USENIX Security '25 Artifact Appendix: Practically Secure Honey Password Vaults: New Design and New Evaluation against Online Guessing

Haibo Cheng[1], Fugeng Huang[1], Jiahong Yang[1], Wenting Li[2], Ping Wang[1]
[1]Peking University [2]Beijing Institute of Graphic Communication

A Artifact Appendix

A.1 Abstract

This paper evaluates the distinguishability and practical security of existing honey password vault schemes using a large-scale dataset. We also propose a novel scheme based on the Transformer model. Our analysis shows that existing schemes are highly vulnerable to distinguishability attacks, with text classification algorithms (especially pre-trained models) achieving 83.75%–95.79% accuracy. In contrast, our Transformer model generates more plausible decoy vaults, limiting attack accuracy to no more than 64.35%. For practical security against online guessing, our model excels, with an average of only 0.51 accounts cracked per 1,000 attempts, which can be further reduced to 0.11 with two simple measures.

A.2 Description & Requirements

This artifact provides both the source code and a comprehensive large-scale dataset for honey vault research. The code includes implementations of three honey vault schemes (Golla et al., Cheng et al., and ours), existing and proposed attack algorithms, and security evaluations with accompanying plotting scripts. The dataset consists of 7 million password vaults.

A.2.1 Security, Privacy, and Ethical Concerns

Our artifact does not perform any harmful operations on the host machines, nor does it disable any security mechanisms during execution. To protect user privacy, the dataset has been sanitized to remove usernames and directly identifiable information, and is provided in encrypted form with restricted access.

A.2.2 How to Access

Both the source code and dataset are hosted on Zenodo. The source code is publicly available at `https://doi.org/10.5281/zenodo.15612143`, while the dataset is accessible at `https://doi.org/10.5281/zenodo.15646753` under restricted access.

A.2.3 Hardware Dependencies

This artifact requires an NVIDIA GPU with at least 24GB VRAM and CUDA 11.8 support (e.g., NVIDIA RTX 3090). A minimum of 80 GB of system RAM and 50 GB of disk storage are also necessary. While no specific CPU model is strictly mandated, a CPU with at least 20 cores is recommended.

A.2.4 Software Dependencies

Our experiments were conducted and tested on Ubuntu Linux 22.04, utilizing CUDA 11.8, Anaconda, and Python 3.10. All additional software dependencies can be installed during the setup phase.

A.2.5 Benchmarks

None.

A.3 Set-up

A.3.1 Installation

1. Download and extract the code and dataset archives from Zenodo (see A.2.2). Place the three files of the full dataset into the `src/honeyvault/data` directory within the extracted artifact.

2. Navigate to the extracted artifact directory. Execute the following commands to set up a Conda environment and install necessary dependencies:

```
conda create -n py310 python=3.10 -y
conda activate py310
pip install -r req_clean.txt
pip install "protobuf>=3.20.0,<4.0.0"
```

   You may encounter warnings about dependency conflicts; please disregard them, as the artifact has been tested with these specific dependency versions.

3. Generate a Wolfram Alpha API key by following the instructions at https://reference.wolfram.com/language/WolframClientForPython/docpages/basic_usages.html#generate-a-secured-authentication-key. Insert this key into line 17 of `src/honeyvault/CONFIG_RCDF.py`. This key is essential for the vault model fitting process within the scheme proposed by Cheng et al. (2021).

A.3.2 Basic Test

Upon successful installation, verify the artifact's basic functionality by executing the following command:

```
python src/honeyvault/__main__.py
```

This command initiates a default quick-start workflow using a small synthetic dataset. The workflow encompasses training for three honeyvault models, attack model training, online attack evaluation, and result plotting. Expect this entire process to take approximately one hour.

Upon successful execution, the terminal output should display messages similar to `The merged PDF is saved to x/xx/xxx.pdf`. For verification, manually inspect one of the generated PDF plots to confirm its general resemblance to Figure 6 in the paper. Note that the exact curve shapes may vary.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): For traditional evaluation of honeyvault schemes against distinguishing attacks, our DeBERTa-based attack achieves the highest accuracy among both existing and our proposed attacks. This is demonstrated by Experiment (E2) (Section 6.2) and depicted in Figure 5 and Table 3.

(C2): Our scheme exhibits superior resistance against distinguishing attacks. This is demonstrated by Experiment (E2) (Section 6.2) and depicted in Figure 5 and Table 3.

(C3): For practical evaluation of honeyvault schemes against online guessing attacks, our DeBERTaBase attack achieves the best performance. This is demonstrated by Experiment (E1) (Section 6.3) and depicted in Figures 6-14 and Table 7.

(C4): For practical evaluation of honeyvault schemes against online guessing attacks, our scheme achieves the best security. This is demonstrated by Experiment (E1) (Section 6.3) and depicted in Figures 6-14 and Table 7.

(C5): Our two simple measures significantly enhance the security of all three honeyvault schemes against online guessing. This is demonstrated by Experiment (E1) (Section 6.3) and depicted in Table 5 and Table 7.

A.4.2 Experiments

It is strongly recommended to execute the experiments sequentially as listed below, as the initial experiment (E1) automates the training of honeyvault and attack models.

(E1): *[Online Guessing] [5 human-minutes + 3 compute-weeks]:*

Preparation: Adjust the configuration to match the paper's experimental setting. In `src/honeyvault/CONFIG_Online_quick_start.py`, comment out the `QuickStart` block and uncomment the `Paper` block. For deep learning-based attacks, only the optimal one (DeBERTaBase) is executed by default; alternatively, modify the `DL_model` variable in `src/honeyvault/__main__.py:57` to include other deep learning-based attacks.

Execution: Execute the following command:

```
python src/honeyvault/__main__.py
```

Results: Results will be saved in the `src/honeyvault/attack/leaky_detection/leak_pic` directory. This directory will contain multiple subdirectories, each named to indicate different experimental conditions:

- A name containing `honey_num_0` indicates Measure II is disabled, while `honey_num_2` indicates it is enabled.
- A name including `LIMIT_WEB=False` signifies that Measure I is not applied, whereas `LIMIT_WEB=True` signifies its application.
- A name including `ENHANCE_MPW=False` implies that a strong master password is not enforced, while `ENHANCE_MPW=True` implies its enforcement.
- A name containing `defender_attack` indicates the ideal case, while `defender_attack_practical` indicates the practical case.

Within the deepest subdirectory, attack curves are saved in three PDF files. Files named with `2016`, `2021`, and `transformer` correspond to the schemes by Golla et al., Cheng et al., and our proposed scheme, respectively. The precise average number of cracked accounts under 10, 100, 1000 login attempts are saved in JSON files named `max_y_with_pos_2016.json`, `max_y_with_pos_2021.json`, and `max_y_with_pos_transformer.json` for each scheme. The correspondence between figures and tables in the paper and their respective result files is detailed in Table 1.

(E2): *[Distinguishing Attacks] [5 human-minutes + 2 compute-hours]:*

Preparation: Adjust the configuration to match the paper's experimental setting. In `src/honeyvault/CONFIG_RCDF.py`, comment out the `Quick_start` block and uncomment the `Paper Config` block (around line 25). Similar to E1, only the optimal deep learning-based attack (DeBERTaBase) is run by default; modify the `DL_model` variable in `src/honeyvault/attack/__main__.py:319` to include other deep learning-based attacks if desired.

Execution: Execute the following commands:

```
cd src/honeyvault
python -m attack
python draw_picture_final_v2.py
```

Results: Results will be saved in the `src/honeyvault/result/final` directory. RCDFs are saved as three PDF files. Files named with `2016`, `2021`, and `transformer` correspond to the schemes by Golla et al., Cheng et al., and our proposed scheme, respectively. The accuracy α or AUC of distinguishing attacks against honey vault schemes are saved in `700w_average_ranks.csv`. The correspondence between figures and tables in the paper and their respective result files is also detailed in Table 1.

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/usenixsec2025/.

Table 1: Summary of the major claims and experiments.

Claim	**Exp.**	**Figure or table in the paper**	**Corresponding result files**
C3, C4, C5	E1	Figure 6	`$PATH_E1/defender_attack/honey_num_0_attack_op/ENHANCE_MPW=False_LIMIT_WEB=False/3_attack_curves_p0_*.pdf`
		Table 4 and Table 5: None	`$PATH_E1/defender_attack/honey_num_0_attack_op/ENHANCE_MPW=False_LIMIT_WEB=False/max_y_with_pos_*.json`
		Figure 7	`$PATH_E1/defender_attack_practical/honey_num_2_attack_op/ENHANCE_MPW=False_LIMIT_WEB=True/3_attack_curves_p0_*.pdf`
		Tables 5 and 7: M1 & M2 (Practical Case)	`$PATH_E1/defender_attack_practical/honey_num_2_attack_op/ENHANCE_MPW=False_LIMIT_WEB=True/max_y_with_pos_*.json`
		Figure 8	`$PATH_E1/defender_attack/honey_num_0_attack_op/ENHANCE_MPW=False_LIMIT_WEB=True/3_attack_curves_p0_*.pdf`
		Tables 5 and 7: M1	`$PATH_E1/defender_attack/honey_num_0_attack_op/ENHANCE_MPW=False_LIMIT_WEB=True/max_y_with_pos_*.json`
		Figure 9	`$PATH_E1/defender_attack/honey_num_2_attack_op/ENHANCE_MPW=False_LIMIT_WEB=False/3_attack_curves_p0_*.pdf`
		Tables 5 and 7: M2 (Ideal case)	`$PATH_E1/defender_attack/honey_num_2_attack_op/ENHANCE_MPW=False_LIMIT_WEB=False/max_y_with_pos_*.json`
		Figure 10	`$PATH_E1/defender_attack_practical/honey_num_2_attack_op/ENHANCE_MPW=False_LIMIT_WEB=False/3_attack_curves_p0_*.pdf`
		Tables 5 and 7: M2 (Practical case)	`$PATH_E1/defender_attack_practical/honey_num_2_attack_op/ENHANCE_MPW=False_LIMIT_WEB=False/max_y_with_pos_*.json`
		Figure 11	`$PATH_E1/defender_attack/honey_num_0_attack_op/ENHANCE_MPW=True_LIMIT_WEB=False/3_attack_curves_p0_*.pdf`
		Tables 5 and 7: MPW	`$PATH_E1/defender_attack/honey_num_0_attack_op/ENHANCE_MPW=True_LIMIT_WEB=False/max_y_with_pos_*.json`
		Figure 12	`$PATH_E1/defender_attack/honey_num_2_attack_op/ENHANCE_MPW=False_LIMIT_WEB=True/3_attack_curves_p0_*.pdf`
		Tables 5 and 7: M1 & M2 (Ideal Case)	`$PATH_E1/defender_attack/honey_num_2_attack_op/ENHANCE_MPW=False_LIMIT_WEB=True/max_y_with_pos_*.json`
		Figure 13	`$PATH_E1/defender_attack/honey_num_2_attack_op/ENHANCE_MPW=True_LIMIT_WEB=True/3_attack_curves_p0_*.pdf`
		Tables 5 and 7: M1 & M2 (Ideal Case) & MPW	`$PATH_E1/defender_attack/honey_num_2_attack_op/ENHANCE_MPW=True_LIMIT_WEB=True/max_y_with_pos_*.json`
		Figure 14	`$PATH_E1/defender_attack_practical/honey_num_2_attack_op/ENHANCE_MPW=True_LIMIT_WEB=True/3_attack_curves_p0_*.pdf`
		Tables 5 and 7: M1 & M2 (Practical Case) & MPW	`$PATH_E1/defender_attack_practical/honey_num_2_attack_op/ENHANCE_MPW=True_LIMIT_WEB=True/max_y_with_pos_*.json`
C1, C2	E2	Figure 5	`src/honeyvault/result/final/700w_*_attack_result.pdf`
		Table 3	`src/honeyvault/result/final/700w_average_ranks.csv`

[*] The variable `$PATH_E1` expands to the path `src/honeyvault/attack/leaky_detection/leak_pic`.