USENIX Security '25 Artifact Appendix: Cloak, Honey, Trap: Proactive Defenses Against LLM Agents

Daniel Ayzenshteyn
Ben-Gurion University, Israel

Roy Weiss
Ben-Gurion University, Israel

Yisroel Mirsky*
Ben-Gurion University, Israel

A Artifact Appendix

A.1 Abstract

The **CHeaT** artifact accompanies our paper ***C**loak, **H**oney, **T**rap: Proactive Defenses Against LLM Agents*. It provides:

- The open-source **CHeaT CLI** for embedding deceptive payloads into text assets.
- **Datasets** compatible with PurpleLlama for reproducing Table 2, Fig. 3, and Fig. 4, as well as the additional dataset `dataset_unicode_honeytokens.json` used specifically for Fig. 2.
- An **evaluation corpus of 11 CTF VMs** used for the end-to-end experiments described in Section 6.2 (Table 3).
- **Token-landmine utilities** (`token-landmines/`) that scan entire vocabularies for rare tokens, trigger hallucinations, and evaluate outputs via GPT-4o.
- An interactive **playground notebook** to load saved *PentestGPT* snapshots, inject new hints/traps, and observe the agent's live reasoning and command stream.

The complete package (~45 GB) is archived on Zenodo (record #15601740) with source mirrored on GitHub.

A.2 Description & Requirements

To reproduce the results from the paper, you will need to use the provided artifact hosted on Zenodo (https://zenodo.org/records/15601740), with source mirrored on GitHub. The artifact includes the complete source code, datasets, and additional evaluation assets necessary to reproduce all experiments described in our paper.

In this artifact, you will find detailed instructions for installation, setup, and usage of the provided tools and datasets. Specifically, the artifact comprises the **CHeaT CLI** tool, evaluation datasets compatible with the PurpleLlama framework, the Unicode honeytoken dataset used for Figure 2, token-landmine utilities, and a Jupyter playground notebook for interactive exploration and experimentation with saved *PentestGPT* snapshots.

*Corresponding Author

The provided evaluation datasets are ready-to-use and correspond exactly to the experiments described in Sections 6.1 and 6.2 of the paper. If you plan to extend this work to evaluate different LLM-based penetration testing agents or experiment with novel payloads, you may need to adjust or regenerate datasets using the methodology described in the paper's experimental sections and the provided **CHeaT CLI** tool.

The provided **token-landmine utilities** were designed specifically for scanning LLM vocabularies, triggering hallucinations, and assessing outputs using GPT-4o. If your experiments involve different LLM architectures or vocabularies, you should rerun these utilities accordingly.

Lastly, while the artifact includes ready-to-run vulnerable CTF VMs, we recommend conducting all experiments strictly within these isolated environments to ensure safety and ethical compliance as detailed in the paper's ethics considerations (Section 9).

A.2.1 Security, privacy, and ethical considerations

All experiments are safe to run on a local machine. However, any evaluation involving autonomous penetration testing agents such as *PentestGPT* must be conducted strictly within the included CTF virtual machines and never against real systems. As detailed in the paper's ethics section, all techniques are purely defensive and designed to mislead malicious agents. The CHeaT tool makes no system-wide changes, operates only on local files, and is fully reversible. The "token-landmine" findings were responsibly disclosed to affected vendors prior to publication and are shared privately with reviewers during the embargo period. Under these constraints, the artifact poses no risk to external systems or user privacy.

A.2.2 How to access

- **Stable archive:** https://zenodo.org/record/15601740
- **Source repository:** https://github.com/Daniel-Ayz/CHeaT
- **Private token-landmine evaluation (review-only):** https://drive.google.com/drive/folders/1YTWdDrkrpaofoH9bAorxN_kR0yQmmPgv

Includes landmine token detection and judgment scripts. This material will be made public after the embargo period (see Section A.2.1).

A.2.3 Hardware dependencies

Most experiments rely on querying LLMs via API and can be run on any standard workstation. Running the CTF virtual machines requires a desktop with basic virtualization support (VT-x/AMD-V) and approximately 50GB of free disk space. For token-landmine detection, local inference depends on the chosen model; our evaluation used an RTX-6000 GPU (24GB VRAM), which sufficed for quantized 70B models. Larger models may require more powerful hardware.

A.2.4 Software dependencies

The CHeaT tool and evaluation scripts require Python 3.9 or later. Dataset-based evaluations rely on the PurpleLlama framework, which should be installed according to its official documentation. Running the CTF virtual machines requires VirtualBox (version 7.0 or higher). To evaluate commercial LLMs such as GPT-4o, an API key (e.g., OpenAI or vendor-specific) is required.

A.2.5 Benchmarks

Datasets are located under `datasets/` for usage with purple llama as documented.

A.3 Set-up

A.3.1 Installation

To install the CHeaT tool, clone the repository (https://github.com/Daniel-Ayz/CHeaT) and follow the instructions in the "Tool Quick Start" section of the main `README.md`. For dataset-based evaluation, install PurpleLlama by following these steps:

1. **Clone the PurpleLlama Repository**

   ```
   git clone https://github.com/
   meta-llama/PurpleLlama.git
   ```

2. **Navigate to the Project and Set Up a Virtual Environment**

   ```
   cd PurpleLlama
   ```

   ```
   python3 -m venv .venv &&
   source .venv/bin/activate
   ```

3. **Install Python Dependencies**

   ```
   pip3 install -r
   CybersecurityBenchmarks/requirements.txt
   ```

4. **Set the Dataset Path**

   ```
   export DATASETS=$PWD/
   CybersecurityBenchmarks/datasets
   ```

You can place our dataset files under a subdirectory such as:

```
$DATASETS/CHeaT/dataset_main.json
```

To run the CTF machines used for end-to-end testing, use the "Import Appliance" option in the VirtualBox GUI to load each `.ova` image and then start the machines.

A.3.2 Basic Test

CHeaT Tool Test: To verify the CHeaT tool is installed correctly, run this basic test from the Tool Quick Start section:

1. Clone the repository and enter the tool folder:

   ```
   git clone https://github.com/Daniel-Ayz/
   CHeaT.git
   cd CHeaT
   ```

2. Create and activate a virtual environment (optional):

   ```
   python3 -m venv .venv && source
   .venv/bin/activate
   ```

3. Install the tool:

   ```
   pip install -e .
   ```

4. Plant a random defense in a test HTML file:

   ```
   echo "<html><body>Hello</body></html>" >
   ./test.html
   ```

   ```
   cheat -action plant -details '{
   "assettype": "web_file",
   "file_path": "./test.html",
   "technique": "random"
   '}'
   ```

5. Verify the installation by listing installed payloads:

   ```
   cheat -action list -type installed
   ```

PurpleLlama Dataset Evaluation Test: For dataset-based evaluations using PurpleLlama, follow the installation instructions provided and refer to the documentation in the `datasets/` directory of the CHeaT repository for specific evaluation procedures with `dataset_main.json`, `dataset_boosted_with_pi.json`, and `dataset_unicode_honeytokens.json`.

CTF Virtual Machine Test: To verify the CTF machines are working correctly:

1. Download the VM files from the Zenodo dataset:
   `https://zenodo.org/records/15601740`

2. Import a CTF machine (e.g., UbuntuX.ova) using VirtualBox's "Import Appliance" option

3. Start the virtual machine and verify it boots successfully

4. Test network connectivity to confirm the machine is ready for evaluation

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): Our payload techniques achieve a Defense Success Rate (DSR) of 55–67% across all evaluated LLMs. This is supported by Section 6.1 of the paper, using the PurpleLlama-compatible datasets (`dataset_main.json` and `dataset_boosted_with_pi.json`). Results are shown in Table 2 and Figures 3 and 4.

(C2): A defended version of the CTF machines achieves 100% DSR against *PentestGPT* when using CHeaT to insert the top 10 technique–datapoint combinations. This is demonstrated by the experiments in Section 6.2 and shown in Table 3.

(C3): There exist Unicode characters that allow honeytokens to distinguish between human and LLM access, based on mismatches in rendering and interpretation. This is supported by evaluations using `dataset_unicode_honeytokens.json` and illustrated in Figure 2.

(C4): Some rare tokens cause LLMs to produce hallucinations or gibberish when injected. This behavior is identified and measured in our token-landmine study and summarized in Table 11.

A.4.2 Experiments

(E1): *Dataset Evaluation with PurpleLlama* [5 human-minutes + 10 compute-minutes]: Evaluates `dataset_main.json` and `dataset_boosted_with_pi.json` using PurpleLlama to reproduce defense success rates (C1). For detailed instructions, please follow the readme under the `datasets` folder.
Preparation: Install PurpleLlama and copy the datasets from `datasets/` into its `data/` folder.
Execution: Run the prompt-injection benchmark script with the selected dataset and model. Enable `-run-llm-in-parallel` for faster evaluation.
Results: PurpleLlama will generate a summary file reporting Defense Success Rate (DSR) by category, as shown in Figures 3–4 and Table 2.

(E2): *Unicode Honeytoken Evaluation [2 human-minutes + 2 compute-minutes]:* Evaluates `dataset_unicode_honeytokens.json` using PurpleLlama to confirm the behavior difference between human- and LLM-visible tokens (C3).
Preparation: Same setup as in E1. Copy the Unicode dataset into PurpleLlama's `data/` folder.
Execution: Run PurpleLlama's benchmark on the dataset using a target LLM (e.g., GPT-4o).
Results: Inspect the model's responses to see whether the password was misinterpreted. Results are summarized in Figure 2.

(E3): *Token Landmine Evaluation [5 human-minutes + variable compute-time]:* Identifies hallucination-triggering tokens (C4) using the `detect_landmine_tokens.py` and `judge_landmines.py` scripts (in `token-landmines/`). Runtime depends on model size and number of tokens scanned.
Preparation: Follow the usage instructions in the `token-landmines/ README.md`. Ensure the GPU is available for large model scans.
Execution: Run the detection script on the chosen model, then label the responses using the judgment script with an OpenAI API key.
Results: The final JSON includes a judgment label ("Yes"/"No") per token. Results are summarized in Table 11.

(E4): *CTF Machine Evaluation with PentestGPT [1+ human-hours per trial]:* Reproduces full agent-based trials with and without defenses to validate the real-world effectiveness of CHeaT (C2).
Preparation: Import the CTF virtual machines into VirtualBox and launch one. Prepare the environment for *PentestGPT* execution with API access.
Execution:
- Step 1: Run *PentestGPT* on the unmodified CTF machine and solve it using the agent's instructions.
- Step 2: Use CHeaT to insert defenses (e.g., top 10×10 entries) into the same machine.
- Step 3: Repeat the process with *PentestGPT* and observe whether it is blocked or misled.

Results: Compare the actions and success rate of *PentestGPT* between the unmodified and defended versions. Results should match Table 3 (100% DSR).

A.5 Notes on Reusability

The provided CTF machines can serve as a reusable testbed for evaluating future LLM-powered penetration testing agents or novel defense techniques. The datasets and payloads included in the repository offer a way to benchmark whether next-generation agents or models remain vulnerable to the same classes of traps. Additionally, the CHeaT tool is modular and can be easily extended to insert custom payloads, support additional asset types, or implement more sophisticated

deception strategies.

A.6 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/usenixsec2025/.