A Artifact Appendix

A.1 Abstract

We provide code, data, and outputs of our experiments. Our artifact is publicly available at https://github.com/Yuanyuan-Yuan/Manifold-SCA with detailed documents. Using our tool, users can perform side channel attacks on media software and localize side channel vulnerabilities of the target software. We also provide a mitigation scheme towards our attack and investigate the noise resilience of our attacking technique.

A.2 Artifact check-list (meta-information)

- **Data set.** See README in our artifact.
- **Run-time environment.** Our experiments are launched on 64-bit Ubuntu 18.04, we recommend users to set up on the same OS. We also provide a docker container with everything set up. Performing `Prime+Probe` attack needs root access.
- **Hardware.** We perform `Prime+Probe` attacks on Intel Xeon and AMD Ryzen CPUs. Nevertheless, our approach is *not* hardware-specific. Users can use our tools on other CPUs. To approximate manifold from known data (i.e., the training split), users are recommended to run scripts on GPUs. Note that our tool requires a relatively large RAM.
- **Execution.** Our experiments are launched on one Nvidia GeForce RTX 2080 GPU. The running time of approximating manifold (i.e., training models) is less than 24 hours. Nevertheless, it will be very slow if the script is executed with only CPUs. We have released our trained models. The data processing and side channel logging are also time-consuming, which may take several days. We also provide our processed data and logged side channels.
- **Security, privacy, and ethical concerns.** Our tool is provided as-is and is only for research purposes. Please use it only on test systems with no sensitive data. Users are responsible for protecting themselves, their data, and others from potential risks caused by our tool.
- **Output.** Our outputs include 1) logged side channel records; 2) trained models which appropriate data manifold; 3) reconstructed media data from unknown side channel; 4) localized side channel vulnerabilities of media software.

  We release 1) scripts for logging side channels and our logged side channel records; 2) scripts for training models and our trained models; 3) scripts for reconstructing media data from unknown side channels (i.e., the test split) and our reconstructed media data; 4) scripts for localizing side channel vulnerabilities and our localized vulnerabilities. Some vulnerabilities have been explored by previous works, and the new-found vulnerabilities have been confirmed by developers of `FFmpeg` and `libjpeg` by the time of writing. See outputs for more details.
- **How much disk space required (approximately)?** We provide 1K samples of processed data and side channel records for each dataset and software. We also provide our trained models and a docker container. To launch experiments using these data samples, which are sufficient to verify our findings, users need to prepare **at least 20G** space. Further, if users want to prepare all data (we also provide the scripts), **2T** space is desired.
- **Experiments.** We provide 1K samples of processed data and side channel records for each dataset and software. These samples are sufficient to verify our statements and results, for instance, reconstructing high-quality media data from side channel records and mitigating side channel attack using perception blinding (e.g., perceptual properties of reconstructed images are dominated by the mask).

  Since experiments are performed on a limited number of data, some numerical results may have relatively large variances. Also, it's worth noting that the 1K samples are *not* enough to train the model (i.e., the trained model has a poor capability of reconstructing media from unknown side channels), but users can see that the reconstructed media data from known side channels gradually have higher quality and get similar to the reference media data. Users can use our provided scripts to produce all data involved in our paper.
- **How much time is needed to prepare workflow/complete experiments (approximately)?**

  1) Set up the environment: less than 1 hour. We also provide a docker container with everything set up.

  2) Download public datasets and process: it requires less than 1 hour to process the data. We provide our processed data samples.

  3) Log side channels: around one week to log side channels of all media and target software. We provide our logged side channels.

  4) Train models: training one model requires less than 24 hours on one Nvidia GeForce RTX 2080 GPU. Our script also supports training on CPUs, but it could be time-consuming. We also release our trained models.

  5) Others: a few minutes.
- **Publicly available?** Our artifact is publicly available at https://github.com/Yuanyuan-Yuan/Manifold-SCA.
- **Code licenses (if publicly available)?** MIT license.
- **Data licenses (if publicly available)?** CC-BY-4.0 license.
- **Workflow frameworks used?** We use Pytorch as the building block of our framework.
- **Archived (provide DOI or stable reference)?** Available at https://zenodo.org/record/5816702#.YdQMHxNByjA.

A.3 Description

See all details in our README.

A.3.1 How to access

Access our artifact at https://github.com/Yuanyuan-Yuan/Manifold-SCA.

A.4 Installation

See README. We also provide a docker container with everything set up.

A.5 Evaluation and expected results

We show that side channel analysis (SCA) towards media software can be largely boosted by manifold learning, which recasts SCA as mapping between side channels and media data via a low-dimensional joint manifold. Enabled by the neural attention mechanism, we can localize side channel vulnerabilities of media software by investigating which records on a logged side channel trace contribute most to the reconstruction of media data. Our findings have been confirmed by the software developers. We further propose the perception blinding that is highly effective for mitigating manifold learning-based side channel attacks. We also show that our approach is highly robust to noise in collected side channels.

Side Channel Attack. By using our released tools, users can log side channel records of the target software when it is processing private data. Based on the collected side channels and corresponding media data, users can train a model to appropriate the manifold. The trained model can reconstruct high-quality media data from unknown side channels (i.e., the test split of each dataset). We provide our trained models and 1K data samples (from test split). Using our trained models, users can observe that the reconstructed media data manifest consistent perceptual properties with the reference media data. Note that some numerical results (e.g., the text inference accuracy) may have large variances since they are calculated on only a few samples. Also, the provided data samples are *not* enough for training models, but users can still observe that the manifold (despite its poor generalization capability) is gradually formed when training models on these samples. To prepare all data records, which require roughly 2T space, users can download the public datasets and process them using our scripts. It's worth noting that due to the non-deterministic operations of Pytorch, training results and some inference results may be slightly different each time, but findings and conclusions derived from these results are consistent. Moreover, the results always largely outperform the baseline.

Localizing Side Channel Vulnerabilities. Users can use our scripts to localize side channel vulnerabilities once the manifold is formed. For instance, to investigate records produced by which functions in `libjpeg` contribute most to reconstructing images, users are expected to observe that `idct` and `mcu` related functions have the highest frequency. We also provide our localized vulnerabilities. Some vulnerabilities have been exported by previous works, and the new-found ones have been confirmed by developers. Note that the produced results on the 1K samples may be slightly different from our provided results. That is reasonable since the frequency of each localized function could have a relatively large variance on only a few examples.

Perception Blinding. We provide scripts for users to perform perception blinding on media data. We also provide blinded images and corresponding side channel records. Users can observe that given the side channels of blinded images, our framework can hardly reconstruct privacy—the perceptual properties are dominated by the blinding masks. Users can also use our scripts to produce other blinded data and use their customized blinding masks.

Noise Resilience. We show that our technique is robust towards noise in collected side channels. By using our provided scripts, users can introduce noise of various types and weights into side channels. The reconstructed media data from noisy side channels are still of high quality and manifest most of the perceptual properties of reference media data.

A.6 Experiment customization

Our artifact supports customized settings. More specifically, users can customize 1) the media datasets, 2) hardware platforms, 3) target software, 4) model architectures, 5) training parameters when appropriating manifold, 6) blinding masks, 7) noise insertion schemes. We provide APIs for customized settings; see details in README.