ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

# USENIX Security '24 Artifact Appendix: Invisibility Cloak: Proactive Defense Against Visual Game Cheating

Chenxin Sun[*], Kai Ye[*], Liangcai Su, Jiayi Zhang, Chenxiong Qian[†]
The University of Hong Kong

## A Artifact Appendix

## A.1 Abstract

The increasing prevalence of visual aimbots in first-person shooter games poses a significant threat to fair play and gaming experience. Visual aimbots utilize game visuals and integrated visual models to extract game information, providing cheaters with automatic shooting capabilities. To counter this, we present *Invisibility Cloak*, the first proactive defense framework against visual game cheating. Our approach introduces imperceptible perturbations to game visuals, rendering them unrecognizable to AI models. This artifact highlights a subset of the experimental data and samples described in the paper. While the same operations can be applied to CF, this artifact only showcases the demos related to CS2. The results indicate that our approach maintains a high defense success rate while ensuring efficiency and smooth gameplay.

## A.2 Description & Requirements

### A.2.1 Security, privacy, and ethical concerns

There is no risk associated with downloading or modeling any of the data or models created through this article. The code does not involve any destructive steps or the disabling of security mechanisms.

### A.2.2 How to access

The artifact can be accessed from the following GitHub repository: https://github.com/GamesecInvisicloak/Invisibility-Cloak/tree/a54a6fa3bf16f617ac7c367201b5978ce91c2265. The repository includes all the necessary code, datasets, and instructions to reproduce the experiments.

### A.2.3 Hardware dependencies

This artifact requires a system with an NVIDIA GPU for running the deep learning models efficiently. The specific hardware used in our experiments includes an Intel(R) Xeon(R) Gold 5418Y CPU with 96 cores and an NVIDIA GeForce RTX 4090 GPU with 24.564 GB of memory. If you use our provided Docker environment, please ensure your GPU model supports CUDA 12.5.0. If you are configuring the environment yourself, any GPU model will suffice.

[*]Equal Contribution.
[†]Corresponding author.

### A.2.4 Software dependencies

The artifact requires Python 3.8 or higher. The key software packages needed are CUDA, Conda (recommended) or a Python virtual environment, PyTorch, etc. All other dependencies can be installed using the provided `requirements.txt` file in the repository. We used YOLO series models ('yolov5n', 'yolov5s', 'yolov5m') in our experiments, but the models are provided within the artifact and do not require additional downloads. You can download CUDA using the following link: https://developer.nvidia.com/cuda-toolkit; You can download Conda using the following link: https://conda.io/projects/conda/en/latest/user-guide/install/index.html. Other dependencies can be downloaded using `requirements.txt` file.

### A.2.5 Benchmarks

None.

## A.3 Set-up

### A.3.1 Installation

We provide a pre-configured Docker image with all the required environments installed, making it easy to run the scripts directly. You can download the image using the following link: https://drive.google.com/file/d/1FcUy_LG8ySqxaLJAb1_W_PWFeucLT1gA/view?usp=sharing. First, download the Docker image file and load it, run:

```
↪ docker load -i invisicloak_configured.tar
```

Then, run the Docker container with GPU support:

```
↪ docker run -it --rm
  --gpus all invisicloak:configured
```

The Docker container already includes CUDA 12.5.0 and the `invisicloak` Conda environment pre-installed. To activate the Conda environment, run:

```
↪ conda activate invisicloak
```

If you prefer not to use Docker, you can manually set it up with Conda. First, create and activate an environment:

```
↪ conda create -n invisicloak python=3.8
↪ conda activate invisicloak
```

Next, install the required packages:

```
↪ pip install -r requirements.txt
```

### A.3.2 Basic Test

After successfully setting up the experimental environment, we use data from different game scenarios in CS2 to test the effectiveness of Invisibility Cloak. The purpose of our basic test is to evaluate whether the Cloak we generate can successfully defend against visual aimbots' detection when applied to the game screen samples. This test will utilize YOLO series models as the local proxy model and the target cheating model to assess Cloak's performance and effectiveness.

To generate cloaked samples for a chosen demo scenario (e.g., `stand`) in the CS2 game, use the following command:

```
↪ python get_cloak.py --scenario stand
```

The script will generate Cloaks and add them to the input scenario samples, creating protected images. We can use various command line arguments to customize the execution, such as specifying the number of iterations, learning rate, attack strength, models to use, and GPU device. For example, here we recommend visualizing GIFs of cloaked samples using the following command to simulate the actual gameplay screen:

```
↪ python get_cloak.py --visualize_gif 1
```

The available options are summarized below:

- `-n_iter`: Number of iterations.
- `-lr`: Learning rate for the optimizer.
- `-epsilon`: The $\ell_\infty$-norm constraint ($\varepsilon$).
- `-local_model`: The local proxy model to use.
- `-target_model`: The target cheating model to defend.
- `-gpu`: GPU device to use for computation.
- `-use_universal_cloak`: Whether to use Universal Cloak.
- `-visualize_gif`: Whether to generate a GIF.
- `-scenario`: CS2 demo scenario to use.

Please refer to the repository's `README` file for detailed information on each argument and additional usage options.

Listing 1: The expected successful output of generating cloaked samples and GIFs for a chosen demo scenario.

```

> python get_cloak.py --scenario stand --visualize_gif 1
We are using the Universal Cloak.
dust2_stand 451
Fusing layers...
YOLOv5n summary: 213 layers, 1867405 parameters, 0
      gradients, 4.5 GFLOPs
Adding AutoShape...
YOLOv5 🚀 2024-6-16 Python-3.8.19 torch-2.3.1+cu121 CUDA:0
       (NVIDIA GeForce RTX 4090, 24118MiB)

Fusing layers...
YOLOv5n summary: 213 layers, 1867405 parameters, 0
      gradients, 4.5 GFLOPs
Adding AutoShape..
2024-06-17 22:48:58.312 | INFO     | __main__:main:243 -
      Attack Begin
2024-06-17 22:48:58.312 | INFO     | __main__:main:245 -
      universal_cloak/cs2/yolov5n_yolov5s_yolov5m/Best-
      Succ-0.84-BS-16-LR-0.001.pt
2024-06-17 22:48:58.312 | INFO     | __main__:main:249 -
      len dataset 451
2024-06-17 22:48:58.656 | INFO     | __main__:main:283 -
      Index:0 LSucc:1 TSucc:1 Single_Time:0.137 Query:1
      SSIM: 0.722
2024-06-17 22:48:58.657 | INFO     | __main__:main:285 -
      Total:451 DSR:1.000 AvgTime:0.137 FPS:7.310 AvgSSIM:
       0.722
Saved 1 image to result/visualization/cs2_demo/stand/
      attack/0
Saved 1 image to result/visualization/cs2_demo/stand/gt/0
                              ...
                              ...
2024-06-17 22:49:31.796 | INFO     | __main__:main:283 -
      Index:450 LSucc:1 TSucc:1 Single_Time:0.005 Query:1
      SSIM: 0.761
2024-06-17 22:49:31.797 | INFO     | __main__:main:285 -
      Total:451 DSR:1.000 AvgTime:0.004 FPS:235.844
      AvgSSIM: 0.745
Saved 1 image to result/visualization/cs2_demo/stand/
      attack/450
Saved 1 image to result/visualization/cs2_demo/stand/gt
      /450
2024-06-17 22:50:13.578 | INFO     | __main__:main:299 -
      Creating Gif every 100 frames
```

After inputting the chosen scenario and other options (e.g., generating GIFs), the expected output is shown in Listing 1. These outputs display various evaluation metrics that can be viewed directly in the terminal. Additionally, the logs and the generated cloaked samples are saved in the `result` directory.

## A.4 Notes on Reusability

The Invisibility Cloak framework can be adapted for other first-person shooter games beyond CF and CS2 by collecting appropriate datasets and fine-tuning the perturbation models. The code is modular and can be extended to include additional visual models for enhanced defense capabilities.

## 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/usenixsec2024/.