USENIX Security '26 Artifact Appendix: 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

A Artifact Appendix

A.1 Abstract

This artifact provides the complete set of materials needed to reproduce our main claims. The artifact contains five components: (1) the 17 challenges in TribeCTF 2025. The challenge set is released in three formats to support experimental conditions; (2) the full code of CTFriend, the agentic, MCP-based chat assistant we provided to human participants during the live competition; (3) autonomous agent run logs from three agent frameworks (Claude Code, NYU Autonomous Framework, and Cybench), each evaluated with three Claude model variants across all 17 challenges; Two scripts are included to generate score tracking figure and performance comparison table; (4) the qualitative analysis codebooks used to code human-AI interaction behavior and attitude shifts; and (5) the pre- and post-CTF survey instruments distributed to human participants. Each components have their own `README.md` file with more details.

A.2 Description & Requirements

Survey instruments, qualitative analysis codebooks, autonomous agent logs, and challenges are readable directly.

CTFriend is a website-based chatbot with CTF domain knowledge supported. In order to reproduce it, the only requirement is to have a standard computer with Docker and Docker Compose.

Due to the nondeterminism in LLMs, it would be challenging to reproduce the autonomous agent performance; thus, we provide the original run logs of all agents during competition, and there are two scripts used to generate Figure 4 and Table 2 in RQ3 in the paper.

A.2.1 Security, privacy, and ethical concerns

The artifact poses minimal risk to evaluators. The challenge files, logs, codebooks, and surveys are static, non-executable data that can be reviewed safely. ***CTFriend*** is a chatbot-style AI assistant working as a website. Evaluators should follow the instruction to deploy and run it on localhost. All interaction data will be stored locally and can only be accessed by users; thus, the deployment and running process has no specific risks.

A.2.2 How to access

The artifact is available at https://doi.org/10.5281/zenodo.20309511. The only step is to download and unzip it. All components have their specific folders.

A.2.3 Hardware dependencies

The artifact can be tested on a commodity x86-64 (or Apple Silicon) machine capable of running Docker and Docker Compose. The complete artifact is about 7 GB, so roughly 4 GB RAM and 15 GB storage are sufficient.

A.2.4 Software dependencies

For the data components (challenges, autonomous agent logs, qualitative codebooks, survey instruments), only a reader or editor able to open .docx, .json, and .yaml/.yml files are required; no compilation is needed.

CTFriend is fully containerized, and its only host prerequisites are Docker and Docker Compose (installation: https://docs.docker.com/compose/install/). All other dependencies are pulled or built automatically when building Docker containers, including the application stack (Python, FastMCP/Streamlit, a Node.js CyberChef backend, and a RAG knowledge-base MCP server) and the supporting services.

Dependencies for two scripts are listed in the `/Autonomous_Agent_Logs/requirement.txt` file, and Python 3.9+ is necessary.

A.2.5 Benchmarks

The competition/experiment uses the 17 TribeCTF 2025 challenges included in this artifact (`/Challenges`) as the eval-

uation benchmark. These challenges were newly designed by a third-party security company and had never been used before. They are released here in three formats, provided to human players/Claude Code, NYU agent, and Cybench agent correspondingly, to match the competition and experiment conditions.

A.3 Set-up

The artifact has both inspectable parts and executable parts; the setup effort depends on which part is being evaluated. After downloading and unzipping the artifact file,

• *Inspectable components:* require no installation; evaluators only need to open the files in `/Challenges`, `/Autonomous_Agent_Logs`, `/Qualitative_Anaysis_Codebook`, and `/Survey_Instrument` with a PDF reader and a text/.docx/.json/.xlsx/.yaml viewer.

• *CTFriend:* is fully containerized and brought up with Docker Compose.

•*Agent-performance script:* is a small standalone Python tool that reproduces the RQ3 results from the provided logs.

A.3.1 Installation

1. **Enter CTFriend folder**

```
cd /CompleteArtifact/CTFriend
```

2. **Setup environment variables**

```
# Get a local environment file
cp .env.example .env

# Edit environment varibles
nano .env
```

POSTGRES_PASSWORD and TOKEN_SECRET_KEY only need to be changed in live deployment. For evaluation, the defaults are fine. Please set the ANTHROPIC_API_KEY to your own key.

3. **Install Docker containers**

```
# Insure docker is running
docker-compose up --build
```

4. **Generating User Tokens**

CTFriend works on whitelisting tokens, where each user has their own individual token to ID them and their chats within the system. To create this token, use the following instructions.

```
# Log in to the ctfriend container
docker exec -it ctfriend bash

cd app

python3 token_manager.py whitelist [email]
# NOTE: Any email works here. For testing purposes, feel free to use test@test.com
```

Then you will get your specific user token. which will be used later.

5. **Configure Grafana**

• Open your web browser and go to `http://localhost:3000`.

• Log in with "username: `admin`" and "password: `admin`".

6. **Prometheus Setup**

• *Add the Prometheus Data Source:* Go to "Connections" -> "Data Sources" -> "Add new data source" -> Select "Prometheus".

• For the "Prometheus server URL", enter `http://prometheus:9090`.

• Click "Save & Test".

7. **PostgreSQL Setup**

• *Add the PostgreSQL Data Source:* Go to "Connections" -> "Data Sources" -> "Add new data source" -> Select "PostgreSQL".

• *Use the following connection details:* Name: `postgres_db`, Host: `db:5432`, Database name: `metrics`, User: `postgres`, Password: `change-me-to-a-secure-password` or `<password>` `set in .env`, TLS/SSL Mode: `disable`, Version: `15`, Timescale DB: `True`.

• Click "Save & Test". You might need to give Grafana's data source a specific UID like postgres_db if you want it to automatically link with the chat data dashboard.

8. **Setup Dashboards**

• *Import the Dashboards:* Go to "Dashboards" -> "New" -> "Import".

• Upload the `system_monitoring_dashboard.json` file. Select the "Prometheus" data source when prompted.

• Repeat the process for the `chat_data_dashboard.json` file, selecting the `PostgreSQL` data source.

9. **Start CTFriend**

• Go to `http://localhost:8501/ctfriend`.

• Input your specific login token generated in **step 4**.

• Select `Anthropic` as provider and `claude-opus-4-1-20250805` as model.

• Now ***CTfriend*** is ready.

A.3.2 Functionality Test

The tests are designed for CTFriend.
Test 1: Authenticated chat.

• Send "Hello, can you confirm you are working?"
Expected: CTFriend replied normally. An invalid/empty token instead returns "Please enter a valid token.", confirming access control is active.

Test 2: MCP tool use.

• Send "Use CyberChef to Base64-decode: Q1RGcmllbmQ"
Expected: the assistant calls the CyberChef MCP tool and returns CTFriend, confirming cyberchef_api and cyberchef_backend are reachable.

Test 3: RAG knowledge base.

• Send "What is the $MFT in NTFS forensics?"
Expected: a grounded answer drawn from the bundled knowledge base via rag_server.

A.4 Evaluation workflow

The study addresses three research questions: **RQ1** — how participants' attitudes toward AI assistance shift before vs. after the competition (based on survey responses); **RQ2** — how humans interact with AI assistance during problem-solving (qualitative coding of CTFriend interaction data); and **RQ3** — how autonomous agents perform on the same challenges across frameworks and models.

Since RQ1 and RQ2 derive from real human responses and interaction, to protect participants' privacy as presented in the institutional IRB, the corresponding claims are validated by inspecting the provided instruments and codebooks. The RQ3 claim is re-executable: the provided logs and workbook can be re-aggregated with two scripts.

A.4.1 Major Claims

(C1): An advanced agent outperforms most human teams while requiring only about 1/5 of the cumulative runtime. Meanwhile, 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.

(C2): *Crypto and forensics are comparatively easier for the agent teams than for humans, whereas reverse engineering remains harder for agents and favors human expertise; these mismatches suggest that human and AI strengths are complementary.*

A.4.2 Experiments

(E1): [Score Tracking] [3 human-minutes + 3 compute-minutes + 5GB disk]: This experiment is used to generate Figure 4 (score tracking) in the paper based on original autonomous agent logs, manual-recorded work records, and human score tracking records exported by the TribeCTF 2025 committee.
How to: go to the `Autonomous_Agent_Logs` folder and run the script and observe the output figure
Preparation: prepare a new terminal window with pre-installed Python and install `requirements.txt`

```
cd /CompleteArtifact/Autonomous_
    Agent_Logs

# install required dependencies
pip3 install -r requirements.txt
```

Execution:

```
# run the figure generation script
python3 scoretracking.py
```

Results: The output should be the same as Figure 4 in the paper, where we can observe that the most advanced model, PA(S), completed the competition in about 4h (Agent Scoreboard), but human teams usually took more than 20h (Human Scoreboard). In Agent Scoreboard, for each agent (colored in gradient color group), sonnet-4.5 (S) always shows the best performance while Haiku-3.5(H) falls behind.

(E2): [Performance Comparison] [3 human-minutes + 3 compute-minutes + 5GB disk]: This experiment is used to generate Table 2 in the paper based on original autonomous agent logs and manual-recorded work records
How to: similarly, run the `statistics.py` file and observe the output table
Preparation: prepare a new terminal window with pre-installed Python and install required dependencies (should be done in E1)
Execution:

```
# run the figure generation script
python3 statistics.py
```

Results: The results should be the same as Table 2 in the paper. It shows the success rate of each agent by categories. The success rate of Crypto and forensics for most agents with Sonnet-4.5 and Opus-4.1 is more than 50%, and even 100%; however, the success rate of reverse engineering is mostly 0%, and only a few agents are able to achieve 33% or 67% with advanced models.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2026/.