlate with early plateaus ($\mathcal{F}_{12}$). At the same time, backbone model capability sets the ceiling; under weaker models, different frameworks converge to similarly low performance ($\mathcal{F}_{13}$). We also find that challenges that are difficult for humans are not necessarily difficult for agents, and vice versa. Some challenges that are operationally hard for agents (e.g., requiring intensive environment interaction) are comparatively manageable for humans, and vice versa ($\mathcal{F}_{14}$). This mismatch motivates human-in-the-loop *pair hacking*, where the agent runs autonomously for high-throughput work while humans provide sparse steering and verification to overcome brittleness ($\mathcal{F}_{15}$).

2 Background

CTF competitions are a widely used educational tool to help students, practitioners, and security enthusiasts demonstrate and polish their skills at solving security challenges. CTF challenges cover a broad spectrum of security domains such as cryptography, reverse engineering, web exploitation, and forensics. They are inspired by real security vulnerabilities, but each CTF competition varies in challenge design. It is rare that the exact same solution for one CTF challenge can be re-used in a future CTF. These factors make CTF competitions a valuable testbed for investigating human-AI collaboration.

To elaborate, most CTF competitions are time-limited and follow a Jeopardy-style format. The same set of challenges is provided teams that contains a challenge description and challenge artifacts (e.g., binaries, source code, packet traces, disk images, etc.). Participants are expected to analyze challenge artifacts, potentially interact with live sandboxed services (e.g., a vulnerable web or network endpoint), and identify the relevant weakness which will lead to solving the challenge. A partcipant prove they have solved the challenge by recovering the "flag", often a secret token or text string hidden in the challenge, and submitting it. Upon solving a challenge, participants gain the points associated with the estimated difficulty tier of the challenge (e.g., easy, medium, and hard challenges). At the end of the CTF, the team with the most points wins, with time-to-solve being used as a

common tie-breaker if needed.

CTFs can be further tailored to scale in difficult according to the target audience. K-12-oriented CTFs are considered to be the easiest challenges leveraging well-known security vulnerabilities with only a few exploitation steps (e.g., picoCTF [34]), followed by university-level CTFs that represent moderate difficulty (e.g., the CSAW CTF [35]), followed by expert-level CTFs (e.g., the DEF CON CTF [36]). As we describe in detail in § 4.1, this study focuses on an in-person university-level CTF that falls on the higher end of the difficulty spectrum, i.e., it has a larger number of expert-level challenges than a typical university CTF.

3 Related Work

Software systems are growing in size and connectivity, which increases the attack surface and the cost of manual security analysis [31]. To scale security analysis beyond what humans can handle alone, programs such as the DARPA Cyber Grand Challenge and the DARPA AI Cyber Challenge (AIxCC) have accelerated interest in using AI, particularly LLMs/AI-agents, to help detect, exploit, and fix software flaws [37, 38]. In this context, CTF competitions have emerged as the de facto benchmark for evaluating the capabilities of AI agents in cybersecurity tasks, as CTFs offer diverse security tasks in a controlled environment and have clear success criteria. This shift is also reflected in the emergence of AI-first CTF competitions, such as Hack The Box's *Neurogrid CTF*, which explicitly organized around deploying AI agents with MCP integration for challenge solving [39].

Recent work on leveraging AI for CTFs has progressed along three directions: benchmark construction, autonomous agent design, and training pipelines. On the benchmarking side, NYU CTFBench curates 200 challenges from prior CSAW CTF competitions at roughly university-level difficulty [23]. Intercode-CTF provides 100 problems collected from PicoCTF, a large-scale security competition aimed at high-school-level participants [24]. CTFKnow [22] measures CTF-relevant technical knowledge using thousands of multiple-choice and open-ended questions.

Beyond static benchmarks, researchers have built autonomous CTF agents that couple LLM reasoning with tools, memory, and iterative environment interaction (e.g., ENIGMA, CRAKEN, and Cybench) [27, 29, 30]. These agents run in a sandbox with access to challenge artifacts and iteratively follow a plan–act–observe loop (reasoning about the steps, executing commands (e.g., Bash/Python, tool calls), and observing outputs as feedback for the next step) until they submit a valid flag or exhaust their allotted turns or resources. To further improve agent performance, training platforms such as CTF-DOJO collect

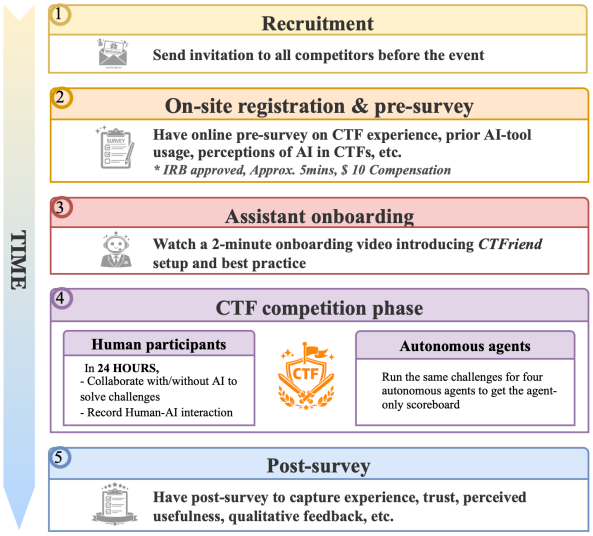

Figure 1: User Participation Workflow

and synthesize executable agent trajectories in realistic cybersecurity environments to support LLM training or fine-tuning on offensive cybersecurity tasks [31]. Our work is different from prior CTF studies by focusing on how humans use AI in practice during live problem-solving. This lens matters because real-world impact depends not only on AI capability in isolation, but also on how practitioners integrate AI into workflows, including what they delegate, how they provide context, how they verify outputs, and how they recover from errors. By combining a live CTF study of human perceptions and interaction logs with a controlled comparison of humans and autonomous agents on the same challenges, we provide empirical insights for designing and evaluating effective human-in-the-loop AI systems for security.

4 Experimental Setting

4.1 A Live University-level CTF Event

We hosted a live, on-site CTF competition on campus and conducted our experiments in this setting. The event included 17 newly designed challenges spanning 5 CTF categories, including forensics, cryptography, reverse engineering, web exploitation, and others. All challenges were newly designed by a third-party security firm with no prior exposure or reuse. Difficulty levels were assigned by the challenge authors based on standard CTF practices, and were further validated by internal CTF experts through pilot testing. Challenges were grouped into three difficulty tiers: `easy`, `medium`, and `hard`, worth `300`, `500`, and `1000` points respectively, for a total of `7700` available points. The distribution can be found in

[40]. This was a university-level competition: most challenges were comparable to (and slightly harder than) typical CSAW-level tasks, with 1-2 challenges closer to picoCTF-level and 2-3 challenges approaching DEFCON CTF-level difficulty. All challenges were authored specifically for this event with support from a security firm and were not previously released, reducing the risk of contamination from existing writeups.

4.2 Study Procedure

This section describes the end-to-end study workflow (Figure 1). For each participant, the study lasted approximately 24.5 hours and consisted of: a 5-minute informed consent process, a 2-minute AI-assistant usage training video, the live CTF competition phase (24 h), and a 20-minute post-study survey. All procedures were approved by our institution's Institutional Review Board (IRB). Participants received $10 compensation for completing two surveys. Our study consists of the following five steps:

❶ **Recruitment.** Prior to the event, we sent an invitation to all competition participants to ask whether they were willing to take part in the study. We recruited 41 participants, with demographics described in § 5.1.

❷ **Onsite registration and pre-survey.** Before the game began, we set up a registration desk where interested participants enrolled in the study. Participants first completed an online pre-survey (approximately 5 minutes) covering CTF experience, prior AI-tool usage, and initial perceptions of AI assistance in CTFs.

❸ **Assistant onboarding.** After the pre-survey, we created an account for each participant to access our CTF AI assistant `CTFriend` (§6.1.1). The assistant provided access to a range of open-source and commercial language models (details in Appendix §B). We also provided a **2**-minute onboarding video demonstrating the interface and expected usage of `CTFriend`.

❹ **CTF Competition phase.** The CTF competition ran for *24 hours*. Participants collaborated with AI to solve challenges. Our backend recorded all human-AI interactions (e.g., prompts and model responses), enabling our log-based analysis of collaboration in **RQ**$_2$ (§ 6).

Parallel autonomous-agent evaluation. In parallel with the human competition, we ran multiple autonomous AI agents on the same challenge set to produce an agent-only scoreboard. These experiments support **RQ**$_3$ by enabling a direct comparison between human and autonomous performance on the same fresh challenges (§ 7).

❺ **Post-survey.** After the competition ended, we distributed a post-CTF survey to all participants to capture their experiences using AI assistance during the event, including perceived value, trust, and feedback. These responses support our survey-based analysis in **RQ**$_1$ (§ 5).

(1) Grade

	N	%
Undergraduate	34	82.9
Graduate	6	14.6
K-12 / High School	1	2.4

(2) Major

	N	%
Computer Science	23	56.1
Cybersecurity	8	19.5
Info Sys. & Engineering	4	9.8
Other	6	14.7

(3) CTF Domain Expertise

	E_S	N	%
Zero Expertise	0-1	12	29.3
Novice	1-2	10	24.4
Intermediate	2-3	16	39.0
Expert	>3	3	7.3

(4) AI Usage Expertise

	E_{AI}	N	%
Zero Expertise	0	5	12.2
Novice	(0,5]	17	41.5
Intermediate	(5,10]	10	24.4
Expert	>10	9	22.0

(5) Prior CTF Experience

# Prior CTFs	N	%		N	%
0	17	41.46	5 — 8	3	7.32
1 — 4	19	46.34	8 +	2	4.88

Table 1: Participant demographics and expertise.

5 RQ$_1$: Users' Perception of AI in CTF

RQ$_1$ examines how participants perceive AI assistance in CTFs (§ 5.2) and analyzes how their performance varies in the context of their backgrounds (§ 5.3).

5.1 Methodology

Our survey design is guided by key questions that intuitively emerge when we consider user perceptions of AI assistance in security-oriented tasks (see details in Appendix §A). For instance, what do participants expect AI tools to accomplish in a CTF setting, and how do these expectations differ after the competition? What aspects of interacting with an AI assistant do participants find helpful versus frustrating during a competition? After hands-on exposure, do participants intend to incorporate AI assistance in future CTFs or broader security workflows?

Participants' Demographics To provide a clearer understanding and categorization of the study participants (N=41), we asked participants to report prior CTFs attended (experience) and rate their proficiency across the different challenge categories in the pre-survey. We then averaged these ratings for each individual (E_S), then mapped participants to four categories (expertise). We also asked users to self-report their AI expertise in terms of number of security challenges solved with AI (E_{AI}. To mitigate self-reporting bias, we evaluated them for consistency, with one participant subsequently excluded (see construct validity in Appendix §A). Generally, we consider zero-expertise and novice participants to have `low expertise`, and intermediate and expert participants to have `high expertise`. Table 1 shows group details.

5.2 Results: Participants' Reflections

Participant Expectations vs. Reflections To evaluate participants' expectations, beliefs, and understanding of AI, we have analyzed the data gathered in the pre-CTF survey by demographic group. We then compare each analysis point to the participants' reflection data gathered in the post-CTF survey. Thus, we can observe how different demographics perceived AI use in security scenarios before the competition, and how those understandings changed after experiencing the live event.

• *Expectation, trust, and attitude toward AI use:* We measure how participants' perceptions of AI shift after hands-on use during the onsite CTF. Overall, participants initially overestimated how much the AI would improve their ability to solve challenges; after the event, expectations for AI solve counts dropped by $\Delta 0.47 \downarrow$, which aligns with the ineffective collaboration behaviors we later observe in logs (e.g., over-delegation and answer shopping; see RQ_2 in §6). Participants' also reported that suggested steps were often difficult to follow under time pressure and occasionally contained hallucinated or misleading details (see $\mathcal{F}_2$). Overall, attitudes were split, with 44% of participants reporting they still plan to use AI in security, viewing it as a fast learning tool, whereas the others reported reluctant due to perceived performance limits and the risks of relying on AI outputs.

Finding 1 ($\mathcal{F}_1$) – Following this CTF competition, participants' perceptions of AI performance on challenges and trust in AI output decreased.

• *Error and Frustrations:* The most frequently observed errors reported in the post-CTF survey were flawed reasoning and hallucinations, which most users reported as a time waste in the competition. Additionally, users reported errors due to AI ethical constraints (i.e. safety constraints, content moderation, or value alignment), with several users singling it out as the most frustrating part of using AI assistance in the competition. However, it is possible that many participants lacked full understanding of what constituted an error or why it occurred, as evidenced by autonomous agents (§ 7.2 not encountering the same issues. This is discussed further in § 6.

Finding 2 ($\mathcal{F}_2$) – Participants identified flawed reasoning, hallucinations, non-working code, and ethical guardrail restrictions as common AI errors that served as points of friction and time loss in the competition.

• *Understanding and Satisfaction:* Similarly, participants reported that AI frequently failed to provide complete, correct and actionable solutions. As shown Figure 5, most participants rated the AI as moderate to low in its ability to generate complete, correct, and actionable solutions. However, participants also reported that AI demonstrated a strong ability to understand user inputs and provide clear and easy-to-understand responses. It is important to note, however, that these actionability and completeness grades may be partially due to users' inability to implement the challenge solutions, which is discussed further in § 6.

Finding 3 ($\mathcal{F}_3$) – Participants reported AI had solid understanding of user input and clarity in output, but solutions lacked in completeness and actionability.

5.3 Results: Expertise and Performance

Competition Results To determine the significance of participants' backgrounds on competition performance, we analyzed users' AI and CTF expertise in conjunction with their individual scores from the competition. Discussion of individual score calculation is provided in Appendix §C.2.

• *Prerequisites for Success:* Unsurprisingly, users who reported high CTF and AI expertise generally scored the highest in the competition. Interestingly, this pattern held regardless of prior CTF experience. For example, three of the top four and 8 of the top 10 highest-scoring individuals in the competition reported attending two or fewer prior CTFs, four of whom reported that this was their first CTF. Meanwhile, all but one of these participants reported at least intermediate-level domain expertise, with the highest-scoring individuals having the highest domain knowledge of this subgroup. For a specific example, User 9, who competed on a high-scoring, three-person team, reported the highest level of domain expertise and accounted for 60% of their team's final score despite having attended the fewest prior CTFs (1).

Finding 4 ($\mathcal{F}_4$) – Participants who reported higher levels of AI and CTF domain expertise saw the highest individual scores in the competition among participants, even when having less direct CTF experience.

While users who reported higher levels of AI and CTF expertise achieved the highest scores, 64% of participants who reported low CTF expertise but high AI expertise were still able to achieve intermediate to high scores in the competition, and 91% completed multiple challenges. This suggests that their AI expertise may have partially made up for their lack of domain expertise, highlighting the value of AI aid in CTFs. In contrast, 85% of participants who reported neither AI nor CTF expertise were only able to complete one or fewer challenges. Interest-

ingly, 2 of the 3 participants who reported at least intermediate CTF expertise but novice or lower AI expertise were only able to complete one or fewer challenges.

Finding 5 ($\mathcal{F}_5$) **–** Participants who reported low CTF domain expertise but high AI expertise were able to achieve high scores in the competition, suggesting the value of AI in filling in user knowledge gaps.

6 RQ_2: Human-AI Collaboration

RQ_1 characterizes participants' perceptions of AI assistance in CTFs. RQ_2 goes one step further by examining how participants *actually* collaborate with AI during security-relevant problem solving. To address RQ_2, we developed and deployed an instrumented AI assistant, `CTFriend`, to record human-AI interactions. In total, we collected **2,299** messages across **168** chat logs from **38** participants. We then qualitatively analyzed these logs, yielding **5** findings on *emergent interaction patterns*, *successful collaboration strategies*, and *AI risk and reward*.

6.1 Methodology

6.1.1 Design and Deployment of `CTFriend`

To study human–AI collaboration during a live CTF, we built and deployed `CTFriend`, a web-based application made available to participants. Importantly, `CTFriend` is not intended to operate as an automatic CTF solver, but rather as an AI assistant, supporting participants in challenge solving by providing access to Claude-family models (Sonnet 4.5, Opus 4.1, and Haiku 3.5), while keeping flag submissions under human control and allowing researchers to monitor and collect human–AI interaction data. We release the `CTFriend` code in [40].

System Overview. Figure 2 presents an overview of `CTFriend`'s architecture. Specifically, (❶) The Streamlit-based web UI serves as the primary interaction layer, providing a conversational interface that mirrors commonly used AI assistants and persistently displays conversation history to support iterative and multi-turn problem solving. (❷) The AI agent layer orchestrates interactions with multiple LLMs through a unified interface, with API access managed by the backend, allowing participants to use diverse AI capabilities without providing their own credentials and reducing friction in time-constrained CTF settings. (❸) To support CTF-specific problem solving, the agent is equipped with a modular MCP tool layer and a retrieval-augmented CTF knowledge base. (❹) All user interactions, conversation histories, and feedback signals are persistently stored in the database layer, supporting systematic analysis of human–AI interaction behaviors. (❺) Finally, a dedicated monitoring and visualization stack provides real-time insights into system health and application usage, ensuring reliable operation during live competitions and comprehensive observability for empirical study. The implementation details are in Appendix § B.

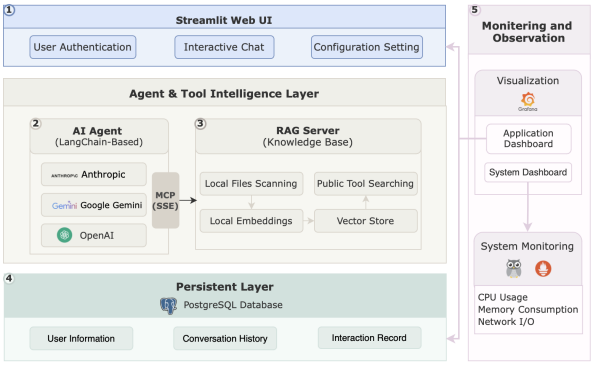

Figure 2: Overview of CTFriend

6.1.2 Qualitative Analysis

During the competition, participants were granted free access to `CTFriend` but were also allowed to use other AI assistants if they want. We then analyze all human-AI interaction logs using a two-pronged qualitative approach: a codebook-based analysis to systematically identify recurring collaboration patterns and error modalities at scale, complemented by an expert case review of selected logs to assess domain-specific technical correctness of AI's responses and user comprehension.

Coding protocol. To analyze a wide set of concepts, we adopted a hybrid inductive-deductive coding protocol.

• *Code System Development:* First, we developed deductive codes based on our survey questions and an existing taxonomy of AI errors and prompt engineering strategies. We then performed inductive coding in two phases: first, we executed a thematic exploratory phase where our coder performed an initial coding of a randomly-selected subset of chat logs until saturation was reached (no new codes for 50 user-turns). Next, we executed a confirmatory phase on 40 randomly-selected logs, where the coder refined content analysis subcodes and definitions based on the observed themes.

• *Coding and Validation:* Upon completion of the exploratory phase, the codebook was frozen and main coding effort performed. Finally, after 21-day washout interval, the coder performed a re-coding of a 50 randomly-selected logs, resulting in a Positive Agreement = 0.74 and median $J_i = .79$ with IQR = 0.17 on significant codes (i.e., codes used in analysis/findings). All coding was performed blind to prior labels, hypotheses, and participant

condition. In total, this process took over 170 hours and uncovered 9312 occurrences of 72 codes and subcodes.

• *Coding Unit:* We coded user prompt-specific behaviors at the user-turn level and AI response behaviors at the assistant-turn level. Other patterns were coded at the episode level, where an episode corresponds to a single user task. A definition/frequency codebook is provided in the artifact (§9.4), with additional information in Appendix §C. The complete code system and protocols and can be found on the website [40].

• *Code and Log Analysis:* Coding and analysis was performed using MaxQDA [41]. To account for variation in log length, we report prevalence as participant-normalized rates (median fraction of eligible turns/episodes per participant). For calculations of co-occurrence or odds ratio, operationalized variables are provided with corresponding uncertainty metrics in Appendix §C. To deepen our analysis, we also invited the author of the CTF challenges to perform an expert review of six representative logs from intermediate and high-scoring teams and nine autonomous agent logs from § 7.2 to provide insight on the critical advantages that helped the best teams stand out in the competition ($\mathcal{F}_8$).

6.2 Results: Interaction Behaviors

We identified several notable interaction patterns between users and AI during our study, including some that demonstrate a likely change in behavior patterns due to the time-constrained nature of the CTF.

Emergent Interaction Patterns. In the pre-CTF survey, we asked participants to report their *expected* collaboration patterns with AI. We then compared these self-reports against their real-world behavior observed through our qualitative coding of the interaction logs. This comparison reveals a clear gap between what participants believed they would do and what they actually did during the competition. Specifically, while *trial-and-error* was the most common interaction pattern reported by users (80%) in the pre-CTF survey, *delegation* was the dominant pattern that emerged during the competition. Specifically, the most common observed strategy was a user providing the full challenge prompt along with a simple instruction such as *"solve this"* or *"how can I do this"*. Additionally, although *collaborative refinement*, *confirmation-seeking*, and *rejection of suggestions* (definitions in the codebook [40]) were reported at moderate rates in the pre-CTF survey (61%, 63%, and 41%, respectively), the actual prevalence of these patterns was much lower (only 16% on average, compared to 38% for delegation). Both observations suggest a breakdown between participants' perspective on their interaction patterns and the empirical evidence.

Finding 6 ($\mathcal{F}_6$) – Although participants expressed the intention to collaborate iteratively with AI, in practice they more often delegated full tasks to the agent.

Successful strategies and Failure mode. As discussed in §5.3, most participants who reported intermediate- or expert-level AI expertise achieved moderate-to-high scores in the competition, while the highest-scorers reported high expertise in both AI and CTF domains. We observe that this driven primarily by two factors: prompting strategy and domain-knowledge injection.

• *Prompting Strategy:* Users who reported greater AI expertise more frequently used prompt-engineering strategies (66% prevalence) and more often achieved successful outcomes in their episodes (`SucR` = 62%) vs low-expertise users who saw 23% and 27% respectively. The most common strategy was *chain-of-thought*, where users collaborated with AI to explore an challenge over multiple logical steps. For reference, representative examples of high- and low-expertise prompts and subsequent agent responses are shown in Figure 3. Additionally, participants who reported zero or novice-level AI expertise also saw higher prevalence of AI errors (42%) compared to experts (17%). This pattern held across all error categories, but especially for guardrail errors, 91% of which occurred in novice AI users' logs. This indicates that much of the frustration reported by participants in §5 may not have been due to technical limitations of the model, but user error. For example, a participant asking *"what is <target>'s ssn?"* led to guardrail activation and AI task refusal.

Finding 7 ($\mathcal{F}_7$) – Users with high AI expertise employed effective prompt engineering strategies and saw higher success rates while novices used low-quality prompts and experienced more errors and task failures.

• *Prompting Rabbit-Holes:* In the pre-CTF survey, most participants reported preferring to fully understand AI-provided code or system commands before running them. In practice, however, participants, including those that reported high AI and CTF expertise, typically executed suggested commands or code immediately then replied with the resulting system output. We observe this behavior in (i) short latencies between assistant command/code suggestions and the user's subsequent system-output message (median $\Delta t = 23$s), and (ii) the extremely low incidence (<1%) of users rejecting AI-suggested commands after review (*User: Command/Code Rejection*). From their manual analysis, the CTF author suggested that this manifested as the user essentially serving as an inter-

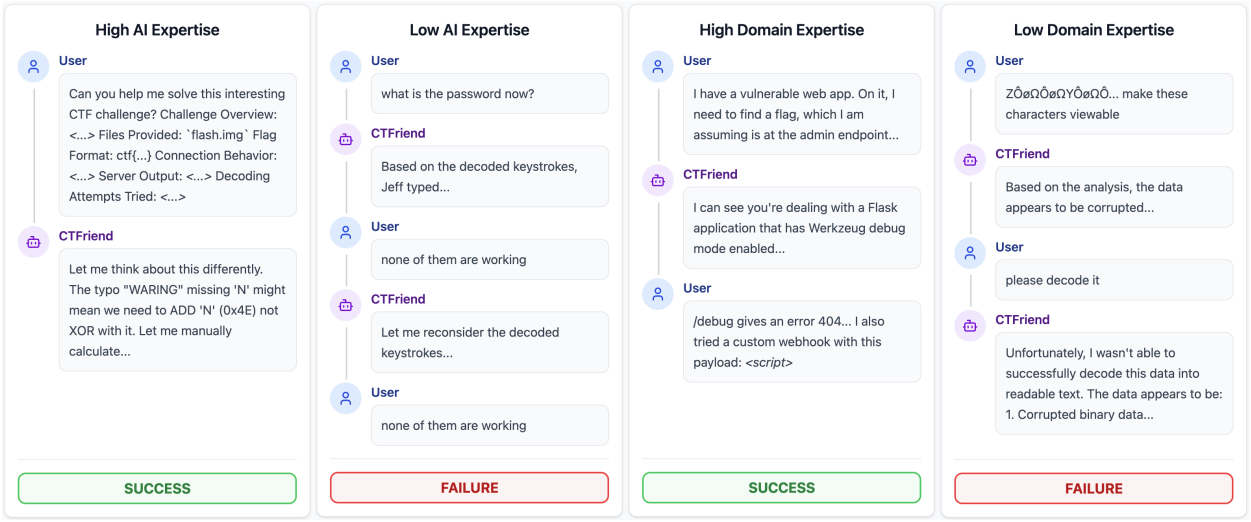

Figure 3: Representative Prompt Examples

face between the AI and the system without appearing to assess problem-solving steps critically for themselves. They specifically identified the AI's tendency to suggest next steps as a critical feature of this behavioral loop. The low median time gap (Δt =14s) between the AI suggesting next steps (*AI: Unprompted, Provide Next Steps*) and the user approving or executing them (*User: Simple Approval*) may be indicative of this pattern. On difficult challenge tasks, the expert assessed that users with low CTF expertise became trapped in "rabbit holes" where, after providing insufficient context in an initial task prompt, they engaged in iteration on an invalid solution thread – the same failure pattern the expert had observed in autonomous agents logs §7.2. Additionally, the expert assessed that when users appeared to lack the necessary domain knowledge to proceed in the challenge, interaction quality would degrade. They highlighted examples of participants repeatedly asking the assistant to explain content or making unproductive requests (e.g., *"give the flag"*). According to the expert, once this pattern emerged, the assistant's suggestions became less actionable, reducing the chance of breaking the pattern.

• *Domain Knowledge-Guided Prompting:* As such, the CTF author suggests two complementary value-adds presented by high domain expertise: first, expert users articulated their understanding of the challenge, including hypotheses, candidate directions, and intermediate observations, which helped steer the assistant toward a plausible approach. Second, they provided execution context that the model would not otherwise have, such as selected relevant tool outputs, error messages, and environment details. For example, a high-scoring team contributed substantive domain signals (e.g., *"there appears to be intentional delay in server ping response"*) and incorporated intermediate results into subsequent prompts, whereas a lower-scoring team provided substantially less context alongside command or script output.

Finding 8 ($\mathcal{F}_8$) **–** Users with low CTF expertise were sucked into "rabbit holes" on harder challenges, while high CTF and AI expertise players achieved higher scores due to their ability to inject domain knowledge into the agent's context.

Risks and Rewards In the course of our analysis we observed AI's potential to both cause harm and benefit, depending on how it is applied.

• *The Agentic Slot Machine:* As discussed in Finding $\mathcal{F}_6$, the most common strategy across all players was to delegate the full challenge to the agent. However, an interesting pattern emerged in how that strategy was applied. Specifically, among the lowest-scoring and lowest domain-knowledge teams, it was common to repeatedly delegate a full challenge (`Del2` = 23%). In effect, this is rolling the dice that the LLM's temperature could lead to random generation of a correct solution. Upon manual inspection of these instances, we observed that this pattern was distinctly unstructured: instead of managing a set of distinct contexts, users would offer no instructions outside the challenge description and keep the prompt chain in the same context window - negating the benefits of *self-consistency*. Rather than intentional strategy, this appears to be an emergent example of *variable-reward reinforcement*, more commonly known as the *"slot machine effect"* [42]. We infer this because the behavior saw a lower success rate (`Del2Fail` = 13%) than the overall success rate for novice users (27%), and thus served no

strategic advantage. From a risk-reward perspective, however, it is possible that the chance of achieving success with little effort expended per attempt made this pattern of *"answer shopping"* more appealing to these participants despite the low chance of success. This would align with previous work regarding AI use in time-sensitive tasks, such as quizzes [43]. Additional research has also investigated the addictive nature of this interaction and the tendency for users to become invested in the potential low-effort reward at the expense of more optimal strategies [44].

Finding 9 ($\mathcal{F}_9$) – Participants with lower levels of domain and AI expertise would engage in "answer shopping", chain-regenerating outputs in hopes of a desirable solution, likely due to its low-cost and high potential reward.

• *An Uplifting Tool:* Furthermore, zero-experience and novice participants who engaged deeply with the agent and were more resilient to failure achieved higher scores than their peers. Notably, rather than the assistant solving problems for them, these participants asked questions that allowed them to iterate through challenges. For example, User 21 reported themselves as having no experience in any CTF domain, which was corroborated by coding, with 60% of their tasks *indicative of low knowledge*. Nonetheless, they applied repeated information-seeking, then leveraged the knowledge gained into new tasks for the agent. As such, they were able to complete multiple challenges, and finished with the second-highest score among zero-experience players. In the post-CTF survey, User 21 expressed that their greatest limitation was not knowing what questions to ask. Nonetheless, they reported that the agent helped them solve multiple challenges they would have not been able to otherwise. This theme of AI providing a crutch to some participants and as a learning and performance aid to others is a finding that has been observed in other research concerning AI use in competitive, time-pressured environments [43].

Finding 10 ($\mathcal{F}_{10}$) – Participants with zero or novice-level domain experience were able to use AI to as a learning tool and constructive competition aid if they engaged in high-quality prompting, especially information-seeking, and were more resilient in their problem-solving attempts.

7 RQ$_3$: Agents vs. Human Teams

RQ$_2$ reveals that human-AI collaboration is often constrained by the human side of the loop, including ineffective prompting and gaps in domain knowledge. In contrast, autonomous agents self-direct key parts of the workflow, including prompt construction, tool use, and sometimes even model selection. At the same time, the models themselves have increased their internal knowledge banks, which likely include substantial security knowledge (e.g., Sonnet 4.5 is trained on security-related data [45]), and expanded their thinking patterns and input contexts. This motivates **RQ$_3$**: how do autonomous CTF agents compare to human teams on the same challenge set, can they outperform humans, and what limitations remain?

7.1 Methodology

Benchmark construction. We converted CTF challenges into a machine-readable format. Following the structured schema [23, 30], we created one JSON specification per challenge containing the same information available to the human participants, including the challenge name, description, category, points, and a list of challenge artifacts. We have released our benchmark dataset on [40].

Agents evaluated. We evaluated four autonomous agents: a coding assistant (Claude Code), two CTF-focused solvers (the NYU agent [23]and Cybench [30]), and a proprietary security assistant that was among the top-performing agents in the Neurogrid AI-only CTF [39]. Agent details are provided in Appendix §D.

Experimental protocol and model selection. Each agent was given up to *three attempts* per challenge per model. An attempt terminated when the agent produced a valid flag or exhausted its budget. Models were drawn from the Claude-family (Sonnet-4.5, Opus-4.1, and Haiku-3.5), the same models that were available to human participants via `CTFriend`.

API cost budgets. We set per-challenge API cost limits to bound resource usage. For *Sonnet-4.5*, the budgets were `$3` (first attempt), `$5` (second), and `$10` (third). For *Opus-4.1*, the budgets were `$10`, `$15`, and `$20`. For *Haiku-3.5*, the budgets were `$1`, `$3`, and `$5`. These limits follow the common practice of using an approximately `$3-per-challenge` budget in prior work [29, 30], and we scale budgets across models with slight adjustments to account for differences in per-token pricing (Opus is more expensive and Haiku is cheaper) [46].

7.2 Agents beat most humans

Figure 4 compares 12 fully autonomous agent teams (four frameworks × three models) against the top-10 human teams. Notably, the rankings are based on all CTF players rather than only the compensated study participants. For agents, *"Hours Since CTF Start"* is computed as cumula-

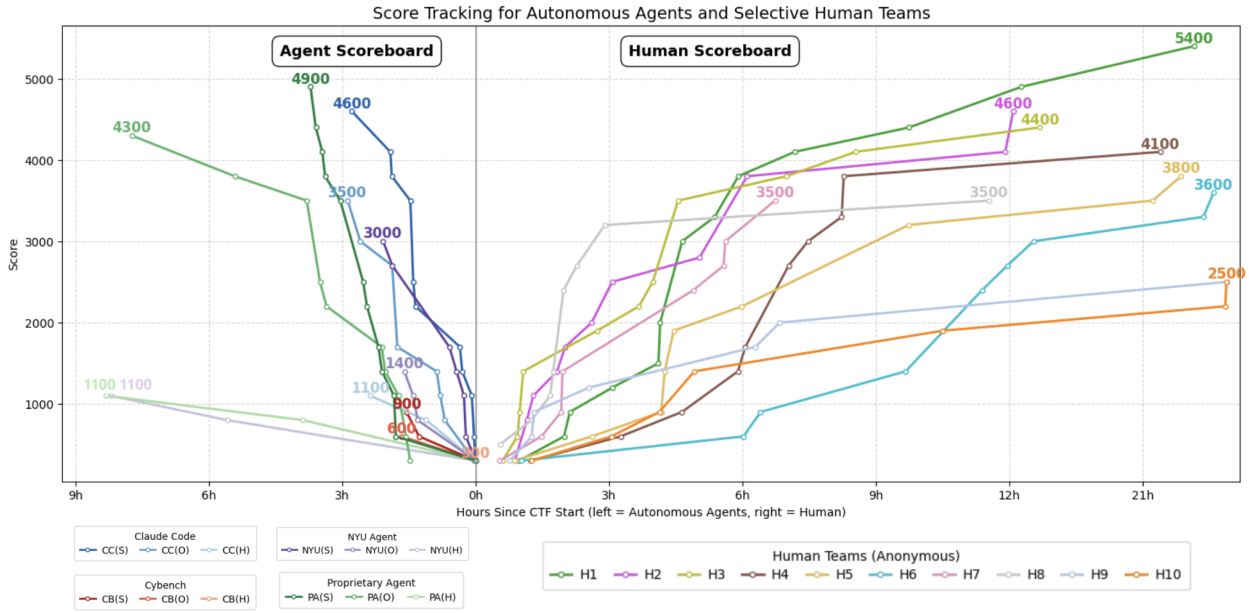

Figure 4: Score Tracking for Agent Teams and Top 10 Human Teams. (S), (O) and (H) mean Sonnet-4.5, Opus-4.1, and Haiku-3.5 models respectively.

tive runtime by sequentially adding (i) the time to the first successful solve for each challenge, or (ii) the full time budget when the agent fails (i.e., exhausts its budget). The strongest AI agent team is the proprietary agent with Sonnet-4.5, which reaches `4900` points (second place on the human scoreboard) in roughly four hours of cumulative runtime, at a cost of $95.32 in API usage. The next best runs are Claude Code with Sonnet-4.5 (`4600`) and the proprietary agent with Opus-4.1 (`4300`), which would rank third and fifth, respectively, among the top-10 human teams. A detailed per-agent analysis of performance and cost is on our website [40].

Finding 11 ($\mathcal{F}_{11}$) **–** An advanced agent outperforms most human teams while requiring only $\sim$1/5 of the cumulative runtime.

From the results and the agent logs, we observe that:

• *Agent design matters when paired with strong models* Across Sonnet-4.5 and Opus-4.1, switching from an assistant-style framework (Proprietary Agent and Claude Code) to a CTF-specific agent framework (NYU agent and Cybench) results in a significant drop in performance. The assistant-style frameworks focus on longer-horizon planning and, most importantly, provide flexible tool interaction. Agents are not restricted to a small subset of custom APIs; they can install tools on demand and execute commands directly in the terminal. This flexibility allows the agent to adapt to unexpected issues during a challenge. In contrast, the lower-performing CTF-specific agents (NYU agent and Cybench) rely on custom tool wrappers (e.g., `read_file` instead of `cat`) that the model may be less familiar with, or on fixed tool sets that cannot adapt to unexpected issues (e.g., a non-interactive network connection tool for an interactive challenge).

Finding 12 ($\mathcal{F}_{12}$) **–** Long-horizon planning and flexible, interactive tool support are the two framework features most associated with top-tier autonomous agent performance.

• *Model capability is the dominant bottleneck.* Across frameworks, Sonnet-4.5 yields the highest success rate across four agents on average (48.53%), followed by Opus-4.1 (35.29%), while Haiku-3.5 collapses performance (14.71%). Even the proprietary agent drops sharply from `4900` (Sonnet-4.5) to `1100` (Haiku-3.5), and other agents converge to similarly low scores under Haiku (e.g., `1100` for NYU and `300` for Cybench).

Finding 13 ($\mathcal{F}_{13}$) **–** Model capability ultimately sets the ceiling. Even strong tooling and architecture cannot compensate for a weak base model, and under weaker models the different frameworks largely converge to similarly low performance.

Model	Agent	Overall Success Rate	Rev			Crypto			Forensics			Web			Other		
			Succ.	Time	Tok.	Succ.	Time	Tok.	Succ.	Time	Tok.	Succ.	Time	Tok.	Succ.	Time	Tok.
Sonnet-4.5	Proprietary Agent	12/17 (70.59%)	2/3 (66.67%)	55.59	6.83	3/3 (100%)	47.02	0.2	3/4 (75%)	34.69	4.53	2/2 (100%)	7.98	0.21	2/5 (40%)	83.68	11.13
	Claude Code	11/17 (64.71%)	2/3 (66.67%)	18.67	4.49	3/3 (100%)	3.00	0.29	3/4 (75%)	12.88	3.04	1/2 (50%)	11.00	3.78	2/5 (40%)	19.00	4.68
	NYU Agent	7/17 (41.18%)	1/3 (33.33%)	34.74	9.26	3/3 (100%)	12.10	3.16	1/4 (25%)	32.00	5.98	0/2 (0%)	10.58	2.19	2/5 (40%)	28.16	6.35
	Cybench	3/17 (17.65%)	0/3 (0%)	12.04	2.73	0/3 (0%)	21.83	2.37	2/4 (50%)	10.22	1.69	0/2 (0%)	11.97	1.45	1/5 (20%)	21.11	1.73
Opus-4.1	Proprietary Agent	10/17 (58.82%)	1/3 (33.33%)	84.18	43.85	3/3 (100%)	11.12	2.85	3/4 (75%)	50.69	22.59	1/2 (50%)	69.49	26.37	2/5 (40%)	66.15	27.28
	Claude Code	8/17 (47.06%)	0/3 (0%)	21.33	27.97	2/3 (66.67%)	17.67	8.24	3/4 (75%)	24.50	15.47	1/2 (50%)	13.50	10.94	2/5 (40%)	40.40	15.43
	NYU Agent	4/17 (23.53%)	0/3 (0%)	35.56	22.48	1/3 (33.33%)	43.36	30.56	1/4 (25%)	38.78	26.23	0/2 (0%)	27.30	17.38	2/5 (40%)	38.97	18.08
	Cybench	2/17 (11.76%)	0/3 (0%)	15.46	6.94	0/3 (0%)	30.25	10.12	1/4 (25%)	15.21	6.85	0/2 (0%)	26.39	7.03	1/5 (20%)	20.08	6.19
Haiku-3.5	Proprietary Agent	3/17 (17.65%)	0/3 (0%)	233.23	9.00	1/3 (33.33%)	150.43	6.08	1/4 (25%))	216.94	6.76	0/2 (0%)	203.91	9.00	1/5 (20%)	202.62	7.61
	Claude Code	3/17 (17.65%)	0/3 (0%)	12.33	0.54	1/3 (33.33%)	9.67	0.58	1/4 (25%)	11.10	0.54	0/2 (0%)	19.50	0.21	1/5 (20%)	36.80	0.67
	NYU Agent	3/17 (17.65%)	0/3 (0%)	144.53	3.60	1/3 (33.33%)	136.01	2.24	1/4 (25%)	58.15	1.29	0/2 (0%)	77.52	1.75	1/5 (20%)	110.67	2.23
	Cybench	1/17 (5.88%)	0/3 (0%)	14.35	0.52	0/3 (0%)	47.49	0.39	1/4 (25%)	12.08	0.27	0/2 (0%)	9.21	0.17	0/5 (0%)	32.90	0.39

Table 2: Performance and cost comparison across models and **challenge categories**. Succ. refers to Success rate, representing what percentage of challenges the agent solved; Time represents the average time spent for one challenge in minutes; Tok. denotes the average token usage for one challenge in dollars.

7.2.1 What makes a challenge hard for the AI?

Challenges that are hard for human teams are not necessarily hard for autonomous agents (Table 3), and the reverse is also true. In general, agents performed better than humans across all categories, except `rev`, where humans showed a higher solved rate. Specifically, for example, on a `crypto challenge` (C2), the solved rate for humans was 51.16% and for agents was 75% (Δ23.84% $\uparrow$). On a `logic-level challenge` (O3), 44.19% human teams solved it, while about 92% agents solved (Δ51.16% $\uparrow$).

When looking deeper, we observed that agents performed best on challenges that minimized task complexity and environmental interactions. In essence, challenge easy for the agents were ones that could be reduced to "write code and run it" (e.g crypto challenges). Where agents struggled, especially the CTF specific agents were challenges that required numerous, complex tooling workflows and stateful, multi-step analysis and interactions (e.g. web challenges). Table 2 breaks down agent teams' performance by challenge category, including success rate and resource usage.

• *Reverse Engineering (Rev):* Rev is among the hardest categories for agents and often incurs the highest time and token costs. Only the strongest Sonnet-4.5 configurations can solve more than a small subset of Rev challenges, and overall agent solved rate is Δ10.47% $\downarrow$ lower than humans. This gap arises because Rev tasks frequently require complex tool workflows and multi-step reasoning where agents remain brittle (see §7.3).

• *Cryptography (Crypto):* Crypto is the most agent-friendly category. Four agent teams (Proprietary Agent and Claude Code with Sonnet-4.5, NYU with Sonnet-4.5, and the Proprietary Agent with Opus-4.1) achieve a 100% solved rate with relatively low time and token cost, while humans are Δ10.47% $\downarrow$ lower. Crypto is comparatively easy for agents because these challenges only involve basic file reading for the initial analysis, followed by writing and running a solver script. LLMs were precisely designed for these tasks [47].

• *Forensics:* Agents substantially outperform humans on forensics challenges (Δ10.47% $\uparrow$ in solved rate). Many tasks in this category involve extracting and transforming evidence from artifacts (e.g., corrupted files) using straightforward, repeatable workflows that involve mostly simple tool sequences. Once the agent creates an analysis workflow, it can iterate much more quickly than a human.

• *Web:* Web challenges are difficult for both the autonomous agents and humans. These challenges require an understanding of network protocol exploitation, packet analysis, and involve intensive environment interaction. The proprietary agent, due to its strong interactive tool support, was able to leverage the capabilities of Sonnet-4.5 to solve both web challenges. With respect to the the NYU and Cybench agents, they failed to solve these challenges even with the strongest model, largely due to limited support for interactive web sessions (e.g., maintaining state, handling multi-step workflows, and reacting to dynamic responses).

• *Other:* This represents the five challenges across OSINT, coding, and hardware. Agents show moderate and relatively consistent performance (roughly 20%-40% success rate). These tasks are often decomposable into structured subtasks (search, transform, implement), although physical hardware constraints can limit automation. Overall, agent teams outperform humans by Δ55.62% $\uparrow$ points in solved rate.

Finding 14 ($\mathcal{F}_{14}$) **–** Based on current model performance, crypto and forensics are comparatively easier for the agents than for humans, whereas reverse engineering remains harder for agents and favors human expertise; these mismatches suggest that human and AI strengths are complementary.

7.3 The power of pair hacking

Agents excel at high-throughput exploration and tedious work [48], but they are brittle when execution depends on environment interaction or when a wrong hypothesis leads to repeated retries. These failure modes are often less challenging from a human perspective. Humans are strong at inferring intent, prioritizing promising directions, and recognizing dead ends. This complementarity makes *pair hacking* a natural winner: a human-in-the-loop workflow in which the agent does the heavy lifting while a human provides sparse, high-leverage guidance.

In our study, we discovered that most agent failures were due to: *(❶) Problem solving loops* Agents can get stuck in unproductive retry cycles, repeatedly exploring the same incorrect path. For example, with Haiku-3.5, we often observed a "reset" behavior where the agent re-read the prompt and artifacts and restarted reconnaissance, effectively discarding prior progress. *(❷) Tooling/Environment limitations* Agents sometimes fail because the required interaction is not supported by the tool layer or execution environment. When this happens, they can spend the remaining budget trying to debug an issue that is outside their control. For example, the NYU agent failed to solve web challenges because the network tool was not designed for interactive web sessions.

The proprietary agent we studied supports both fully autonomous runs and a collaborative *pair hacking* mode. The developers built a UI that allows humans to monitor the agent's trajectory and intervene at critical decision points. Interventions can be lightweight, such as providing a short hint, or operational, such as modifying the execution environment by installing additional tools or performing an interaction on the agent's behalf. We asked the developers to revisit the five challenges their agent failed to solve autonomously and attempt them in pair hacking mode. Using this workflow, they solved two additional challenges, increasing the final score to *5700* and placing first on both the human and agent leaderboards.

Case study. One of the two additional solves enabled by pair hacking was a reverse engineering challenge involving a GUI binary that revealed the flag after a correct password was entered. The autonomous agent attempted to run the binary to observe its behavior, but the sandbox was command-line only (headless), causing execution to fail due to the missing display. The agent then exhausted its budget trying to "fix" execution rather than progressing on the underlying Rev task. In pair hacking mode, a human with reverse-engineering experience provided three lightweight interventions that redirected the trajectory: they (i) broke the execution-fix loop and shifted the agent to static analysis, (ii) resolved an external dependency by manually supplying a blocked dictionary file (`rockyou.txt`) after a firewall prevented download, and (iii) prompted broader code inspection after verifying the decryption logic, which led the agent to discover and replicate thousands of `random()` calls required to recover the correct flag. This case illustrates how sparse human guidance can bypass environment friction, interrupt unproductive loops, and restore forward progress without replacing the agent's core problem-solving work.

Finding 15 ($\mathcal{F}_{15}$) – This pair hacking scenario highlights the advantage of human-in-the-loop workflows: the agent can run autonomously while humans provide sparse steering and verification.

8 Threats to Validity

Internal Validity While we manually cross-referenced individual scores to reduce missatribution (§ 5.3), some participant submissions may reflect flags derived from team effort, potentially overstating individual performance. Additionally, we elected to have hybrid coding performed by a single, trained, expert coder due to the technical nature of the corpus and codes (§6.1.2). This introduces the threat of idiosyncratic coding, which we mitigated by adopting a two-stage exploration-confirmation codebook development process, however some risk remains. Furthermore, although we assessed single-coder stability via a blinded test–retest procedure, intra-coder agreement measures consistency rather than correctness; a degree of residual construct validity risk remains for ambiguous cases.

External Validity As our study combines survey responses with qualitative content analysis of AI chat logs within a specific context, we do not claim statistical generalizability. Instead, our goal is **analytic** generalization/-transferability: we identify recurring patterns and mechanisms that may plausibly apply to similar users, tasks, and deployment contexts, supported by detailed reporting of study context and participant/task characteristics and the full coding system [40]). Furthermore, our coding process is sound, but the nature of the corpus is such that there was variance in the number of logs obtained from each participant, and the number of tasks contained in each log. Although we normalize by episode/turn counts, residual differences in task complexity may still affect code prevalence (§ 6.2). Finally, because LLM behavior varies across model versions, interfaces, and tool configurations, observed patterns may shift over time and may not directly generalize beyond the specific system configuration studied.

9 Discussion and Conclusion

In this section, we distill our findings into four key discussion themes that conclude with practical takeaways for the security community.

9.1 For CTF Organizers

CTFs have historically co-evolved with automation. Our results observe that a similar transition is now underway for AI agents: in our setting, a fully autonomous agent nearly matched the performance of the top human teams ($\mathcal{F}_{11}$). If a CTF's goal is to evaluate security reasoning under pressure, organizers should anticipate that autonomous agents can now solve a non-trivial fraction of standard challenge archetypes, and quickly. A naïve countermeasure would be to add inordinately difficult or convoluted challenges to confuse AI; however, such challenges would also discourage human players, defeating the educational agenda that motivates CTFs. One design adaptation could be to include tasks that are solvable by humans but are *operationally hard* for current agents because they require robust interaction with complex, stateful tooling or environments. For instance, hardware challenges and tasks requiring real-time monitoring are likely to favor human participants. Additionally, intentional misdirection, such as prompt injections in the challenge may confuse AI agents but is often easily recognized by humans, amplifying human advantages further. In our experiments, agents commonly failed from brittle execution and interaction, such as reproducing a challenge in a specific environment, driving a browser workflow with multiple steps, synchronizing with services that require waiting or state transitions, or handling tasks where progress depends on careful, iterative inspection rather than a single-shot script. Such challenges will remain fair to humans if organizers provide clear setup instructions, stable artifacts, and good debugging signals, while resisting fully autonomous "pipeline" solving ($\mathcal{F}_{12}$-$\mathcal{F}_{15}$).

Takeaway 1

CTFs will need to adapt to the use of AI agents and deploy interactive challenges that engage the human players while being operationally hard for AI agents.

9.2 For Cybersecurity Educators

When used as a scaffold rather than a substitute for reasoning, AI assistance can lower the entry barrier for CTF novices ($\mathcal{F}_{10}$), improve skill acquisition, and help novices progress more quickly toward independent problem solving. This was observed especially in participants who prioritized information-seeking and were resilient to failure. However, unscaffolded learning can potentially lead to performance without transfer: students complete tasks more efficiently in the presence of AI guidance, yet may fail to develop skills that persist when guidance is removed. Zhou et al. illustrate this risk in an introductory programming setting: LLM-generated hints improved immediate debugging success, but the advantages disappeared when AI support was withdrawn [49]. Essentially, if an LLM routinely proposes the next action, learners may bypass the metacognitive work of deciding what to do next and why ($\mathcal{F}_9$), weakening strategic planning, error diagnosis, and critical evaluation of evidence. In cybersecurity education, where expertise depends on recognizing patterns across unfamiliar systems and reasoning under uncertainty, such reliance is especially problematic.

AI might be turned into a meaningful benefit, however, as new research argues for theory-driven, adaptive scaffolding that is contingent on learners' demonstrated understanding and that fades as competence increases [50]. For example, a future CTF could include an "educational" division with an AI assistant that could: (i) require learners to articulate their reasoning (e.g., prompting for hypotheses and justification before revealing hints), (ii) provide tiered assistance that starts with conceptual guidance rather than executable steps, and (iii) progressively reduce specificity as the learner demonstrates improvement. Such an approach could preserve the motivational and accessibility benefits of timely help while mitigating cognitive offloading.

Takeaway 2

Our findings demonstrate an opportunity for security educators to adapt existing work in AI scaffolding to the context of solving security challenges, in order to retain the educational value of CTFs and security challenges in general while helping students learn to leverage AI constructively.

9.3 For Emerging Security Practitioners

Our study suggests that alongside traditional domain expertise, *AI literacy* is becoming an increasingly important skill for security practitioners. As we observed, in the AI-assisted CTF setting, the highest-performing teams were not simply those with the strongest CTF background, but those who could maximize AI capability under time pressure ($\mathcal{F}_4$). Consistent with our interaction analysis, participants with strong AI-use skills (e.g., high-quality prompting and structured iteration) were often able to compensate for gaps in CTF domain knowledge and still achieve strong outcomes, whereas some domain-expert participants who struggled to elicit actionable guidance or validate outputs saw their performance degrade despite their underlying expertise ($\mathcal{F}_5$).

Takeaway 3

Our findings show that "knowing security" and "using AI effectively" are now complementary competencies. Next-generation practitioners should be explicitly trained to use AI-enabled workflows to accelerate security tasks while preserving human responsibility for correctness and risk.

9.4 For AI Agent Developers

Our results show that the most effective workflow is often neither "AI assistance" nor full autonomy, but a *human-in-the-loop* pattern in which the agent performs high-throughput work while a human provides sparse, high-leverage supervision ($\mathcal{F}_8$-$\mathcal{F}_{10}$). Autonomous agents excel at rapid reconnaissance, broad exploration, and repetitive tasks, yet we repeatedly observed brittle failure modes that lightweight human intervention can resolve ($\mathcal{F}_{15}$). For example, agents enter unproductive loops, issuing repetitive commands or attempting brute-force methods even if evidence suggests that the current path is unlikely to succeed. Practically, this motivates the injection of a Human-in-the-loop layer in autonomous agentic workflow design for security tasks, which (1) allows humans to monitor progress and re-plan when a strategy stops paying off; (2) expose intermediate evidence and assumptions so humans can quickly verify or correct them; and (3) seamlessly integrate both high-level or precise feedback in the agentic loop. However, building such a human-in-the-loop layer would require us to rethink the balance of autonomy and human intervention in the design of agentic systems, in order to involve the human in a manner that is usable, while also simultaneously maximizing agentic autonomy.

Takeaway 4

At the task of solving CTF challenges, agentic systems work best when there is a human in the loop to help the agent strategize, identify conclusive evidence, and recover from deadlocks. This paradigm is human-AI collaboration in its truest form, and exposes a valuable opportunity for effective use of AI in security that may go beyond CTFs.

Ethical Considerations

This study was reviewed and approved by our Institutional Review Board (IRB). All participants provided informed consent before data collection and were informed that participation in the research was voluntary and separate from participation in the CTF. Participants could compete without joining the study, could choose whether and how to use AI assistance, and could withdraw from the study without penalty.

Cybersecurity research ethics. Cybersecurity research is inherently dual-use as the same techniques that improve defensive capability and education can be abused for unethical and illegal activities. We therefore designed our challenges to limit exposure of participants, non-participants, and other third parties to avoidable risk while preserving the scientific and educational value of observing AI-assisted security problem solving. Furthermore, like all offensive-security education, CTFs carry some risk of teaching skills that may be misused. AI assistance can amplify this concern by lowering barriers to exploration, automation, and exploitation-oriented reasoning. We mitigated this risk through standard CTF safeguards: the activity occurred in a scoped, controlled competition environment; challenges used purpose-built artifacts and infrastructure rather than third-party targets; participants operated under CTF rules and boundaries; and flag submission remained under human control. We also configured the AI assistance around proprietary models with strong safety alignment and provider-level guardrails, reducing the likelihood that the assistant would provide unsafe guidance outside the intended CTF context. When safety guardrails limited responses, we treated those events as part of the empirical record rather than attempting to bypass them.

Compensation. The CTF is a standalone university event with independent cash prizes, and we exclusively recruited participants who were already registered for and attending the competition. In other words, this study is an ancillary product. As such, the $10 compensation is for two surveys of about 15-20 minutes total, not the CTF itself. We offered additional compensation to those who provided additional logs from other AI chatbots at $2 per log. According to the US Bureau of Labor Statistics (BLS) [51], our compensation of $10 for 20 minutes is fair and reasonable.

Affected stakeholders. The primary stakeholders included:

- *Study participants* were exposed to potential risks that included time burden, stress or frustration from using imperfect AI tools under competition pressure, distraction from normal play, learned overreliance on AI outputs, and the possibility that the small external-log incentive could encourage tool use that degraded performance or reduced learning. At the same time, participants could benefit from exposure to AI-assisted security workflows, opportunities for learning and scaffolding, social and collaborative engagement, and enjoyment from experimenting with new tools in a controlled setting.
- *Non-participating CTF competitors* may have also been affected. AI-assisted participants may have changed the

competitive environment by raising the effective level of some competitors and reducing it in others. We reduced this concern by recruiting only from already registered competitors, ensuring that non-participants could proceed normally, and not giving study participants privileged challenge information or direct score-related incentives.
• *CTF organizers* were affected because the study introduced additional operational complexity and may have influenced the character of the event. However, most organizers were directly involved in planning or running the study, allowing research procedures to be coordinated with event rules and logistics.
• *CTF sponsors* are also indirect stakeholders. Although sponsors were made fully aware of the study, they could have been affected by secondary consequences, such as participants having a worse experience, learning less, or becoming less likely to pursue a cybersecurity career. We sought to mitigate these risks by preserving the CTF's educational goals, keeping AI use optional, monitoring participant experience, and reporting findings in ways that support improved future CTF design.

Data handling and confidentiality. We minimized data collection to what was necessary for the study: survey responses, CTFriend interaction logs, submitted external AI logs, competition-related metadata, and any optional qualitative feedback covered by the approved protocol. We avoided collecting sensitive personal identifiers where possible and linked logs to study identifiers rather than participant identities. Raw data access was limited to the research team, and data were stored in access-controlled institutional systems. External AI logs provided by participants were handled under the same confidentiality procedures as CTFriend logs. We report results in aggregate, anonymize quoted excerpts, and remove or generalize information that could enable re-identification. Any released artifacts, quotes, or other data are de-identified and limited to materials needed for scientific transparency and reproducibility.

Open Science

To support our findings, we release study artifacts at https://doi.org/10.5281/zenodo.20309511, including (1) the pre- and post-survey instruments, (2) the qualitative codebook, (3) the `CTFriend` code implementation, (4) TribeCTF 2025 challenge set (in §4.1), and (5) autonomous agent experiment logs (in §5.3). More details are also available on http://tingxuantang-txt.github.io/CTF_website/.

References

[1] P. Liu, J. Liu, L. Fu, K. Lu, Y. Xia, X. Zhang, W. Chen, H. Weng, S. Ji, and W. Wang, "Exploring ChatGPT's capabilities on vulnerability management," in *33rd USENIX Security Symposium (USENIX Security 24)*, (Philadelphia, PA), pp. 811–828, USENIX Association, Aug. 2024.

[2] L. Huynh, Y. Zhang, D. Jayasundera, W. Jeon, H. Kim, T. Bi, and J. B. Hong, "Detecting code vulnerabilities using llms," in *2025 55th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 401–414, 2025.

[3] X. Ma, L. Luo, and Q. Zeng, "From one thousand pages of specification to unveiling hidden bugs: Large language model assisted fuzzing of matter IoT devices," in *33rd USENIX Security Symposium (USENIX Security 24)*, (Philadelphia, PA), pp. 4783–4800, USENIX Association, Aug. 2024.

[4] A. Lekssays, H. Mouhcine, K. Tran, T. Yu, and I. Khalil, "LLMxCPG: Context-Aware vulnerability detection through code property Graph-Guided large language models," in *34th USENIX Security Symposium (USENIX Security 25)*, (Seattle, WA), pp. 489–507, USENIX Association, Aug. 2025.

[5] G. D. Pasquale, I. Grishchenko, R. Iesari, G. Pizarro, L. Cavallaro, C. Kruegel, and G. Vigna, "ChainReactor: Automated privilege escalation chain discovery via AI planning," in *33rd USENIX Security Symposium (USENIX Security 24)*, (Philadelphia, PA), pp. 5913–5929, USENIX Association, Aug. 2024.

[6] X. Du, G. Zheng, K. Wang, Y. Zou, Y. Wang, W. Deng, J. Feng, M. Liu, B. Chen, X. Peng, T. Ma, and Y. Lou, "Vul-rag: Enhancing llm-based vulnerability detection via knowledge-level rag," *ACM Trans. Softw. Eng. Methodol.*, Feb. 2026. Just Accepted.

[7] B. Boi, C. Esposito, and S. Lee, "Smart contract vulnerability detection: The role of large language model (llm)," *SIGAPP Appl. Comput. Rev.*, vol. 24, p. 19–29, Aug. 2024.

[8] F. Weissberg, L. Pirch, E. Imgrund, J. Möller, T. Eisenhofer, and K. Rieck, "Llm-based vulnerability discovery through the lens of code metrics," *arXiv preprint arXiv:2509.19117*, 2025.

[9] W. Bai, Q. Wu, K. Wu, and K. Lu, "Exploring the influence of prompts in llms for security-related tasks," in *Workshop on Artificial Intelligence System with Confidential Computing (AISCC 2024)(San Diego, CA). USA. https://dx.doi.org/10.14722/aiscc.2024.23015*, 2024.

[10] W. Bai, K. Xuan, P. Huang, Q. Wu, J. Wen, J. Wu, and K. Lu, "Apilot: Improving the security and usability of llm code suggestions via outdated api mitigation," in *Annual Computer Security Applications Conference*, 2025.

[11] F. Dong, L. Wang, X. Nie, F. Shao, H. Wang, D. Li, X. Luo, and X. Xiao, "DISTDET: A Cost-Effective distributed cyber threat detection system," in *32nd USENIX Security Symposium (USENIX Security 23)*, (Anaheim, CA), pp. 6575–6592, USENIX Association, Aug. 2023.

[12] Y. Zhang, W. Song, Z. Ji, N. Meng, *et al.*, "How well does llm generate security tests?," *arXiv preprint arXiv:2310.00710v2*, 2023.

[13] W. Peng, L. Ye, X. Du, H. Zhang, D. Zhan, Y. Zhang, Y. Guo, and C. Zhang, "PwnGPT: Automatic exploit generation based on large language models," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, eds.), (Vienna, Austria), pp. 11481–11494, Association for Computational Linguistics, July 2025.

[14] R. Fang, R. Bindu, A. Gupta, and D. Kang, "Llm agents can autonomously exploit one-day vulnerabilities," *arXiv preprint arXiv:2404.08144*, 2024.

[15] Y. Huang, S. Gupta, M. Xia, K. Li, and D. Chen, "Catastrophic jailbreak of open-source llms via exploiting generation," in *International Conference on Learning Representations*, vol. 2024, pp. 13707–13727, 2024.

[16] Y. Kim, S. Shin, H. Kim, and J. Yoon, "Logs in, patches out: Automated vulnerability repair via Tree-of-Thought LLM analysis," in *34th USENIX Security Symposium (USENIX Security 25)*, (Seattle, WA), pp. 4401–4419, USENIX Association, Aug. 2025.

[17] Y. Nong, H. Yang, L. Cheng, H. Hu, and H. Cai, "APPATCH: Automated adaptive prompting large language models for Real-World software vulnerability patching," in *34th USENIX Security Symposium (USENIX Security 25)*, (Seattle, WA), pp. 4481–4500, USENIX Association, Aug. 2025.

[18] S. Wu, R. Wang, Y. Cao, B. Chen, Z. Zhou, Y. Huang, J. Zhao, and X. Peng, "Mystique: Automated vulnerability patch porting with semantic and syntactic-enhanced llm," *Proc. ACM Softw. Eng.*, vol. 2, June 2025.

[19] U. Kulsum, H. Zhu, B. Xu, and M. d'Amorim, "A case study of llm for automated vulnerability repair: Assessing impact of reasoning and patch validation feedback," in *Proceedings of the 1st ACM International Conference on AI-Powered Software*, pp. 103–111, 2024.

[20] Y. Zhang, Z. Jin, Y. Xing, G. Li, F. Liu, J. Zhu, W. Dou, and J. Wei, "Patch: Empowering large language model with programmer-intent guidance and collaborative-behavior simulation for automatic bug fixing," *ACM Transactions on Software Engineering and Methodology*, vol. 35, no. 1, pp. 1–35, 2025.

[21] R. Fang, R. Bindu, A. Gupta, and D. Kang, "Llm agents can autonomously exploit one-day vulnerabilities," 2024.

[22] Z. Ji, D. Wu, W. Jiang, P. Ma, Z. Li, and S. Wang, "Measuring and augmenting large language models for solving capture-the-flag challenges," in *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pp. 603–617, 2025.

[23] M. Shao, S. Jancheska, M. Udeshi, B. Dolan-Gavitt, H. Xi, K. Milner, B. Chen, M. Yin, S. Garg, P. Krishnamurthy, F. Khorrami, R. Karri, and M. Shafique, "Nyu ctf bench: A scalable open-source benchmark dataset for evaluating llms in offensive security," in *Advances in Neural Information Processing Systems* (A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, eds.), vol. 37, pp. 57472–57498, Curran Associates, Inc., 2024.

[24] J. Yang, A. Prabhakar, S. Yao, K. Pei, and K. R. Narasimhan, "Language agents as hackers: Evaluating cybersecurity skills with capture the flag," in *Multi-Agent Security Workshop@ NeurIPS'23*, 2023.

[25] R. P. Buckley, D. A. Zetzsche, D. W. Arner, and B. W. Tang, "Regulating artificial intelligence in finance: Putting the human in the loop," *Sydney Law Review, The*, vol. 43, no. 1, pp. 43–81, 2021.

[26] Y. Zou, J. Liu, and W. Fan, "Ctfagent: An llm-powered agent for ctf challenge solving," *Journal of Information Security and Applications*, vol. 96, p. 104305, 2026.

[27] T. Abramovich, M. Udeshi, M. Shao, K. Lieret, H. Xi, K. Milner, S. Jancheska, J. Yang, C. E. Jimenez, F. Khorrami, *et al.*, "Enigma: Interactive tools substantially assist lm agents in finding security vulnerabilities," *arXiv preprint arXiv:2409.16165*, 2024.

[28] V. Mayoral-Vilches, L. J. Navarrete-Lozano, F. Balassone, M. Sanz-Gómez, C. R. Chavez, M. d. M. de Torres, and V. Turiel, "Cybersecurity ai: The world's top ai agent for security capture-the-flag (ctf)," *arXiv preprint arXiv:2512.02654*, 2025.

[29] M. Shao, H. Xi, N. Rani, M. Udeshi, V. S. C. Putrevu, K. Milner, B. Dolan-Gavitt, S. K. Shukla, P. Krishnamurthy, F. Khorrami, *et al.*, "Craken: Cybersecurity llm agent with knowledge-based execution," *arXiv preprint arXiv:2505.17107*, 2025.

[30] A. K. Zhang, N. Perry, R. Dulepet, J. Ji, C. Menders, J. Lin, E. Jones, G. Hussein, S. Liu, D. Jasper, P. Peetathawatchai, A. Glenn, V. Sivashankar, D. Zamoshchin, L. Glikbarg, D. Askaryar, H. Yang, A. Zhang, R. Alluri, N. Tran, R. Sangpisit, K. Oseleononmen, D. Boneh, D. Ho, and P. Liang, "Cybench: A framework for evaluating cybersecurity capabilities and risks of language models," in *International Conference on Learning Representations* (Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, eds.), vol. 2025, pp. 25094–25243, 2025.

[31] T. Y. Zhuo, D. Wang, H. Ding, V. Kumar, and Z. Wang, "Training language model agents to find vulnerabilities with ctf-dojo," *arXiv preprint arXiv:2508.18370*, 2025.

[32] A. K. Sood, S. Zeadally, and E. Hong, "The paradigm of hallucinations in ai-driven cybersecurity systems: Understanding taxonomy, classification outcomes, and mitigations," *Computers and Electrical Engineering*, vol. 124, p. 110307, 2025.

[33] Z. Mousavi, C. Islam, K. Moore, A. Abuadbba, and M. A. Babar, "An investigation into misuse of java security apis by large language models," in *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, pp. 1299–1315, 2024.

[34] Carnegie Mellon University, "picoctf - cmu cybersecurity competition." https://picoctf.org/, 2026. picoCTF.org is now CyLab Security Academy as of May 2026.

[35] NYU OSIRIS Lab, "Csaw capture the flag (ctf)." https://www.csaw.io/ctf, 2025.

[36] DEF CON Communications, Inc., "Def con® hacking conference - capture the flag archive." https://defcon.org/html/links/dc-ctf.html, 2026.

[37] J. Song and J. Alves-Foss, "The darpa cyber grand challenge: A competitor's perspective," *IEEE Security & Privacy*, vol. 13, no. 6, pp. 72–76, 2015.

[38] DARPA, "Artificial intelligence cyber challenge (AIxCC)." https://aicyberchallenge.com/, 2025.

[39] H. T. Box, "Neurogrid CTF: the ultimate ai security showdown." https://ctf.hackthebox.com/event/details/neurogrid-ctf-the-ultimate-ai-security-showdown-2712, 2025.

[40] "Human-AI collaboration wm-ctf website." http://tingxuantang-txt.github.io/CTF_website/, 2026.

[41] "The top 1 qualitative data analysis software with the best ai integration." https://www.maxqda.com/, 2026.

[42] M. Chóliz, "Experimental analysis of the game in pathological gamblers: Effect of the immediacy of the reward in slot machines," *Journal of Gambling Studies*, vol. 26, no. 2, pp. 249–256, 2010.

[43] J. Zheng, L. Hao, K. Lu, A. Garg, M. Reese, M.-J. Yap, I.-J. Wang, X. Wu, W. Huang, J. Hoffman, *et al.*, "Do students rely on ai? analysis of student-chatgpt conversations from a field study," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, vol. 8, pp. 2796–2807, 2025.

[44] M. K. Shen and D. Yoon, "The dark addiction patterns of current ai chatbot interfaces," in *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, CHI EA '25, (New York, NY, USA), Association for Computing Machinery, 2025.

[45] Anthropic, "Introducing claude sonnet 4.5." https://www.anthropic.com/news/claude-sonnet-4-5, September 2025.

[46] Anthropic, "Claude api pricing." https://platform.claude.com/docs/en/about-claude/pricing, 2026.

[47] S. Joel, J. Wu, and F. Fard, "A survey on llm-based code generation for low-resource and domain-specific programming languages," *ACM Trans. Softw. Eng. Methodol.*, Oct. 2025.

[48] "5 hard truths about ai pentest agents vs humans." https://meetcyber.net/5-hard-truths-about-ai-pentest-agents-vs-humans-0f6e4b05a816, 2025.

[49] Y. Zhou, M. Pankiewicz, L. Paquette, and R. Baker, "Impact of llm feedback on learner persistence in programming," in *International Conference on Computers in Education*, 2025.

[50] C. Cohn, S. Guo, S. Rayala, H. D. Wang, N. Mohammed, U. Timalsina, S. Jain, A. Eeds, M. Deweese, P. J. O. Popp, *et al.*, "Evidence-decision-feedback: Theory-driven adaptive scaffolding for llm agents," *arXiv preprint arXiv:2602.01415*, 2026.

[51] U.S. Bureau of Labor Statistics, "Usual Weekly Earnings of Wage and Salary Workers: First Quarter 2026," Economic News Release USDL-26-0622, U.S. Bureau of Labor Statistics, Apr. 2026.

[52] Y. Sun, D. Sheng, Z. Zhou, and Y. Wu, "Ai hallucination: towards a comprehensive classification of distorted information in artificial intelligence-generated content," *Humanities and Social Sciences Communications*, vol. 11, no. 1, pp. 1–14, 2024.

[53] T. Debnath, M. N. A. Siddiky, M. E. Rahman, P. Das, A. K. Guha, M. R. Rahman, and H. D. Kabir, "A comprehensive survey of prompt engineering techniques in large language models," 2025.

A Survey Design

Pre-survey design The pre-survey (17 questions) covered: (i) CTF background (experience, typical solve volume, self-rated skills by category), (ii) AI familiarity (prior use and comfort with prompting/LLM tools), (iii) expectations and trust (expected performance impact and trust in outputs), (iv) anticipated interaction patterns with AI, and (v) self-reported validation practices.

Post-survey design The post-survey measured: (i) perceived helpfulness (including categories helped most/least and time savings), (ii) perceived output quality (e.g., clarity, completeness, actionability, correctness), (iii) failure experiences and coping strategies (responses to incorrect or unhelpful outputs), and (iv) future adoption and updated expectations.

Construct Validity While we designed our survey to minimize bias or participant misunderstanding (§5.1, there still exists the threat of ambiguous constructs or

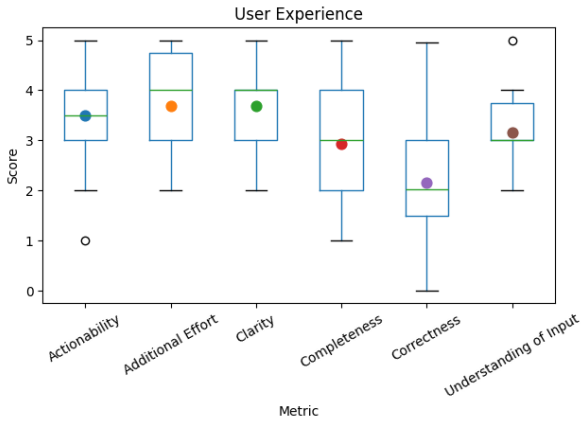

Figure 5: User Experience: user ratings across six evaluation dimensions. Box plots show the median and interquartile range. Open circles represent outliers, and colored dots indicate mean values.

inconsistent interpretation (e.g. participants may interpret "expertise," "confidence," "usefulness," "hallucination," or "trust" differently). Additionally, the length of the post-CTF survey, and with many participants taking it directly following a long competition may have led to acquiescence or satisficing. Thus, we allowed post-survey submissions for two days post-competition. Further threats to construct validity, such as poorly calibrated self-assessment (e.g. over-estimation of expertise), impression management (e.g. overreporting desirable behaviors such as carefulness, skepticism) may be present, but represent part of the value of this study through their comparison to obvservable behavior during the competition (§6.2).

B The Design and Implementation of `CTFriend`

To study human–AI collaboration in live CTF competitions, we design and deploy `CTFriend`, an AI assistant tailored for CTF problem solving. `CTFriend` is not intended to function as an automatic CTF solver; instead, it provides a controlled and observable collaboration platform that supports CTF participants in using AI assistant during competitions, while enabling researchers to observe and analyze human–AI interaction behaviors systematically. We released the platform in [40].

B.1 Design Goals

`CTFriend` is designed to support the empirical study of human–AI collaboration in live CTF competitions, guided by the following design goals:

• *Participant-Centered Collaborative Usability.* `CTFriend` should adopt a conversational interface mirroring commonly used AI assistants, enabling natural and familiar interaction while preserving human decision-making authority. To support effective use in time-constrained CTF competitions, the system should minimize setup overhead and provide a unified interface for accessing multiple LLMs, reducing cognitive burden during live events.

• *CTF-Specific Domain Orientation.* The system is explicitly tailored for CTF problem solving and should provide effective assistance for common CTF tasks and reasoning patterns, improving the relevance of AI assistance in competitive CTF settings.

• *Comprehensive Observability.* The platform should enable the detailed observation of human–AI interactions while also providing real-time monitoring of system health and application usage, ensuring reliable operation and high-quality data collection during live deployments.

B.2 Implementation

`CTFriend` is implemented as a modular, tool-augmented conversational agent centered around a unified reasoning core. The agent is designed to integrate LLMs, external tool services, and persistent state management within the framework.

• *User Interface.* User interaction is handled through a lightweight Streamlit-based frontend, which renders the chat interface, displays conversation history, and captures user input and feedback. To handle conversation history and manage user sessions, a token-based authentication mechanism is used along with a uniquely identifiable token provided to each user. Successful authentication of the user's token results in the establishment of an authenticated session. This session would then handle all interactions and subsequent interactions for the duration of the session. Upon interaction with the frontend, all user input is passed via an API call to the corresponding backend system, resulting in an AI-generated response. Furthermore, runtime configuration parameters, such as LLM provider and model selection, can be selected via the Streamlit-based frontend and are later transmitted to the backend as structured metadata embedded in the API request.

• *Agent Core.* Using the LangChain framework, the responsibility for language model orchestration, conversational context management, and tool invocation falls to the backend agent core system. Support for multiple LLM providers (Gemini, OpenAI, Anthropic) is realized through dynamic client instantiation based on runtime configuration. Additionally, short-term conversational context is maintained in memory and backed to persistent storage, allowing for multi-turn reasoning and persistent conversation history and context, isolated for each conversation.

• *Knowledge Augmentation via Microservices.* A central

aspect of CTFriend assistant is tool-augmented reasoning through the Model Context Protocol (MCP). The agent establishes persistent Server-Sent Events (SSE) connections to tool servers and dynamically registers available tools at startup. During inference, the agent is capable of autonomously invoking tools through structured requests, integrating returned results into its reasoning process, keeping the tool-using logic external to the agent.

A retrieval-augmented generation (RAG) knowledge base is implemented as a standalone MCP tool server, constituting the primary knowledge augmentation mechanism in the system. The RAG service preprocesses local PDF and Markdown documents into semantically coherent splits, embeds them using a locally hosted sentence transformer model, and indexes the resulting vectors in an in-memory Facebook AI Similarity Search (FAISS) store. When invoked, the service performs a rapid and efficient semantic search, returning all relevant documents to the agent, grounding responses in authoritative local knowledge.

• *Management and Monitoring* All long-term interaction data, including user identities, conversation sessions, messages, and feedback signals, is persisted in a PostgreSQL database. By externalizing these data, the agent core remains stateless across executions, enabling reliable recovery.

System behavior and health are continuously observed through an integrated monitoring stack based on cAdvisor and Prometheus. Additionally, Grafana provides visibility into container-level resource usage and application-level interaction patterns.

C Qualitative Analysis

C.1 Coding Protocol

• *Deductive Codes:* AI Error codes were derived from Sun et al. [52]'s classification of LLM errors. While this work is primarily focused on issues regarding communication and social sciences, we believe it to be, with minor adaptations, a high-quality and comprehensive classification of AI errors encountered by participants. Prompt Engineering strategy codes development guided by the following Debnath et al.'s survey of LLM prompting techniques, adapted for user-agent interaction in a CTF environment [53]. Finally, base interaction pattern codes were derived from those developed for the pre- and post-CTF surveys, discussed in §5.1.

• *Deductive Codes:* Unless otherwise indicated, codes are horizontally non-exclusive and multiple codes may be applied to the same coding unit. All units were coded and counted at the lowest subcode. We provide a definition and frequency-only codebook in this appendix, while the full code system with per-code protocols, decision rules, instructions, and examples can be found on the website [40].

C.2 Supplemental Information

• *Operationalized Variable Definitions:*

`SucR`: *Success* as a % of *Success+Failure*. *Resolved but Unknown* excluded. Essentially, "what % of known outcome task episodes were *Success* for the given demographic."

`Del2`: Binary Odds Ratio of *Delegate Full Task* → *Delegate Full Task - Multiple Challenges in Same Context*. Essentially, "what % of *Delegate Full Task* are followed by another *Delegate Full Task* excluding those that co-occur with *Multiple Challenges in Same Context* (i.e., the same challenge was regenerated).

`Del2Fail`: Binary Odds Ratio of `Del2Fail` → *Failure*. Essentially, "what % of `Del2Fail` instances co-occured with a *Failure* in the same task episode."

• *CTF Scores:* We found that team scores were too granular as our analysis showed meaningful variation in behavior and performance within teams. Meanwhile, not all CTF players joined the study, which means that self-reported scores were unavailable for some team members. As a result, reliable team-level expertise measures could not be obtained. Since the CTF platform tracked flag captures per player, we computed an individual score for each participant based on their recorded flag captures. To reduce misattribution, we manually cross-referenced each flag submission with user logs to confirm the submitting player worked on the challenge and that no other teammate completed it. Specifically, 68% of participants competed solo, making individual scores fully accurate in these cases. For team participants, we manually verified the challenges that each competitor engaged with during log analysis and granted the full challenge score only in cases where one individual visibly completed the entire challenge solo, otherwise granting partial credit to all team members. Figure 6 illustrates the links between expertise and individual score.

• *Interaction Length:* Chat length varied wildly among participants, with some participants able to find a flag using only a few prompts, and others appearing to work on a problem over many iterations without finding a solution. Overall, the average chat length was 13.6, with a median of 9, a minimum of 2 and a maximum of 56. The average chat length is skewed by a small minority of participants who maintained the same conversation for most or all of the competition, attempting to solve multiple challenges within the same context window despite being instructed to not do so during onboarding. This behavior was observed exclusively among users who reported zero or novice-level AI experience and co-occurred strongly with low-quality prompts, suggesting that these users may not have had enough experience with AI agents to

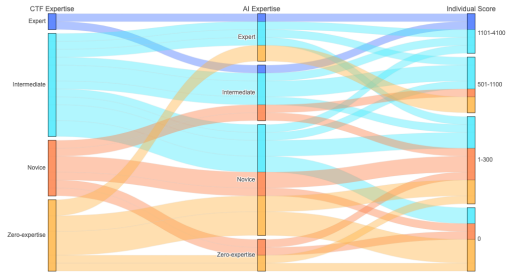

Figure 6: Correlation between Expertise and Score

fully understand the researchers' instructions or general best practices for AI use.

• *Changes Over Time:* Over time, lower-scoring teams and less experienced AI users became more willing to delegate larger tasks to the AI, rather than asking pointed questions. Specifically, prompts generally became larger in scope, allowing the AI more room for interpretation and generating strategies autonomously. For example, early in the competition, a participant would be more likely to ask *"What command should I use in <software> to achieve <specific output>?"*. Meanwhile, the same participant, in a later stage of the competition or facing a more difficult challenge would be more likely to ask *"What should I try next?"*. However, higher-scoring teams saw the opposite effect. These teams were more likely to attempt to delegate the full challenge to the model in the earliest stages of the competition, then increasingly break task into smaller subtasks before providing them to the agent at later stages. This may to be a manifestation of a difference in strategic philosophy, with higher-scoring players prioritizing easy points by delegating the challenges the AI is most likely to complete, then increasing their level of effort and individual involvement as the competition wore on.

D Evaluated agents

• *Claude Code.* We configured the Claude Code assistant as an autonomous agent for CTF challenge-solving. We design a specific prompt to encourage the agent to follow a traditional CTF solving workflow (i.e., interpret the prompt, reason about an approach, iteratively refine, and output a flag when found). To improve robustness, we prepared **six** prompt variants with minor adjustments for common failure scenarios (e.g., warning about decoy flags). All prompts are provided on [40]. We report results from the best-performing prompt configuration.

• *CTF Solving Agents* We used the NYU CTF automation framework [23] and the Cybench agent framework [30], which both provide agents specifically designed for CTF solving. We note the Cybench agent framework expects a different input format than the NYU CTF framework. Our benchmark uses the NYU CTF syntax so some benchmark conversion was necessary, but the information provided to the agent remains the same. It is given a challenge JSON and any files referenced in the JSON.

• *Proprietary agent.* In November 2025, Hack The Box hosted the Neurogrid CTF, an AI-first CTF competition to benchmark AI agents capabilities. We contacted one of the top performing teams and asked them to evaluate their agent on our benchmark dataset. According to them, the agent was simply told to "solve the challenge" based on the provided challenge description and files.

Execution environment. The proprietary agent was evaluated on a Ubuntu 22.04 server with a 64 core Intel(R) Xeon(R) CPU E5-4650 0 @ 2.70GHz and 756GB memory. Claude Code and NYU agent were evaluated on a Debian (ARM 64-bit) virtual machine with an 8-core processor, 7.4 GB of memory, and a 124.8 GB virtual disk, running under a VirtualBox hypervisor with NAT networking. Cybench agent was evaluated on a macOS 14.5 with a arm64 architecture, 8-core CPU, 16 GB of memory, and 460Gi of disk space. For local challenges, the evaluation environment exposed the challenge artifacts (e.g., binaries, source files, or data files) to the agent through an accessible directory. For remote challenges, we hosted the target services on a separate server and allowed agents to connect to them remotely. This setup mirrors how participants interacted with local and remote challenges during the live event while keeping the evaluation procedure reproducible.

Category	Challenge	Human	AI	Δ Performance
Forensics	F1	86.05%	100.00%	Δ44.76% ↑
	F2	32.56%	41.47%	
	F3	11.63%	33.34%	
	F4	0.00%	0.00%	
Cryptography	C1	62.79%	33.33%	Δ10.47% ↑
	C2	51.16%	75.00%	
	C3	25.58%	41.47%	
Reverse Engineering	R1	23.26%	25.00%	**Δ10.47%** ↓
	R2	34.88%	25.00%	
	R3	2.33%	0.00%	
Web	W1	9.30%	8.33%	Δ27.72% ↑
	W2	4.65%	33.34%	
Other	O1	2.33%	0.00%	**Δ55.62%** ↑
	O2	39.53%	50.00%	
	O3	44.19%	91.67%	
	O4	0.00%	0.00%	
	O5	0.00%	0.00%	

Table 3: Human vs. AI **solved rate** differences by **category**. Solved rate represent what percent of teams solved this challenge during on-site competition.