Invisibility Cloak: Proactive Defense Against Visual Game Cheating

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

https://www.usenix.org/conference/usenixsecurity24/presentation/sun-chenxin

This paper is included in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Open access to the Proceedings of the 33rd USENIX Security Symposium is sponsored by USENIX.

Invisibility Cloak: Proactive Defense Against Visual Game Cheating

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

Abstract

The gaming industry has experienced remarkable innovation and rapid growth in recent years. However, this progress has been accompanied by a concerning increase in First-person Shooter game cheating, with aimbots being the most prevalent and harmful tool. Visual aimbots, in particular, utilize game visuals and integrated visual models to extract game information, providing cheaters with automatic shooting abilities. Unfortunately, existing anti-cheating methods have proven ineffective against visual aimbots. To combat visual aimbots, we introduce the first proactive defense framework against visual game cheating, called *Invisibility Cloak*. Our approach adds imperceptible perturbations to game visuals, making them unrecognizable to AI models. We conducted extensive experiments on popular games CrossFire (CF) and Counter-Strike 2 (CS2), and our results demonstrate that *Invisibility Cloak* achieves real-time re-rendering of high-quality game visuals while effectively impeding various mainstream visual cheating models. By deploying *Invisibility Cloak*[1] online in both CF and CS2, we successfully eliminated almost all aiming and shooting behaviors associated with aimbots, significantly enhancing the gaming experience for legitimate players.

1 Introduction

In recent years, there have been major advancements in gaming technology and user engagement, leading to rapid industry growth. Market projections indicate that the revenue of video game markets is expected to reach $282.3 billion in 2024, with a year-on-year growth of 8.76% until 2027 [47]. However, this progress has been accompanied by a concerning rise in in-game cheating, which poses a serious threat to game integrity and player experience. This threat is particularly prominent in First-person Shooter games, renowned for their exceptional competitiveness and entertainment value, resulting in losses exceeding $23 billion [1].

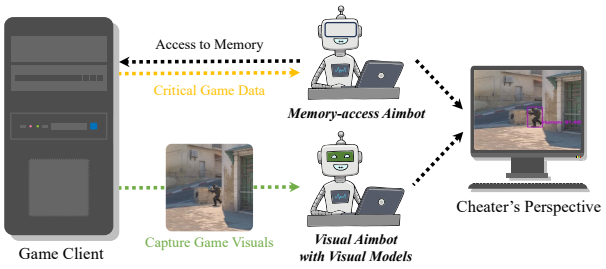


Figure 1: Workflows of Memory-access and Visual aimbots.

In game cheating, the most pervasive and harmful cheating, is the *aimbot*. These tools artificially enhance a player's aiming and shooting accuracy, fundamentally disrupting the integrity and fairness of gameplay. The aimbot process mainly involves two steps: acquiring game data and automatic shooting. Depending on the data acquisition method, aimbots can be broadly classified into two types: *Memory-access aimbots* and *Visual aimbots*. Memory-access aimbots directly access game data in the client's memory, typically including precise target objects and resource positions. To prevent such cheating, researchers have developed anti-aimbot tools [28, 29, 39, 59–61] to block unauthorized access to game memory. For instance, the most recent system, BotScreen [11], leverages Intel SGX [13] to protect game data from accessing and tampering, rendering the memory-access aimbots non-functional. Unlike memory-access aimbots, visual aimbot relies solely on the game's visual screen, utilizing integrated visual models(e.g., YOLO [50], NanoDet-Plus [41]) to identify game objects. As shown in Figure 1, it first captures the normal game screen and then extracts information (e.g., coordinates, distance, and size of targets) based on the visual model. This cheating is also known as Visual Game Cheating.

Visual game cheating has become increasingly powerful and prevalent with the development of AI and cloud gaming. Currently, various visual aimbots [19, 35, 44] and tutorials [9, 40] are easily accessible and popular. The harmfulness of this trend is primarily demonstrated in two aspects: (a) **High cheating success rate**. With AI continuously mak-

*Equal Contribution.
†Corresponding author.
[1]Our project can be found at: https://invisicloak.github.io/

ing breakthroughs in the field of computer vision, fast and accurate object detection [5, 18, 41, 50] and skeleton recognition models [15, 49, 58] are being integrated into aimbots. Moreover, RootKit [45] has announced plans for international competitions [43] to further enhance the development of their AI-Aimbots [44]. (b) **Wide applicability and stealthiness**. The only data visual aimbots utilize are legitimate and accessible game footage, making them highly adaptable and undetectable to various games and deployable across different platforms, including mobiles and computers. In particular, cloud gaming is emerging as the next-generation gaming architecture due to its low latency, scalability, and conveniences [42, 46]. The cloud server and the player's client are entirely separated throughout the cloud gaming process, with the client only receiving video streams from the cloud server. While cloud gaming renders traditional cheats unfeasible, it inevitably positions visual aimbots as the mainstream choice for next-generation cloud game cheats.

Disappointingly, when it comes to visual aimbots, passive detection-based approaches are currently the only viable anti-cheating techniques [2, 17, 23, 25, 29, 34, 63]. However, these techniques identify cheaters by analyzing abnormal game data after the fact, failing to guarantee a smooth gaming experience for normal players. Moreover, it suffers from a certain degree of misjudgment and lack of explainability. Rather than relying on passive cheating detection, is it possible to achieve proactive defense against visual game cheating?

To this end, we propose a proactive defense framework for visual game cheating, namely *Invisibility Cloak*. To the best of our knowledge, this is the first work to proactively perform anti-cheating in visual games. The core idea is to introduce an invisible perturbation to the game screen. This perturbation is undetectable to human players but prevents visual models from accurately recognizing the game screen targets. Considering the method's usability, practicality, and deployability, proactive defense against visual game cheating needs to overcome the following challenges. **(C1) Transferability.** The approach should be transferable because it is impossible to know the cheating models used by cheaters in advance. **(C2) Robustness.** The approach should be applicable to various games and complex in-game scenarios. In addition, to ensure a normal experience for players, the method also needs to meet the requirements, including **(C3) Imperceptibility** of the perturbation and **(C4) Real-Time Performance**.

To address the aforementioned challenges within the framework of the Invisibility Cloak, we propose a *Cloak Crafter* module. This module comprises two components: offline Universal Cloak Identification and online Cloak Refinement. In the Universal Cloak Identification, we first adopt multiple visual models and diverse game screenshots to search for a Universal Cloak, where the cloak refers to an imperceptible perturbation. Further, in the Cloak Refinement, we refine the universal perturbation in the game environment to make it adapt to specific game scenarios. To meet C3 and C4, we require the perturbation to be sufficiently small throughout the entire process and limit the number of queries. To evaluate the effectiveness of our defense, we conduct comprehensive experiments on two of the most popular First-person Shooter games, *CrossFire* (CF [2]) and *Counter-Strike 2* (CS2 [3]). The results demonstrate that the Invisibility Cloak can effectively resist visual game cheating without affecting players' gaming experience. We also performed a simplified online deployment and found that the visual aimbot was rendered ineffective in both CF and CS2 when the Invisibility Cloak was enabled. Our main contributions are summarized as follows:

- ***Proactive Anti-Cheating:*** We introduce a novel proactive anti-cheat method specifically designed to combat visual aimbots that are difficult for traditional anti-aimbot techniques to detect. Our approach represents a more dynamic and preemptive form of cheat prevention.
- ***Non-Invasive Implementation:*** Our approach seamlessly integrates with the game rendering process, overlaying the Cloaks onto the player's display without user monitoring.
- ***Comprehensive Dataset Collection:*** We collected extensive image datasets that contain different game maps and scenarios from the most popular games, CS2 and CF, serving as the foundation for training and evaluating *Invisibility Cloak*.
- ***Invisible Protection:*** Experiments conducted demonstrate that our approach effectively meets the demands of First-person Shooter games, safeguarding players against visual aimbots without compromising visual quality or real-time performance, resulting in an unobtrusive gaming experience that can be considered virtually invisible.

2 Background

In this section, we first present the workflow of visual aimbots, benefiting from state-of-the-art object detection models. Then, we examine the limitations of existing adversarial attacks targeting object detection models, particularly when implementing these attacks within First-person Shooter games.

2.1 Aimbot Cheating Using Visual Models

As Computer Vision (CV) technology advances, visual aimbots are gaining popularity, especially in First-person Shooter games featuring human character models such as CF and CS2. Developing an aimbot using these visual models is relatively simple and efficient. Numerous publicly available pre-trained models have high accuracy and typically incorporate *person* as part of their training datasets [7, 52]. This accessibility enables cheaters to utilize these pre-trained models in their aimbots without the need for retraining, thereby streamlining the creation of highly effective cheating tools.

[2] https://cf.qq.com/
[3] https://www.counter-strike.net/

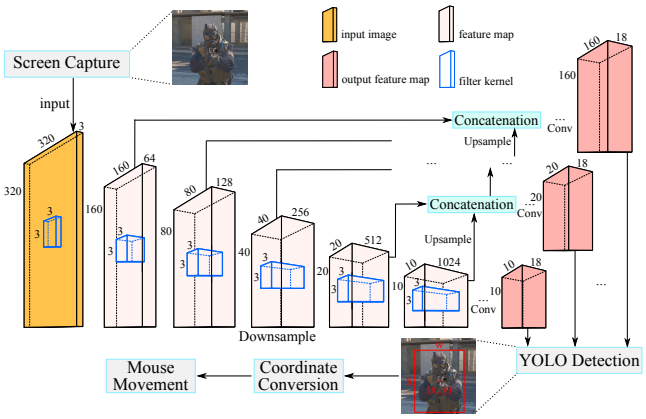


Figure 2: The workflow of aimbot cheating using YOLO.

Figure 2 shows the workflow of a typical visual aimbot. Cheaters first initiate the aimbot process by capturing the game screen on their devices. Normally, the screen resolution is huge, but cheaters usually resize the captured image to a small predetermined resolution (e.g., 320×320 pixels) to facilitate subsequent input into object detection models and only focus on the game screen around the crosshair, thus reducing model overhead and preventing the detection of too many enemies. The resized image is then input into a trained visual detection model, such as the widely used YOLO. As illustrated in Figure 2, YOLO employs a convolution neural network (CNN) [36] for the feature extraction phase. Within the CNN, the image undergoes a series of convolutional and pooling layers. It generates multiple feature maps at different levels during the downsampling, each capturing distinct scales and aspects of the image, which is critical for identifying objects of varying sizes. Subsequently, the output feature maps are synthesized through an upsampling process and concatenated with prior layer feature maps, culminating in a comprehensive detection output. Once detection is achieved, YOLO provides the coordinates of the detected objects in the format (x, y, w, h), where (x, y) are the center coordinates, and w and h are the width and height of the bounding box, respectively. Cheaters exploit these coordinates to locate enemy characters, particularly targeting the head for more effective eliminations. The offset of this target from the center of the screen is then used to automatically adjust the aimbot's cursor, typically via Windows APIs (e.g., win32api and win32con) to enable precise and automated targeting.

2.2 Adversarial Attacks in Object Detection

Object detection models based on DNN are already very powerful, but it has been found that they can still be rendered ineffective by adversarial attacks. Adversarial attacks in object detection models typically involve carefully designed perturbations that are too subtle for the human eye to recognize but are effective in deceiving the visual model. These perturbations can be injected into the input images, causing the model to incorrectly identify non-existent objects, misclassify objects, or fail to detect objects altogether. The essence of these attacks lies in their ability to exploit the model's reliance on high-dimensional data and its linearity, thus revealing a critical vulnerability in object detection systems.

Some methods affect object detectors by adding noise to the entire image [12, 56], while others attack object detectors by adding patches into the image [30, 55, 64]. However, these methods either suffer from easily noticeable perturbations (patching), or the attack speed cannot keep up with the object detector's inference speed. Traditional object detection attack methods are no longer suitable for combating visual game cheating because it is impossible to predict which object detector will be used in advance. Hence, gradient-based white-box attack methods are difficult to transfer sufficiently. Although query-based black-box attacks typically exhibit better transferability, the substantial number of queries required for black-box attacks is another challenge for real-time attacks. Even assuming a 30 frames per second (FPS) game and a lightweight object detector like YOLOv5n with a 7ms inference time (for CPU) [22], the maximum acceptable query count is about 5. Generative attack methods can meet the real-time requirements [21, 54], but training a model that balances transferability and attack success rate is challenging (instead of lowering the mean average precision (mAP), the goal is to make the detection box of a specific target disappear). Therefore, to defend against visual aimbots, it is important to disable the target detector quickly while maintaining a high attack success rate without being noticed.

3 Overview

3.1 Threat Model

Scenario. In this paper, we discuss how to defend against visual game cheating, a scenario rarely discussed in existing anti-cheating technology. A visual aimbot captures the rectangular area around the mouse cursor in real-time from the game screen. The detection resolution is typically small (e.g., 320×320 pixels [44]), that is to maximize accuracy and prevent the mouse from erratic movements when multiple targets are detected simultaneously. The aimbot uses a cheating visual model to obtain the position boundary box of the character from the screenshot. After detection, the aimbot will automatically move the mouse to the identified target box and start shooting. This cheating method is non-invasive and relies only on game screens, so it can theoretically be applied to various game scenarios and devices. Especially in cloud games that only transmit video streams to players, visual game cheating is inherently suitable, while traditional cheating methods are completely unsuitable because they cannot obtain effective game data from memory or game processes.

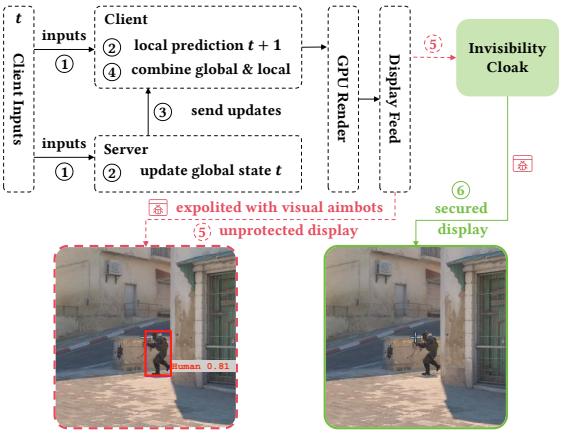


Figure 3: The functional stage of Invisibility Cloak under modern MMOGs architecture.

3.2 Invisibility Cloak

Figure 3 depicts the core conceptual framework of the Invisibility Cloak. Steps ① to ④ represent the standard architecture of modern Massively Multiplayer Online Games (MMOGs). After completing the rendering process, the visual output can be easily affected without protection, as shown in the step ⑤. Such unprotected displays are easily exploited by cheating software that uses visual information for unfair advantage. However, the pivotal innovation introduced by our approach is the inclusion of Invisibility Cloak after rendering. The Invisibility Cloak is designed to disrupt the effectiveness of detection models in visual game cheating. It disables visual aimbots by adding imperceptible perturbations to the rendered game screen before the aimbot performs object detection. As shown in step ⑥, the cheating methods cannot detect any targets on the screen. At the same time, a notable aspect of Invisibility Cloak is that the Cloak added is sufficiently subtle to remain undetected by players, so it does not affect the player's visual experience. In summary, Invisibility Cloak can provide strong anti-cheat protection while remaining *invisible*.

The Cloak Crafter is the core component of Invisibility Cloak system. It is specifically designed to receive incoming game frames that require cloaking and rapidly output cloaked frames. This invisibility makes it undetectable to both the game players and the aimbots. From the player's perspective, the gaming experience remains unchanged, as no discernible alterations are perceived. Conversely, for the aimbots, this cloaking disrupts their core functioning, which is the ability to accurately recognize opponents on the screen, so they can not move the mouse to complete the elimination.

3.3 Challenges for Proactive Defense

The swift refresh rates of modern First-person Shooter games [51] necessitate crafting the target frame as quickly as possible to complete the cloaking process, thus preventing the aimbot from assisting cheaters in completing elimination before becoming invisible. It is important to emphasize that, unlike video processing, our approach relies solely on current and past frames, because future frames remain indeterminate and cannot be accurately predicted. Consequently, there is no requirement for preemptive cloaking of future frames. The Cloaks must be crafted before the object is detected by the aimbots, ensuring real-time invisibility and effectively neutralizing the functionality of visual aimbots. In addition to the real-time rendering challenges, our cloaking solution must also demonstrate transferability. Given the wide variety of vision models that cheaters may employ, predicting the specific cheating architectures is impossible. Even within the same family of object detection models, such as the various iterations of YOLO, architectural differences are significant. Therefore, our Invisibility Cloak is designed to be as general as possible across a wide range of cheating models. Furthermore, in-game robustness must be considered, because each game presents a unique array of maps and backgrounds.

In summary, addressing the challenge of visual aimbots within the diverse and unpredictable realm of First-person Shooter games is a task of great complexity and significance. Invisibility Cloak demonstrates its indispensable value in preserving the integrity of competitive gaming environments with its swiftness, versatility, and resilience.

4 Cloak Crafter

Inspired by TOG [12], we designed Cloak Crafter to add imperceptible perturbations to the game screen in real-time, preventing the visual aimbots detection model from correctly detecting targets in the game. Overall, our objective is to devise a defensive perturbation; in this paper, we call it a *Cloak*. When applied to the original game imagery, this Cloak can address the challenges outlined in § 3.3, including real-time performance, imperceptibility, transferability, and robustness.

Problem Definition. Consider a sequential set of game screenshots denoted as $\mathcal{X} = \{x_1, x_2, \cdots, x_T\}$, where x represents a frame. Cheating players utilize an object detection model $\mathcal{M}^C$ as the cheating model, which takes game screenshots x as input and outputs the position, class, and confidence score $\mathcal{S}$ of the game objects. Our proactive defense aims to find a Cloak δ and add it to x so that it reduces $\mathcal{S}$. The process is formulated as follows:

$$\arg\min_{\delta} \mathcal{S} = \mathcal{M}^C(x+\delta), s.t., \|\delta\|_\infty < \varepsilon \quad (1)$$

where ε is the upper limit of the perturbation magnitude. The defense typically succeeds if $\mathcal{S}$ is below the threshold θ.

Cloak Crafter is comprised of two primary components. The first component involves finding a Universal Cloak which is similar to a universal perturbation [33] that can disrupt the ability of various cheating models to detect targets in different scenarios. The second component leverages an efficient

search mechanism based on momentum gradient optimization to refine and optimize this Cloak, ensuring its efficiency and effectiveness. This subsection will delve into the characteristics of Cloak Crafter and discuss how our approach systematically addresses the challenges outlined in § 3.3.

Transferability. Our investigation reveals that visual aimbots use different object detection models. Even within the same framework (such as the YOLO series), there is considerable variation in the architecture of different pre-trained models. Therefore, the training strategy for Cloak Crafter is designed to be model-agnostic, targeting a broad spectrum of visual detection architectures. To circumvent the potential issue of overfitting the Cloak to a single model, we use different proxy models in the training phase to find a Cloak. This strategy enhances the transferability of Cloak Crafter, ensuring its efficacy across a diverse array of object detection models employed in visual aimbots cheating.

Robustness. Beyond its transferability across different visual models, Cloak Crafter's robustness extends to its efficacy in diverse gaming scenarios and player settings. Different games have various maps, backgrounds, and resolutions, necessitating a universal perturbation to accommodate these variations. To achieve this, our training dataset includes a wide array of game scenes, encompassing different maps, special effects (e.g., smoke, Molotov), and scenes with varying numbers of characters and objects (e.g., animals, footballs). Additionally, dataset augmentation through resolution adjustments further ensures our Cloak's effectiveness across various settings. This strategy guarantees the universal applicability of Cloak Crafter, making it robust against changes in in-game environments and player preferences.

Imperceptibility. We use ℓ_∞ to control the perturbation size, which is the most popular and widely used norm [10, 32]. Although the ℓ_∞-norm constraint is more stringent, it makes perturbations less detectable by human eyes and enhances robustness against detection models that are overly sensitive to extreme features, thus increasing applicability in real-world scenarios. We observe that altering the position of the bounding box through perturbations is more challenging than reducing its confidence score. Therefore, our invisibility criterion is based on the confidence score of the predicted box falling below a set threshold rather than using the intersection over union (IoU) between the predicted box and the ground truth. This criterion requires less information from the model and avoids pre-annotating bounding box positions in the dataset.

Real-Time Performance. Given the high frame rate demands of modern First-person Shooter games, the deployment strategy is designed to insert pre-trained, static universal perturbation (Universal Cloak) in all game scenes, resulting in zero latency and a direct defense against visual cheating in most cases. Furthermore, refined by efficient momentum gradient search and strict query constraints, the Cloaks can be generated almost instantaneously. Then, these refined Cloaks are intervals deployed to the game frames. This deployment strategy ensures that Cloak Crafter maintains game flow by providing real-time defense without sacrificing game quality.

In the following subsections, we delve into the specifics of our methodology, first discuss the process of identifying the Universal Cloak (§ 4.1), followed by exploring how we optimize this Cloak across various cheating visual models (§ 4.2).

4.1 Universal Cloak Identification

In this subsection, we introduce how to find a Universal Cloak that can effectively defend against as many cheating models and game scenarios as possible. Next, we will gradually introduce the process of identifying a Universal Cloak. Since this process is completed offline, we temporarily do not consider the limitation of training time in this part.

Step 1: Initialization. We randomly initialize a noise δ_U with the same size as the input x. Each frame of game screenshot x used here is collected from different complex scenarios. Suppose we have K proxy visual models $\{M^1, M^2, \cdots, M^K\}$, which encompass various object detection models under different architectures. Thus, the initial input is $x+\delta_U$.

Step 2: Confidence Score Evaluation. Given the perturbed input $x+\delta_U$, the k-th proxy model M^k will produce a set of bounding boxes with associated confidence scores $S^k \in \mathbb{R}^N$, where N is the number of boxes. For anchor-based methods (e.g., YOLOv5 [50]), the confidence represents the probability that each box contains an object. For anchor-free methods (e.g., NanoDet [41]), it is the probability that each box is classified as the target category, usually *person* in First-person Shooter games. The process is formulated as follows:

$$\mathcal{S}^k = M^k(x+\delta_U) \tag{2}$$

Step 3: Loss Function Construction. Our goal is to minimize the confidence score of the cheating model when perturbing the inputs, ensuring the game target cannot be correctly recognized. Therefore, we use a binary cross-entropy (BCE) loss, setting all box confidence labels to 0, as follows:

$$\mathcal{L} = -\sum_{k=1}^{K}\sum_{i=1}^{N} log(1-s_i^k) \tag{3}$$

where $s_i^k \in \mathcal{S}^k$ is the i-th confidence score of S^k.

Step 4: Universal Perturbation Update. As the proxy models are fully accessible, we can update the universal perturbation using gradient information. The basic form is:

$$\delta_U = \texttt{Clamp}(\delta_U - \alpha \cdot \nabla_{\delta_U}\mathcal{L}) \tag{4}$$

where α represents the learning rate; $\texttt{Clamp}(\cdot)$ is the truncation function that constrains perturbations to have an ℓ_∞-norm below a threshold ε, typically set at 8/255 or 16/255 [62].

Next, we repeat steps 2 through 4 until the defense success rate of Universal Cloak on the validation set reaches a threshold or the loss $\mathcal{L}$ cannot be further reduced. Upon completion, we can obtain a Universal Cloak that can be statically

deployed for real-time protection, with the entire process conducted offline. It is visually unobtrusive to both players and cheaters, offering protection across various game scenarios and a wide range of visual aimbot detection models.

However, it must be acknowledged that due to differences in cheating model frameworks and their respective confidence scoring algorithms, the Universal Cloak is only a generic perturbation with robustness but is not tailored for any specific model or game scenario. Consequently, it may not perform optimally in all situations. Therefore, our next phase involves dynamically updating this Cloak in real-time, adapting to different cheating models and game scenarios encountered during gameplay to strengthen its defense mechanism further.

4.2 Cloak Refinement for Model Variability

In the dynamic environment of First-person Shooter games, where various cheating models and constantly changing game scenarios are encountered, the Universal Cloak identified in the preceding subsection may not always perfectly align with the current data stream. Consequently, we need a process to fine-tune the Universal Cloak for real-game environments.

Due to the online nature of this process, efficiency is the highest priority. To ensure Cloak refinement, we first satisfy the following conditions: (1) The number of query iterations is limited, typically constrained to 5 or fewer; (2) The proxy model must be lightweight and suitable for efficient inference. Thus, we assume a lightweight proxy visual model denoted as M, with the current game frame input denoted as x. The process is delineated into two steps.

Step 1: Universal Cloak Efficacy Assessment. First, we test whether the Universal Cloak δ_U can prevent the current proxy model M from correctly identifying the target object, i.e., whether the maximum confidence score $Max(\mathcal{S})$ is below the threshold θ (usually 0.4). If successful, we can directly use the Universal Cloak, avoiding additional refinement overhead.

Step 2: Conditional Cloak Refinement. If the universal perturbation is not effective, then we initialize the noise δ as δ_U and update the perturbation based on the query. Specifically, we repeat steps 2 to 4 of § 4.1, using only the M model and frame data x. Given the limited number of updates, we adopt a fast optimization scheme inspired by Adam [26], updating the initial learning rate α before each iteration as follows:

$$\alpha = \alpha_i \frac{\mu^t}{\sqrt{s(t)}} \tag{5}$$

where α_i is the initial learning rate, μ^t and $s(t)$ are the intermediate values in the iteration updates of the Adam optimizer.

4.3 Cloak Deployment Strategy

Overall, Cloak Crafter works by first assessing the success of Universal Cloak on the proxy model. Then, for Cloaks that require refinement, we query the proxy model to update them. Our deployment strategy consists of three parts.

Strategy 1: Universal Cloak Deployment. Universal Cloak is initially deployed as the default protective measure by game developers. It is pre-crafted offline and integrated into the game's rendering pipeline, ensuring zero latency during gameplay and providing basic protection for all frames. Note that Universal Cloak can be customized by scenarios (e.g. game maps) as well as regularly updated to ensure that it is not easily accessible to attackers.

Strategy 2: Interval Inserted Refined Cloaks. Given the need for high frame rates in modern First-Person Shooter games, where a comfortable gameplay experience starts at 60 FPS [27], Cloak generation speed often cannot match the game's rendering speed. To maintain smooth gameplay without sacrificing protection, frames are sampled at intervals for Cloak refinement. This means not all game frames will have refined Cloaks, potentially compromising defense on some frames. However, the primary goal is to significantly disrupt cheaters' reliance on visual aimbots. Cheaters use aimbots to gain an unfair advantage, enhancing their experience at the expense of other players. Our strategy ensures that even if some individual frames are not successfully defended, the overall efficacy of aimbot is greatly compromised. The user study (§ 5.8.1) indicates that the high accuracy of cheating tools is crucial, and our defense can substantially reduce aimbots' accuracy, leading cheaters to gradually abandon the aimbots.

Strategy 3: Deployment Environment. Refined Cloaks are generated based on the gaming application. For cloud gaming platforms, where graphics are server-rendered, the refined Cloak is securely generated server-side. For personal computer games relying on client-side rendering, Trusted Execution Environments (TEEs [20]) can be used to prevent Cloak modification or deletion.

5 Evaluation

We answer the following research questions:

- **RQ1.** How effective is our approach in proactively defending against vision-based game cheating methods (§ 5.2)?
- **RQ2.** Is our method transferable to various object detection models that might be adopted by cheaters (§ 5.3)?
- **RQ3.** How robust is Invisibility Cloak in mitigating the impact of different in-game scenarios (§ 5.4)?
- **RQ4.** How does each part of the Invisibility Cloak affect the defense effect (§ 5.5)?
- **RQ5.** How effective is our approach when cheaters try to counter our defense (§ 5.6)?
- **RQ6.** How effective is our approach in real-world games at preventing aimbots while ensuring invisibility (§ 5.8)?

Table 1: Dataset descriptions collected from CS2 and CF.

Dataset	Map	#frames[1]	#eng.[2]	#targets[3]	aTPF[4]
ACVC-CS2	dust2	10,500	350	13,877	1.322
	anubis	480	16	495	1.031
	mirage	570	19	612	1.074
	vertigo	1,140	38	1,184	1.039
	inferno	420	14	448	1.067
	nuke	960	32	993	1.034
	desert_atrium	150	5	150	1.000
	repository	330	11	331	1.003
	desert_town	270	9	302	1.119
	Total	14,820	494	18,392	1.241
ACVC-CF	coconut_island	16,110	537	16,404	1.018
	aquarium	10,260	342	11,587	1.129
	ship	15,840	528	17,130	1.081
	pyramid	3,360	112	3,557	1.059
	training_ground	2,730	91	3,375	1.236
	stable	7,650	255	7,804	1.020
	dust	1,380	46	1,502	1.088
	Total	57,330	1,911	61,359	1.070

[1]**#frames**: the number of frames;
[2]**#eng.**: the number of engagements (30 frames window per engagement);
[3]**#target**: the number of detected targets;
[4]**aTPF**: the averaged number of targets per frame.

5.1 Experimental Setting

5.1.1 System Configuration

Our experimental setup 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. This configuration, simulating a user client, enables a comprehensive evaluation of our defense mechanism against visual aimbots.

5.1.2 Data Collection

To evaluate the proposed method, we constructed two datasets, termed **A**nti-**C**omputer-**V**ision-**C**heats-**CS2** (**ACVC-CS2**) and **A**nti-**C**omputer-**V**ision-**C**heats-**C**ross**F**ire (**ACVC-CF**). These datasets consist of 72,150 consecutive screenshots from real matchmaking gameplay footage, each cropped to a resolution of 320×320 pixels, aligning with the default capture size for visual aimbots [44]. Specifically, the **ACVC-CS2** dataset contains 14,820 screenshots from actual gameplay in *Counter Strike 2*, while the **ACVC-CF** dataset includes 57,330 consecutive screenshots from *CrossFire*. TThe images were filtered to retain only those identifiable by visual aimbots. Details and characteristics of these datasets are outlined in Table 1.

5.1.3 Proxy Model

We first selected the latest open-source visual aimbot tool [44] as proxy models for simulating cheating models. This tool uses YOLOv5 series models, which are anchor-based and widely recognized as effective object detection models. We also considered NanoDet [41], which stands out in the anchor-free category as a lightweight yet powerful model, boasting 5.5k stars on GitHub. Depending on experimental goals, specific models from the YOLOv5 and NanoDet series were used as proxy models for different evaluations. The performance of our Cloak Crafter on these diverse models underscores its overall effectiveness and adaptability in proactively defending against visual aimbots across different detection frameworks.

5.1.4 Universal Cloak Training

According to our approach described in section 4, we select the top 10 widely used models from YOLOv5 series [22] to train two Universal Cloak on two datasets. During training, we observed that using more models generally improves the Universal Cloak's transferability. However, conflicts between the loss functions of different models make it impractical to increase the number of training models indefinitely. Therefore, we selected models from the YOLOv5 series to ensure a high defense success rate and satisfactory transferability.

5.1.5 Baselines

In our evaluation, we focus on the performance of Cloak Crafter, as it is the primary mechanism to achieve defensive capabilities for Invisibility Cloak. Note that there is no dedicated defense method against visual game cheating. Therefore, to rigorously evaluate Cloak Crafter's effectiveness and efficiency, this section compares its defensive performance with state-of-the-art (SOTA) adversarial attack methods, namely RayS [10], DAG [56] and TOG [12]. Among them, RayS and TOG represent black-box and white-box models, respectively, while DAG is widely acclaimed for its unique approach, which has a very excellent performance. Comparing Cloak Crafter with these models can comprehensively evaluate the efficiency of our method.

5.1.6 Evaluation Metric

In assessing Cloak Crafter, we adopt a multi-dimensional metric, focusing on efficiency, effectiveness, and impact on player experience. These metrics are essential for validating the practical applicability of our approach in mitigating visual aimbots in gaming scenarios.

- ***Query:*** The number of queries to the object detection model that are required to generate a Cloak is a critical measure of computational efficiency. A lower Query count indicates a more efficient algorithm, which is essential for real-time applications.
- ***Frames Per Second (FPS):*** The FPS rate is indicative of the number of frames that Cloak Crafter can process per second. Higher FPS rates signify greater efficiency and smoother integration into gameplay. It should be emphasized that, as we mentioned in § 4.3, the Cloak can be deployed at intervals, and the FPS rate here is the crafting speed of the refined Cloak, not the actual game frame rate.

- ***Defense Success Rate (DSR):*** The effectiveness of Cloak Crafter is quantitatively evaluated by its defense success rate (DSR), which measures the proportion of instances where the method successfully prevents visual cheating attempts.

- ***Structural Similarity (SSIM):*** SSIM [53] is employed to predict the perceived quality of digital images or videos. A higher SSIM score indicates that the Cloak's impact on visual quality is negligible. Additionally, we conduct practical impact analysis in § 5.8 and provide demo displays to further demonstrate the Cloak's minimal visual impact.

5.2 Defensive Performance

To tackle **RQ1**, Table 2 demonstrates the performance of different defense methods on the **ACVC-CS2** and **ACVC-CF** datasets, using imperceptible random noise as a baseline to compare Cloak Crafter and three other SOTA methods. Because random noise is static, it does not have Frames Per Second (FPS) or Query metrics. As shown in the table, random noise exhibits a high SSIM, indicating good visual quality. However, its defense success rates (DSR) are consistently poor across all models and datasets, with most DSRs in the single digits. This demonstrates that random noise is not an effective defense strategy. Cloak Crafter consistently achieves a 100% DSR across all tested anchor-based cheating models (YOLOv5 variants), where RayS shows significantly lower DSRs. Although DAG and TOG display high DSRs, they still fall short of Cloak Crafter's complete success. Even when tested on NanoDet-Plus-m, an anchor-free model not used in training, Cloak Crafter shows excellent DSR, comparable to other methods. Additionally, Cloak Crafter demonstrates remarkable efficiency, reflected in its high FPS rates and low Query counts. By constraining the number of queries, Cloak Crafter's FPS metrics can achieve dozens of times higher than the other defense models. This efficiency suggests that deploying the other three defense methods directly in actual gameplay could significantly hinder the game's fluidity, causing noticeable lag. In contrast, Cloak Crafter's high FPS allows quick generation of refined Cloaks based on the Universal Cloak, covering most of the game frames by our deployment strategy (§ 4.3). Besides, the SSIM results indicate that our method maintains high image quality. However, the perceptibility of perturbations is not solely dependent on SSIM; in human visual perception, large areas of noise with consistent perturbations are more noticeable, even if individual pixel perturbations are small. This explains why RayS has a high SSIM score, but its visualization results are easily noticeable. Please refer to § 5.7.2 for visualization examples.

Furthermore, Table 3 highlights the superior adaptability of Cloak Crafter. We compared different defense models using the same series of visual models as both proxy and cheating models. Although this is a basic transferability test, the table shows that all other defenses, except Cloak Crafter, exhibit markedly low DSR and FPS when tested on YOLO, indicating their failure to protect against even slight model variations. Although RayS achieves 100% DSR on NanoDet, its FPS is so low that it requires hundreds of queries to find a suitable perturbation, which is impossible for real-world applications. Conversely, Cloak Crafter consistently maintained high DSR and FPS. This performance evaluation shows that Cloak Crafter can ensure real-time defense without compromising gameplay fluidity, providing invisible protection. This is in sharp contrast to the other three defense methods.

5.3 Transferability Evaluation

Following our preliminary horizontal transferability tests, we conducted a comprehensive vertical evaluation to delve into Cloak Crafter's transferability, addressing **RQ2**. It's acknowledged that achieving high transferability in adversarial attacks, especially in object detection, remains a significant challenge with no existing method attaining notable success [8, 54]. However, for the specific application of defending visual aimbots, our defense success is not entirely dependent on perfect transferability; rather, it aims to disrupt cheaters' reliance on aimbots, as we mentioned in § 4.3. Against this backdrop, Cloak Crafter exhibits a commendable performance.

Table 4 presents Cloak Crafter's transferability across different models. We select the lightest and most commonly used models, YOLOv5n and NanoDet, as proxy models to ensure the practicality of transferability discussion. Because our Cloak is trained with anchor-based visual models, the DSR test results on anchor-free cheating models such as Faster RCNN, RTMDet, and NanoDet are not as high but still acceptable. The observed data indicate that Cloak Crafter performs well when YOLOv5n is used as the proxy model, achieving an impressive average FPS of 63.67. On the other hand, using NanoDet as the proxy model requires more queries, resulting in a lower FPS. Nevertheless, the overall performance remains strong, with an average DSR of 52.71 and an average FPS of 41.62. These metrics highlight Cloak Crafter's efficiency and transferability across different model architectures, demonstrating its potential for broad real-world applications.

5.4 In-Game Robustness

To answer **RQ3**, we scrutinize Cloak Crafter's in-game robustness through three experiments. These evaluations cover its adaptability to various game maps, effectiveness against single and multiple enemies on-screen, and consistency across different screen resolutions. We use a series of YOLOv5 models according to the popular open-source cheating project [44] to simulate the cheating models. This comprehensive assessment aims to validate Cloak Crafter's reliability and effectiveness in diverse gaming scenarios, ensuring its robust defense mechanism against visual aimbots in realistic conditions.

Table 2: Performance comparison of different defense methods on the **ACVC-CS2** and **ACVC-CF** datasets.

Cheating Model	Dataset	Random Noise		RayS				DAG				TOG				Cloak Crafter			
		DSR(%)	SSIM	DSR(%)	FPS	#Query	SSIM	DSR(%)	FPS	#Query	SSIM	DSR(%)	FPS	#Query	SSIM	DSR(%)	FPS	#Query	SSIM
YOLOv5n	ACVC-CS2	13.90	**0.87**	52.90	1.89	164.04	0.86	**100.00**	16.32	3.82	0.82	99.90	5.74	10.00	0.79	**100.00**	**76.70**	**1.27**	0.74
	ACVC-CF	8.00	0.86	47.70	1.55	156.31	**0.94**	**100.00**	21.82	3.13	0.86	99.90	5.81	10.00	0.85	**100.00**	**99.78**	**1.18**	0.72
YOLOv5s	ACVC-CS2	3.70	**0.87**	50.90	1.74	189.35	0.78	**100.00**	10.76	5.57	0.82	99.80	5.67	10.00	0.79	**100.00**	**71.38**	**1.32**	0.74
	ACVC-CF	1.40	**0.86**	48.70	1.35	183.33	**0.90**	99.90	13.09	4.92	0.86	99.80	5.85	10.00	0.84	**100.00**	**110.67**	**1.19**	0.72
YOLOv5m	ACVC-CS2	2.20	**0.87**	54.90	1.24	205.62	0.36	99.90	6.85	6.72	0.82	99.70	4.48	10.00	0.79	**100.00**	**59.13**	**1.39**	0.74
	ACVC-CF	1.30	**0.86**	53.90	0.97	204.08	0.52	**100.00**	8.63	5.72	**0.86**	99.80	4.69	10.00	0.84	**100.00**	**82.95**	**1.17**	0.71
NanoDet-Plus-m*	ACVC-CS2	17.90	0.87	**100.00**	0.76	168.11	**0.96**	99.90	6.07	4.19	0.82	99.70	2.82	10.00	0.79	99.10	**20.73**	**1.91**	0.75
	ACVC-CF	27.90	0.86	**100.00**	1.11	129.73	**0.97**	**100.00**	12.60	2.47	0.86	**100.00**	2.57	10.00	0.83	**100.00**	**38.93**	**1.34**	0.72

*: Anchor-free models.

Table 3: Transfer performance comparison of different defense methods on the **ACVC-CS2** and **ACVC-CF** datasets.

Model Information		Dataset	RayS			DAG			TOG			Cloak Crater		
Proxy Model	Cheating Model		DSR(%)	FPS	#Query	DSR(%)	FPS	#Query	DSR(%)	FPS	#Query	DSR(%)	FPS	#Query
YOLOv5n	YOLOv5s	ACVC-CS2	10.20	1.40	164.04	6.50	18.07	3.81	44.00	6.39	10.00	**85.50**	**84.29**	**1.27**
		ACVC-CF	17.50	1.63	156.31	3.50	23.72	3.12	38.90	5.51	10.00	**91.50**	**109.90**	**1.18**
NanoDet-Plus-m*	NanoDet-Plus-m-1.5x*	ACVC-CS2	**100.00**	0.88	161.82	18.50	9.79	3.35	61.10	2.74	10.00	62.20	**20.53**	**1.91**
		ACVC-CF	**100.00**	1.14	125.58	32.20	18.21	2.13	74.30	2.52	10.00	82.20	**48.50**	**1.34**

*: Anchor-free models.

Table 4: Transferability of Cloak Crafter on different model architectures.

Proxy Model	Cheating Model	AVCA-CS2			AVCA-CF		
		DSR(%)	FPS	#Query	DSR(%)	FPS	#Query
YOLOv5n	Faster RCNN*	14.00	55.10	1.66	12.60	73.75	1.53
	RTMDet*	22.90	61.17	1.66	46.00	82.68	1.53
	NanoDet*	46.80	48.40	1.66	66.30	77.94	1.53
	YOLOv3n	32.80	56.32	1.66	57.50	78.34	1.53
	YOLOv5n	100.00	60.39	1.66	100.00	65.38	1.53
	YOLOv5s	65.90	55.79	1.66	72.60	65.41	1.53
	YOLOv5m	49.80	59.95	1.66	70.80	64.93	1.53
	YOLOv8n	45.80	49.98	1.66	46.40	63.14	1.53
	Average	47.25	55.89	1.66	59.03	71.44	1.53
Nanodet	Faster RCNN*	13.90	13.29	2.44	12.10	28.40	1.72
	RTMDet*	22.40	13.71	2.44	45.30	30.79	1.72
	NanoDet*	96.70	12.14	2.44	99.80	25.26	1.72
	YOLOv3n	32.10	13.92	2.44	56.50	28.36	1.72
	YOLOv5n	65.40	12.78	2.44	72.60	24.89	1.72
	YOLOv5s	57.00	12.58	2.44	67.10	24.91	1.72
	YOLOv5m	45.30	12.42	2.44	68.20	25.01	1.72
	YOLOv8n	40.20	12.27	2.44	41.80	22.59	1.72
	Average	46.63	12.89	2.44	57.93	26.28	1.72

*: Anchor-free models.

5.4.1 Robustness in Varied Game Scenarios

In this robustness test, we evaluated Cloak Crafter across 9 maps in CS2 and 7 in CF, using three distinct proxy models for each game. The results, detailed in Table 5, reveal consistently high DSRs across all proxy models. YOLOvx6 achieved an exceptional average DSR of 99.92% across 16 maps, while the other two models also exceeded 70% DSRs on average. Furthermore, Cloak Crafter maintained a stable FPS of around 30 in all maps. These outcomes affirm the excellent versatility and operational effectiveness of Cloak Crafter in various maps and game scenarios across different games.

5.4.2 Performance Stability with Single/Multiple Object

Table 6 demonstrates Cloak Crafter's performance across scenarios with varying numbers of characters visible on screen, specifically against popular YOLOv5 series models used in cheating. Note that here we select the most powerful model, YOLOv5x6, as the proxy model to illustrate the improvement in transferability from a powerful proxy model. Similar to the game scenario robustness test, Cloak Crafter performed well on all tested cheating models, maintaining stability whether

Table 5: Robustness of Cloak Crafter across different game scenarios and maps with YOLOv5x6 as a cheating model.

Dataset	Scenario	Proxy Model: YOLOv5n			Proxy Model: YOLOv5s			Proxy Model: YOLOv5x6		
		DSR(%)	FPS	#Query	DSR(%)	FPS	#Query	DSR(%)	FPS	#Query
AVCA-CS2	desert_atrium	93.30	23.48	1.91	96.70	22.46	1.91	100.00	21.58	1.91
	repository	97.80	93.86	1.01	100.00	64.80	1.01	100.00	65.49	1.01
	desert_town	80.70	28.12	1.60	86.70	28.76	1.60	100.00	27.77	1.60
	dust2	78.90	34.24	1.40	85.30	33.75	1.45	100.00	33.12	1.40
	Mirage	61.70	23.21	1.86	55.30	23.65	1.86	100.00	24.17	1.86
	Inferno	70.50	23.01	1.77	75.20	24.27	1.77	100.00	24.79	1.77
	Nuke	54.80	17.96	2.18	42.30	18.20	2.18	99.60	18.10	2.18
	Anubis	43.30	20.88	1.97	45.80	20.89	1.97	100.00	21.33	1.97
	Vertigo	60.00	24.45	1.74	57.70	24.70	1.74	99.80	24.20	1.74
	Average	71.22	32.13	1.72	71.67	29.05	1.72	99.93	28.95	1.71
AVCA-CF	coconut_island	70.20	21.74	1.92	80.00	21.78	1.92	99.90	22.49	1.92
	aquarium	60.30	35.23	1.46	60.20	32.95	1.46	100.00	33.58	1.46
	pyramid	58.80	13.65	2.73	66.90	12.92	2.73	99.90	13.43	2.73
	stable	93.00	32.95	1.49	92.70	30.99	1.49	99.90	38.39	1.49
	ship	70.00	31.05	1.75	77.85	30.20	1.75	99.80	31.02	1.75
	training_ground	53.80	32.40	2.06	72.50	19.23	2.06	100.00	19.19	2.06
	dust	77.80	28.02	1.59	82.80	29.74	1.59	99.90	39.27	1.59
	Average	69.13	27.86	1.86	76.14	25.40	1.86	99.91	28.20	1.86

Table 6: Robustness of Cloak Crafter in multi-object scenarios.

Dataset	Proxy Model	Cheating Model	#Object = 1			#Object ≥ 2		
			DSR(%)	FPS	#Query	DSR(%)	FPS	#Query
AVCA-CS2	YOLOv5x6	YOLOv5n	75.00	33.37	1.49	71.20	23.79	1.53
		YOLOv5s	78.20	31.70	1.49	82.10	22.57	1.53
		YOLOv5m	74.40	31.24	1.49	76.50	23.29	1.53
		YOLOv5l	78.60	31.75	1.49	78.10	21.49	1.53
		YOLOv5x	84.90	33.58	1.49	85.80	23.77	1.53
		YOLOv5n6	49.60	31.49	1.49	27.30	21.72	1.53
		YOLOv5s6	59.80	31.24	1.49	42.50	22.04	1.53
		YOLOv5m6	85.00	30.11	1.49	85.20	23.52	1.53
		YOLOv5l6	88.30	30.75	1.49	87.60	22.05	1.53
		YOLOv5x6	99.80	31.77	1.49	100.00	23.22	1.53
		Average	77.36	31.70	1.49	73.63	22.75	1.53
AVCA-CF	YOLOv5x6	YOLOv5n	57.40	17.30	2.08	41.50	13.69	2.65
		YOLOv5s	68.60	16.74	2.08	59.40	13.65	2.65
		YOLOv5m	64.90	16.61	2.08	51.50	13.66	2.65
		YOLOv5l	68.90	16.05	2.08	52.40	12.65	2.65
		YOLOv5x	73.40	17.01	2.08	56.70	14.02	2.65
		YOLOv5n6	42.80	16.40	2.08	30.30	13.23	2.65
		YOLOv5s6	49.20	16.42	2.08	42.40	13.75	2.65
		YOLOv5m6	71.00	16.59	2.08	53.60	13.39	2.65
		YOLOv5l6	70.30	17.16	2.08	53.00	13.39	2.65
		YOLOv5x6	99.60	16.98	2.08	100.00	12.83	2.65
		Average	66.61	16.72	2.08	54.08	13.43	2.65

Table 7: Robustness of Cloak Crafter across different resolutions with YOLOv5n as a local proxy model.

Dataset	Resolution	Cheating Model	DSR(%)	FPS	#Query
AVCA-CS2	224×224	YOLOv5n	99.90	40.00	2.28
		YOLOv5m	52.20	21.15	2.75
		YOLOv5x	46.40	16.91	2.54
		Average	66.17	26.02	2.52
	320×320*	YOLOv5n	100.00	77.94	1.51
		YOLOv5m	75.80	45.74	1.57
		YOLOv5x	74.40	44.31	1.44
		Average	83.40	55.99	1.51
	416×416	YOLOv5n	99.90	47.15	1.73
		YOLOv5m	73.60	32.05	2.12
		YOLOv5x	68.50	19.15	2.27
		Average	80.67	32.78	2.04
AVCA-CF	224×224	YOLOv5n	99.90	41.65	2.30
		YOLOv5m	47.70	30.25	2.61
		YOLOv5x	41.40	18.46	2.38
		Average	63.00	30.12	2.43
	320×320*	YOLOv5n	100.00	71.97	1.55
		YOLOv5m	74.20	58.61	1.51
		YOLOv5x	73.10	40.43	1.47
		Average	82.43	57.00	1.51
	416×416	YOLOv5n	100.00	50.96	1.98
		YOLOv5m	61.50	32.96	2.93
		YOLOv5x	54.80	16.91	2.24
		Average	72.10	33.61	2.38

*: The baseline resolution.

one or multiple characters were present. These results show that Cloak Crafter can consistently operate in real-world gaming scenarios, even in complex multiplayer battles.

5.4.3 Consistency Across Different Resolutions

This subsection delves into Cloak Crafter's effectiveness across varying game resolutions, recognizing that players employ diverse settings in real gameplay. By simulating different screen resolutions through image resizing, with a baseline of 320×320, we rigorously tested Cloak Crafter's defensive capabilities. The results, detailed in Table 7, indicate that Cloak Crafter maintains its performance without significant degradation, whether at higher or lower resolutions than the baseline. It is worth mentioning that our method demonstrates enhanced efficacy in scenarios of image magnification, surpassing its performance in reduction contexts.

5.5 Ablation Study

In this subsection, we examine the critical components of our Invisibility Cloak's efficacy, including the necessity of Universal Cloak, the choice of proxy model, initialization parameters, and the noise amplitude to answer **RQ4**.

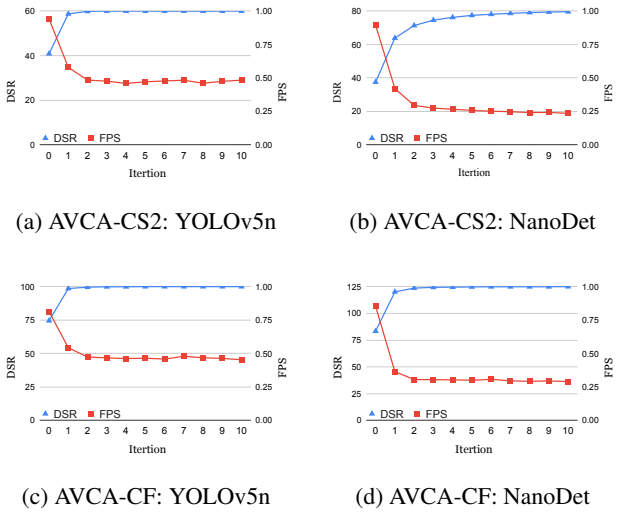
5.5.1 The Effect of Proxy Model and Universal Cloak

We first discuss the impact of Universal Cloak and different local proxy models. Since our approach is based on the successful defense of the proxy model with Invisibility Cloak, the generalization of the local model is crucial in our approach.

Table 8: Performance comparison of different proxy models.

Dataset	Proxy Model	w/o Universal Cloak			w/ Universal Cloak		
		DSR(%)	FPS	#Query	DSR(%)	FPS	#Query
AVCA-CS2	YOLOv5n	99.80	19.19	2.17	100.00	40.14	1.51
	YOLOv5x	23.70	8.53	3.24	74.40	40.33	1.44
	YOLOv5x6	26.60	6.70	3.47	74.50	24.30	1.53
	NanoDet-Plus-m*	17.10	12.98	2.17	60.9	13.28	2.99
AVCA-CF	YOLOv5n	100.00	29.25	2.10	100.00	46.92	1.55
	YOLOv5x	14.60	9.23	3.23	73.10	24.33	1.51
	YOLOv5x6	18.40	6.70	3.56	72.20	17.19	1.73
	NanoDet-Plus-m*	10.80	19.54	1.86	64.40	17.47	2.42

*: Anchor-free models.



(a) AVCA-CS2: YOLOv5n (b) AVCA-CS2: NanoDet

(c) AVCA-CF: YOLOv5n (d) AVCA-CF: NanoDet

Figure 4: Variation of DRS and FPS with the number of initialization iterations.

We believe game cheaters prefer using powerful and widely used lightweight models such as YOLOv5n. Therefore, we use this model to simulate the cheating model in our experiments. As shown in Table 8, our investigation employs a variety of proxy models to compare their performance with and without our Universal Cloak. The results show that, the presence of the Universal Cloak consistently enhances the performance across all evaluated models, emphasizing its importance in our defense approach. Besides, when the proxy model has a similar architecture to the cheating model, such as the YOLOv5 series, the generalization of the proxy model is better. Additionally, it can be deduced that using a more powerful proxy model leads to better transfer defense, such as the better transfer performance of YOLOv5x6 compared to YOLOv5x in the table. Therefore, the results show that when selecting a proxy model, we should choose a model series that cheaters are more likely to choose and a more powerful model, resulting in better invisibility performance.

5.5.2 Initialization

In this subsection, we explore the influence of two hyperparameters set during Cloak Refinement § 4.2. Since the

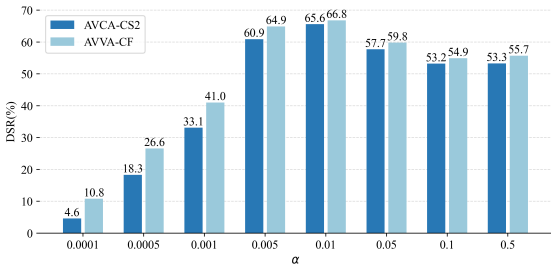


Figure 5: The impact of learning rate α on DSR.

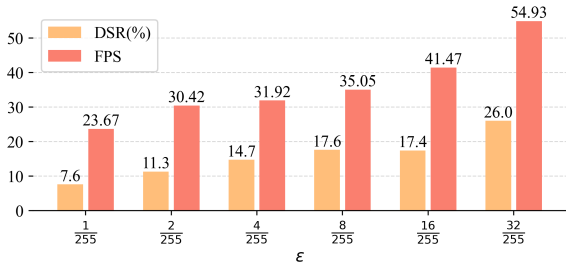


Figure 6: The impact of epsilon ε on DSR and FPS.

refinement process must be completed in real-time, these hyperparameters significantly impact the final performance.

Firstly, we discuss the maximum iteration during refining. When encountering bad cases, an increase in the maximum iterations correlates with an augmentation in the defense duration (although from the Table 2 to Table 7, the results show that in most cases, only 1-3 queries are needed for successful defense). We select two proxy models of different architectures and the same cheating model. Figure 4 shows that establishing an upper limit of approximately 5 iterations ensures the preservation of a Detection Success Rate (DSR) close to 100% while also maintaining a high Frames Per Second (FPS) rate. A further increment in iterations results in a marginal enhancement in DSR at the expense of a reduced FPS. Consequently, if the DSR satisfy the minimum criteria, it is advisable to refrain from excessive iteration limits to prevent diminishing returns in terms of performance.

We then discuss the impact of learning rate α on Cloak Refinement. We choose YOLOv5n as the proxy model and YOLOv8n as the cheating model without a Universal Cloak. Note that we fixed the number of 20 iterations without an early stop to more clearly show the effect of the learning rate on DSR. As shown in Figure 5, the results show that a learning rate of 0.1 or 0.005 is appropriate. The reason for choosing 0.005 as the default is that we want to minimize the size of the Cloak to make it imperceptible without affecting DSR.

5.5.3 Amplitude of Noise

Finally, we explore the relationship between the size of the Cloak and its performance metrics. Typically, noise below 8/255 epsilon (ε) is not easily detectable by human eyes in RGB images. In our ablation experiments, we set different Cloak epsilon values as defined in Equation 1. Figure 6 demonstrates how varying the maximum ℓ_∞-norm constraint (ε) of the Cloak affects the DSR and FPS on the AVCA-CF dataset. Although a larger-sized Cloak has a higher DSR, it also means a greater impact on the image. While maintaining satisfactory FPS, we need to ensure a stable DSR and that the Cloak remains imperceptible to the player. In this experiment, we also chose YOLOv5n as the proxy model and YOLOv8n as the cheating model without Universal Cloak, and we fixed the number of iterations to a default of 5 with an early stopping criterion. The results show that setting ε to 8/255 is more appropriate, with a sufficiently high DSR without a significant decrease in FPS. Notably, when ε is set to 16/255, the DSR for transfer defense actually decreases, which means that a higher ε does not necessarily improve Cloak transferability.

5.6 Adaptive Attack

To address **RQ5**, we examine the effectiveness of our approach when cheaters realize the Invisibility Cloak and attempt to counteract it. First, we employ a straightforward experiment that adds random noise before and after the Cloak to evaluate its robustness to such noise. As shown in Table 9, it indicates that random noise impacts the DSR by less than 5%. In most cases, Cloak Crafter maintains a DSR of over 95% while achieving satisfactory FPS and Query performance.

Table 9: The performance of Cloak Crafter under basic adaptive attack using random noise.

Dataset	Cheating Model	Pre-Cloak Random Noise			Post-Cloak Random Noise		
		DSR(%)	FPS	#Query	DSR(%)	FPS	#Query
AVCA-CS2	YOLOv5n	100.00	64.77	1.87	97.20	54.86	1.66
	YOLOv5s	100.00	54.01	2.10	94.80	49.40	1.80
	YOLOv5m	100.00	34.10	2.38	90.50	33.53	2.08
	NanoDet-Plus-m*	96.70	14.02	2.57	78.00	15.17	2.44
AVCA-CF	YOLOv5n	99.90	64.77	1.77	98.60	77.57	1.53
	YOLOv5s	100.00	52.55	2.00	98.20	71.75	1.62
	YOLOv5m	99.90	46.38	1.93	97.90	58.04	1.62
	NanoDet-Plus-m*	99.80	26.63	1.88	94.40	29.91	1.72

*: Anchor-free models.

Additionally, we conduct further adaptive attack evaluations. We fine-tune the visual aimbot [44] on our datasets and perform adversarial training with protected samples on top of the fine-tuned models to simulate cheaters' resistance. Note that we use 20% of the datasets for adversarial training because we assume cheaters have much fewer collected samples than protectors. As shown in Table 10, adversarial training reduces the DSR by less than 20% on the AVCA-CS2 dataset, yet it fails to attack our defense on the AVCA-CF. We believe this is because CF has simple scenarios and consistent graphics, which results in only a limited improvement of the cheating model's robustness after adversarial training. Notice that when $\varepsilon = 8/255$ on the AVCA-CF dataset, the DSR is slightly below 100%. This could be attributed to fine-tuning, which improves the cheating model's performance, resulting in a more challenging situation for our Cloak. However, the

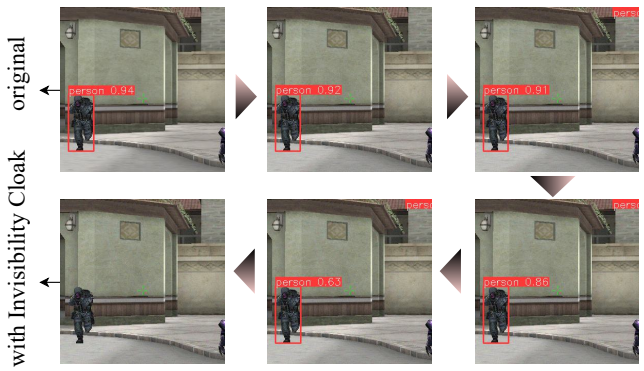


Figure 7: The refinement process of Invisibility Cloak in CF.

simple scenarios in CF conflict with adversarial samples [14], which allows our DSR to instead reach 100% after adversarial training. Further discussion is provided in section 6.

Table 10: Performance under adaptive attack using adversarial training with fine-tuned YOLOv5n as a local proxy model.

Dataset	Cheating Model	ε = 8/255			ε = 16/255		
		DSR(%)	FPS	#Query	DSR(%)	FPS	#Query
AVCA-CS2	YOLOv5n+	100.00	31.30	2.26	100.00	86.05	1.05
	YOLOv5n+AT	81.70	31.30	2.26	86.00	93.49	1.05
AVCA-CF	YOLOv5n+	99.80	19.24	3.22	100.00	65.25	1.32
	YOLOv5n+AT	100.00	19.44	3.22	100.00	65.78	1.32

+: Models that have been fine-tuned;
AT: Models that have been adversarial trained.

5.7 Case Study

5.7.1 The Refinement Process of Invisibility Cloak

In this subsection, we first visualize the refinement process of our Invisibility Cloak through a demo in Figure 7. As we describe in § 4.3, the deployment of Universal Cloak can achieve real-time defense with no latency, but some bad cases still require iterative refinement of the Cloak. As a result, after we craft a Universal Cloak, the refinement process is very important for practical applications. Figure 7 visualizes this process; it demonstrates the bounding box's confidence score of the target in the image keeps decreasing through our refinement until the cheating model cannot detect it.

Next, we discuss how to craft a Cloak for an input game frame. To achieve this, we reduce the confidence score of the bounding box detecting the target in the frame by minimizing the loss function (Equation 3). Figure 8 illustrates the decrease in loss during the Cloak searching process. The notations in Figure 8 indicate successful defense of YOLOv5n and YOLOv8n at the 2nd and 7th iterations, respectively. Here, we fixed the number of iterations to 20 without early termination. In our experiments, we observed that among all samples with successful YOLOv8n defense, only one sample preceded the successful YOLOv5n defense, accounting for less than 1% of the total. This indicates the difficulty of defending against unknown models using white-box gradient information. Although we can use multiple commonly used white-box cheating models to design loss functions and co-construct the gradient, the forward and backward overhead in these additional models is impractical for real-time defense scenarios. Therefore, in our approach, the Universal Cloak described in section 4 effectively addresses the challenge of generalizability.

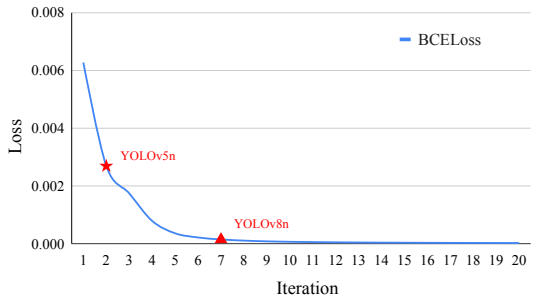


Figure 8: The loss decreases with iterations until the cheating model is successfully defended.

5.7.2 Normal/Cheating Player's Perspective

Figure 9 presents visualization examples of different methods under the ℓ_∞-norm constraint with $\varepsilon = 8/255$ in real game scenarios. It can be observed that despite having the same constraint on individual pixel perturbations, different methods yield different visualization results. In particular, although RayS has a higher SSIM, as shown in Table 2, it exhibits large areas of similar perturbations, making it more noticeable to players. While our method does not outperform others in the SSIM evaluation metric, the invisibility of our Invisibility Cloak is evident from the figure.

5.8 Practical Impact Analysis

We perform user study and real-world effect verification to answer **RQ6** and demonstrate that our Cloak is not perceived by players, thereby not impacting the game experience.

5.8.1 User Study

We conduct a paid anonymous survey in the form of a questionnaire to evaluate the **indistinguishability** and **naturalness** of game screens protected by our Cloak (randomly selected from a subset of experimental data). The data used were refined Cloaks adhering to the standards outlined in section 4. All 122 participants are over 18 years old, and 80% of them have shooting games experience. We followed best practices for ethical human subjects survey research, and we did not collect unnecessary personal information. Preliminary research indicates that it is challenging for people to determine whether a game scene has been added with Cloaks.

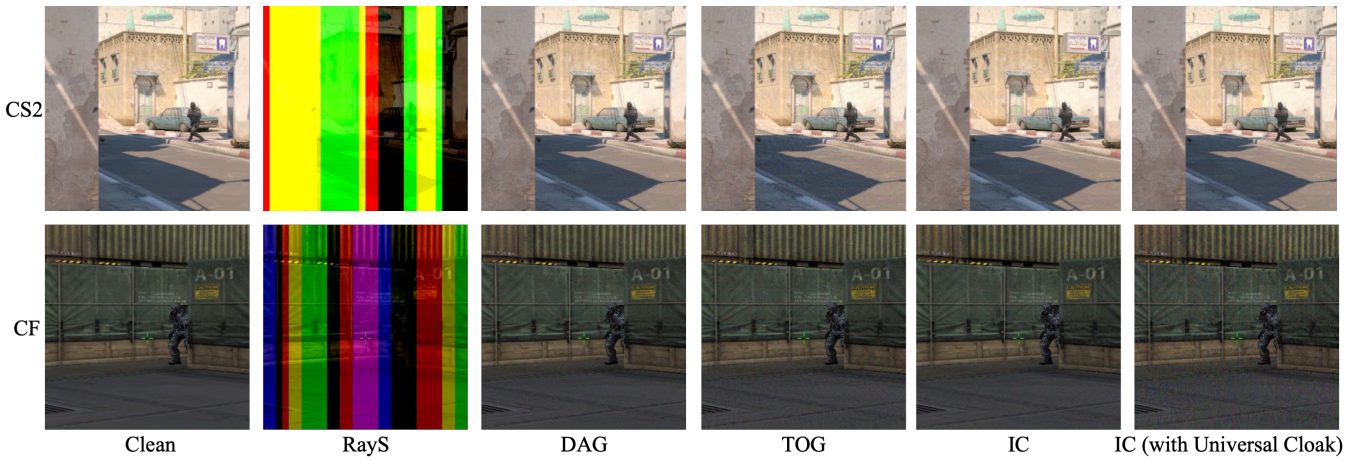


Figure 9: Visual examples of different methods in two popular First-person Shooter games, CS2 and CF, under the ℓ_∞-norm constraint with $\varepsilon = 8/255$. IC denotes our Invisibility Cloak.

Indistinguishability. The study consists of two main types of tasks to evaluate the indistinguishability of the protected game screen. The first type involved binary choices, where participants were asked to select a true game scene between two options: one with the Cloak and the other without any modifications. The total accuracy for this task was 47.87%, which is lower than the random guess threshold of 50%. The second type of task required participants to judge the authenticity of pairs of game scenes, with four possible combinations: both true, both false, one true and one false (in either order). The overall proportion of correct answers was 23.77%, below the random guess threshold of 25%. This indicates that participants cannot perceive our Cloak, thereby demonstrating the Cloak's indistinguishability.

Naturalness. We use the task of choosing the naturalness of the game screen to evaluate the protected game screen's naturalness. Participants are asked to choose the naturalness of a randomly selected screen (both the original screen and the protected screen) from five options, as shown in Figure 10. The Pearson correlation coefficient between original and protected frames is 0.983 (with $p = 0.0027 < 0.05$). This provides strong statistical evidence that the correlation between original and protected frames is not accidental, illustrating the naturalness of our Cloak.

Figure 10: Results of the user study on naturalness.

Additionally, we use multiple-choice questions to assess participants' perceptions of the cheat's reliability. Only 9.3% of participants would continue using a visual aimbot when its success rate is below 70%, while 27.8% indicated that this accuracy rate needed to be at least 90%. These findings highlight the importance of aimbot accuracy for users, suggesting that even minor performance deviations could lead to user abandonment, consistent with the discussion in § 4.3.

5.8.2 Real-World Effect Verification

Evaluating our method in real-world scenarios involves the activation of visual aimbots within genuine game matches. To address ethical concerns and control variables, we adopted video feeds instead of directly implementing the SOTA visual aimbot [44] in the game. The two most effective settings among all Universal Cloak configurations for CS2 and CF, pre-trained with different combinations of YOLOv5, were selected and superimposed onto the corresponding real-time game video feed for live verification purposes.

Table 11 shows two open-sourced real-world whole-match video feeds of CS2 [4] and CF [5] used for the evaluation. Note that this additional dataset is completely different from the previous evaluation dataset. In this experiment, over approximately 39.5 minutes of gameplay for CS2 and 29.4 minutes for CF, thousands of auto-aim and aimbot detections were reduced by three orders of magnitude. By using our Invisibility Cloak, auto-aim and aimbot detections were controlled to single digits, effectively preventing visual cheating.

Table 11: Anti-Cheat effect verification with real-world match examples.

Game	Duration(sec)	w/o Invisibility Cloak		w/ Invisibility Cloak	
		#Detection	#Auto-aim	#Detection	#Auto-aim
CS2	2,370.61	3,860	3,549	**5**	**5**
CF	1,765.69	2,394	2,197	**2**	**2**

[1] **#Detection**: the number of successful detections by aimbot;
[2] **#Auto-aim**: the number of successful auto-aiming by aimbot.

[4] https://www.youtube.com/watch?v=iUwdyHcpUe8
[5] https://www.youtube.com/watch?v=UdKwp7O4Uq4

6 Discussion and Limitation

In this section, we explore avenues for future enhancement. Despite § 5.3 demonstrating the superiority of our method over other baselines and its defense effectiveness, there is potential for further optimization of our Cloak.

Generalization of Universal Cloak. Since we can only use a limited number of models to train our Universal Cloak, yet need to contend with an unknown and infinite variety of cheating models, incorporating regularization into the training process may improve its generalization performance.

Transferability of Cloak Refinement. For instance, we observe that YOLOv8 often requires more iterations than YOLOv5 for a successful defense. Setting a low query count to expedite Cloak generation led to decreased defense effectiveness. This indicates that adjusting the number of iterations based on different cheating models and FPS requirements could improve defense effectiveness and transferability.

Adaptive Attack. Despite the results from § 5.6 showing that adaptive attacks cannot significantly break through our Cloak, if the cheater collects sufficient adversarial samples and adopts more targeted fine-tuning strategies, our defense may become vulnerable. Therefore, we can design inducive perturbations for the fine-tuned model (such as identifying additional wrong targets), which we leave as future work.

7 Related Work

In this section, we introduce two types of anti-cheating techniques: detection-based and prevention-based.

Detection-based techniques predominantly employ conventional security measures to passively identify anomalies, such as irregular processes, memory reads and writes, and the signature codes of cheating malware [16, 31]. Contemporary advancements have incorporated machine-learning and deep-learning strategies for statistical and behavioral anomaly detection [2,11,17,23,25,29,34,37,63]. Additionally, computer-vision-driven methods are utilized to identify illicit overlays or patterned trajectories employed by cheat scripts [24, 48]. Although these approaches are noted for their effectiveness and efficiency in cheat detection, they inherently lag behind cheating activities because the infractions and consequent damages have already occurred.

In contrast, prevention-based techniques are active in nature. However, they are still in the nascent stages of development, with limited related literature for reference. Traditional methods, dating back a decade, focused on core data access control, authentication, and network-based protections [4,57]. More recent efforts have explored using Trusted Execution Environments to safeguard core data and applying randomized binaries for preventive measures [3,6,38]. However, these approaches often require specific hardware, such as Intel SGX enclaves, limiting their applicability to all users. Furthermore, some approaches can not effectively address certain attack methodologies, such as the DLL injection, which remains prevalent in contemporary cheating strategies.

For visual aimbots, the traditional detection and prevention methodologies face inherent limitations due to the unique nature of these novel cheats. Unlike conventional cheating aimbots that manipulate game memory or data, visual aimbots operate by analyzing the visual output of the game, rendering traditional game memory-related detection methods inapplicable. Similarly, traditional prevention methods are ineffective against visual sights because these cheats do not require altering game files or exploiting network vulnerabilities, thus bypassing the usual security checkpoints. Therefore, the only feasible approaches to counter visual aimbots are detection-based methods such as player behavior classification [2, 17, 25, 29] and screen anomalies detection [23, 34, 63]. However, for the former, these methods could misclassify some high-level players as cheaters, resulting in a very low accuracy rate. Moreover, such methods lack explainability and thus require additional human review. For the latter, visual aimbots can hide the detection box on the screen, which means there will be no anomaly on the screen at all.

Given these challenges, there's a clear gap in the effectiveness of existing anti-cheat approaches against visual aimbots. This emphasizes the need for a novel solution to proactively counteract advanced cheating strategies without relying on after-the-fact detection or extra human verification. In response, our work introduces *Invisibility Cloak*, a proactive defense mechanism specifically designed to address the challenges of visual aimbots. Moreover, although our approach may impact computer-vision-driven detection methods due to the Cloak perturbations, it can still work in parallel with traditional detection methods. By employing Invisibility Cloak for proactive defense, we can block most visual aimbots. Subsequently, detection-based techniques can identify and address the few remaining cheats, thus ensuring comprehensive protection and enhancing the robustness of anti-cheat systems.

8 Conclusion

In this paper, we introduce *Invisibility Cloak*, the first proactive defense framework against visual aimbot cheating in First-person Shooter games, which uses imperceptible perturbations to disrupt computer vision-based aimbot detection without impairing player experience. Our comprehensive evaluation across various game scenarios, object detection models, and real-world investigation demonstrates the technique's effectiveness, efficiency, and adaptability. Invisibility Cloak addresses the challenge of visual aimbots, a problem that traditional anti-cheat methods fail to solve, marking a significant advancement in ensuring game security. This innovative approach offers a new paradigm for combating the inevitable rise of AI-driven cheating techniques in gaming. Our method guarantees real-time defense without compromising the fluidity of gameplay, achieving invisible protection.

9 Acknowledgement

We thank the anonymous reviewers and shepherd for their constructive comments on improving this research. This work is supported by the NSFC for Young Scientists of China (No. 62202400). Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSFC.

References

[1] Your move: Tackling video game cheating head on, 2022. Accessed: 2024-01-26.

[2] Hashem Alayed, Fotos Frangoudes, and Clifford Neuman. Behavioral-based cheating detection in online first person shooters using machine learning techniques. In *2013 IEEE conference on computational inteligence in games (CIG)*, pages 1–8. IEEE, 2013.

[3] Md Sakib Anwar, Chaoshun Zuo, Carter Yagemann, and Zhiqiang Lin. Extracting threat intelligence from cheat binaries for anti-cheating. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*, RAID '23, page 17–31, New York, NY, USA, 2023. Association for Computing Machinery.

[4] N. Baughman, Marc Liberatore, and B. Levine. Cheat-proof playout for centralized and peer-to-peer gaming. *IEEE/ACM Transactions on Networking*, 15:1–13, 2007.

[5] Aduen Benjumea, Izzeddin Teeti, Fabio Cuzzolin, and Andrew Bradley. Yolo-z: Improving small object detection in yolov5 for autonomous vehicles. *arXiv preprint arXiv:2112.11798*, 2021.

[6] André Brandão, João S. Resende, and Rolando Martins. Employment of secure enclaves in cheat detection hardening. In *Trust, Privacy and Security in Digital Business: 17th International Conference, TrustBus 2020, Bratislava, Slovakia, September 14–17, 2020, Proceedings*, page 48–62, Berlin, Heidelberg, 2020. Springer-Verlag.

[7] Antonio Brunetti, Domenico Buongiorno, Gianpaolo Francesco Trotta, and Vitoantonio Bevilacqua. Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing*, 300:17–33, 2018.

[8] Zikui Cai, Xinxin Xie, Shasha Li, Mingjun Yin, Chengyu Song, Srikanth V Krishnamurthy, Amit K Roy-Chowdhury, and M Salman Asif. Context-aware transfer attacks for object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 149–157, 2022.

[9] River's Educational Channel. I tried to make a valorant ai using computer vision. https://www.youtube.com/watch?v=LXA7zXVz8A4, 2021.

[10] Jinghui Chen and Quanquan Gu. Rays: A ray searching method for hard-label adversarial attack. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1739–1747, 2020.

[11] Minyeop Choi, Gihyuk Ko, and Sang Kil Cha. Botscreen: Trust everybody, but cut the aimbots yourself. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 481–498, 2023.

[12] Ka-Ho Chow, Ling Liu, Mehmet Emre Gursoy, Stacey Truex, Wenqi Wei, and Yanzhao Wu. Tog: targeted adversarial objectness gradient attacks on real-time object detection systems. *arXiv preprint arXiv:2004.04320*, 2020.

[13] Victor Costan and Srinivas Devadas. Intel sgx explained. *Cryptology ePrint Archive*, 2016.

[14] Ziyi Dong, Pengxu Wei, and Liang Lin. Adversarially-aware robust object detector. In *European Conference on Computer Vision*, pages 297–313. Springer, 2022.

[15] Hao-Shu Fang, Jiefeng Li, Hongyang Tang, Chao Xu, Haoyi Zhu, Yuliang Xiu, Yong-Lu Li, and Cewu Lu. Alphapose: Whole-body regional multi-person pose estimation and tracking in real-time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[16] Wu-chang Feng, Ed Kaiser, and Travis Schluessler. Stealth measurements for cheat detection in on-line games. In *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, pages 15–20, 2008.

[17] Luca Galli, Daniele Loiacono, Luigi Cardamone, and Pier Luca Lanzi. A cheating detection framework for unreal tournament iii: A machine learning approach. In *2011 IEEE Conference on Computational Intelligence and Games*, pages 266 – 272, 10 2011.

[18] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[19] Grantolope. Ai yolo aimbot apex. https://www.unknowncheats.me/forum/downloads.php?do=file&id=40016, 2023.

[20] OMTP Hardware Working Group et al. Omtp hardware requirements and defragmentation. *Trusted Environment OMTP TR0. Open Mobile Terminal Platform*, 2006.

[21] Shuai Jia, Yibing Song, Chao Ma, and Xiaokang Yang. Iou attack: Towards temporally coherent black-box adversarial attack for visual object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6709–6718, 2021.

[22] Glenn Jocher. YOLOv5 by Ultralytics, May 2020.

[23] Aditya Jonnalagadda, Iuri Frosio, Seth Schneider, Morgan McGuire, and Joohwan Kim. Robust vision-based cheat detection in competitive gaming. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 4(1):1–18, 2021.

[24] Aditya Jonnalagadda, Iuri Frosio, Seth Schneider, Morgan McGuire, and Joohwan Kim. Robust vision-based cheat detection in competitive gaming. *CoRR*, abs/2103.10031, 2021.

[25] Salman Khalifa. Machine learning and anti-cheating in fps games. *Master's thesis*, 2016.

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] lenovo. What is an fps? https://www.lenovo.com/my/en/glossary/fps/?orgRef=https%253A%252F%252Fwww.google.com%252F, 2024.

[28] Daiping Liu, Xing Gao, Mingwei Zhang, and Haining Wang. Shoot for the moon and you will never miss: Characterizing and detecting aimbots in online games. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*, pages 401–402, 2016.

[29] Daiping Liu, Xing Gao, Mingwei Zhang, Haining Wang, and Angelos Stavrou. Detecting passive cheats in online games via performance-skillfulness inconsistency. In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 615–626. IEEE, 2017.

[30] Xin Liu, Huanrui Yang, Ziwei Liu, Linghao Song, Hai Li, and Yiran Chen. Dpatch: An adversarial patch attack on object detectors. *arXiv preprint arXiv:1806.02299*, 2018.

[31] Anton Maario, Vinod Kumar Shukla, A Ambikapathy, and Purushottam Sharma. Redefining the risks of kernel-level anti-cheat in online gaming. In *2021 8th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 676–680. IEEE, 2021.

[32] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[33] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017.

[34] Anh Nhu, Hieu Phan, Chang Liu, and Xianglong Feng. A comprehensive defense approach targeting the computer vision based cheating tools in fps video games. In *2023 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 168–177. IEEE, 2023.

[35] NTUYWANG103. Yolov7 based apex aimbot. https://github.com/NTUYWANG103/APEX_AIMBOT, 2023.

[36] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[37] J. Park, Mee Lan Han, and H. Kim. A study of cheater detection in fps game by using user log analysis. 15:177–188, 2015.

[38] Seonghyun Park, Adil Ahmad, and Byoungyoung Lee. Blackmirror: Preventing wallhacks in 3d online fps games. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 987–1000, New York, NY, USA, 2020. Association for Computing Machinery.

[39] José Pedro Pinto, André Pimenta, and Paulo Novais. Deep learning and multivariate time series for cheat detection in video games. *Machine Learning*, 110(11-12):3037–3057, 2021.

[40] pylesson. Counter-strike global offensive realtime yolov4 object detection aimbot. https://pylessons.com/YOLOv4-TF2-CSGO-aimbot#google_vignette, 2020.

[41] RangiLyu. Nanodet-plus: Super fast and high accuracy lightweight anchor-free object detection model. real-time on mobile devices. https://github.com/RangiLyu/nanodet?tab=readme-ov-file, 2021.

[42] Jacob Roach. The best cloud gaming services of 2024: Wave of the future. https://www.cloudwards.net/top-five-cloud-services-for-gamers/, 2024.

[43] RootKit. Ai aimbot launcher full explanation & ai aimbot international 2024. https://www.youtube.com/watch?v=gEdIocq_W2o, 2024.

[44] RootKit-Org. World's best ai aimbot. https://github.com/RootKit-Org/AI-Aimbot, 2023.

[45] RootKit-Org. Rootkit: a non-profit with a mission to certify and educate the next generation of aspiring developers. https://github.com/RootKit-Org, 2024.

[46] Saeed Shafiee Sabet, Steven Schmidt, Saman Zadtootaghaj, Babak Naderi, Carsten Griwodz, and Sebastian Möller. A latency compensation technique based on game characteristics to mitigate the influence of delay on cloud gaming quality of experience. In *Proceedings of the 11th ACM Multimedia Systems Conference*, pages 15–25, 2020.

[47] Statista. Video games - worldwide. https://www.statista.com/outlook/dmo/digital-media/video-games/worldwide, 2023.

[48] Yueyang Su, Di Yao, Xiaokai Chu, Wenbin Li, Jingping Bi, Shiwei Zhao, Runze Wu, Shize Zhang, Jianrong Tao, and Hao Deng. Few-shot learning for trajectory-based mobile game cheating detection. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 3941–3949, New York, NY, USA, 2022. Association for Computing Machinery.

[49] Torben Teepe, Ali Khan, Johannes Gilg, Fabian Herzog, Stefan Hörmann, and Gerhard Rigoll. Gaitgraph: Graph convolutional network for skeleton-based gait recognition. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2314–2318. IEEE, 2021.

[50] Ultralytics. 'you only look once', yolov5. https://github.com/ultralytics/yolov5, 2020.

[51] unity. Performance profiling tips for game developers. https://unity.com/how-to/best-practices-for-profiling-game-performance, 2024.

[52] Sergio A Velastin and Diego A Gómez-Lira. People detection and pose classification inside a moving train using computer vision. In *Advances in Visual Informatics: 5th International Visual Informatics Conference, IVIC 2017, Bangi, Malaysia, November 28–30, 2017, Proceedings 5*, pages 319–330. Springer, 2017.

[53] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[54] Xingxing Wei, Siyuan Liang, Ning Chen, and Xiaochun Cao. Transferable adversarial attacks for image and video object detection. *arXiv preprint arXiv:1811.12641*, 2018.

[55] Zuxuan Wu, Ser-Nam Lim, Larry S Davis, and Tom Goldstein. Making an invisibility cloak: Real world adversarial attacks on object detectors. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 1–17. Springer, 2020.

[56] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1369–1378, 2017.

[57] Amir Yahyavi, Kévin Huguenin, Julien Gascon-Samson, Jörg Kienzle, and Bettina Kemme. Watchmen: Scalable cheat-resistant support for distributed multi-player online games. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*, pages 134–144, 2013.

[58] Fan Yang, Yang Wu, Sakriani Sakti, and Satoshi Nakamura. Make skeleton-based action recognition model smaller, faster and better. In *Proceedings of the 1st ACM International Conference on Multimedia in Asia*, pages 1–6, 2019.

[59] Siu Fung Yeung, John Lui, Jiangchuan Liu, and Jeff Yan. Detecting cheaters for multiplayer games: theory, design and implementation. 2006.

[60] Su-Yang Yu, Nils Hammerla, Jeff Yan, and Peter Andras. Aimbot detection in online fps games using a heuristic method based on distribution comparison matrix. In *Neural Information Processing: 19th International Conference, ICONIP 2012, Doha, Qatar, November 12-15, 2012, Proceedings, Part V 19*, pages 654–661. Springer, 2012.

[61] Su-Yang Yu, Nils Hammerla, Jeff Yan, and Peter Andras. A statistical aimbot detection method for online fps games. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2012.

[62] Bohang Zhang, Tianle Cai, Zhou Lu, Di He, and Liwei Wang. Towards certifying l-infinity robustness using neural networks with l-inf-dist neurons. In *International Conference on Machine Learning*, pages 12368–12379. PMLR, 2021.

[63] Shiwei Zhao, Jiaheng Qi, Zhipeng Hu, Han Yan, Runze Wu, Xudong Shen, Tangjie Lv, and Changjie Fan. Vespa: A general system for vision-based extrasensory perception anti-cheating in online fps games. *IEEE Transactions on Games*, 2023.

[64] Yue Zhao, Hong Zhu, Ruigang Liang, Qintao Shen, Shengzhi Zhang, and Kai Chen. Seeing isn't believing: Towards more robust adversarial attack against real world object detectors. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 1989–2004, 2019.