GlitchHiker: Uncovering Vulnerabilities of Image Signal Transmission with IEMI
Qinhong Jiang, Xiaoyu Ji, Chen Yan, Zhixin Xie, Haina Lou, and Wenyuan Xu, Zhejiang University
https://www.usenix.org/conference/usenixsecurity23/presentation/jiang-qinhong

This paper is included in the Proceedings of the 32nd USENIX Security Symposium.
August 9–11, 2023 • Anaheim, CA, USA
978-1-939133-37-3

Open access to the Proceedings of the 32nd USENIX Security Symposium is sponsored by USENIX.
GlitchHiker: Uncovering Vulnerabilities of Image Signal Transmission with IEMI

Qinhong Jiang  
Zhejiang University  
Xiaoyu Ji  
Zhejiang University  
Chen Yan*  
Zhejiang University  
Zhixin Xie  
Zhejiang University  
Haina Lou  
Zhejiang University  
Wenyuan Xu  
Zhejiang University

Abstract

Cameras have evolved into one of the most critical gadgets in various applications. In this paper, we identify a new class of vulnerabilities involving the hitherto disregarded image signal transmission phase and explain the underlying principles of camera glitches for the first time. Based on the vulnerabilities, we design the GlitchHiker attack that can actively induce controlled glitch images of a camera at various positions, widths, and numbers using intentional electromagnetic interference (IEMI). We successfully launch the GlitchHiker attack on 8 off-the-shelf camera systems in 5 categories in their original packages at a distance of up to 30 cm. Experiments with 2 case studies involving 4 object detectors and 2 face detectors show that injecting one ribboning suffices to hide, create or alter objects and persons with a maximum success rate of 98.5% and 80.4%, respectively. Then, we discuss real-world attack scenarios and perform preliminary investigations on the feasibility of targeted attacks. Finally, we propose hardware- and software-based countermeasures.

1 Introduction

The proliferation of cameras enables many applications that rely on images to make critical decisions, e.g., autonomous vehicles with object detection and surveillance camera systems with face detection. The security of these applications depends on the trustworthiness of the captured images, i.e., whether the images reflect reality correctly, since a distorted image may mislead the detector. In this paper, we uncover and investigate a new class of vulnerabilities that can be exploited to falsify the captured images using IEMI. The falsified images can potentially cause computer vision (CV) applications to make incorrect decisions, as illustrated in Fig. 1.

Figure 1: GlitchHiker attacks falsify the output images of an autonomous vehicle’s camera by injecting EMI signals, which can hide the cars to be detected, resulting in collisions.

* Chen Yan is the corresponding author.

Different from existing attacks against cameras [11, 16, 19, 27, 34, 37, 49, 50] that exploited image sensors as the attack surface, our work calls for attention to the “Image Signal Transmission” phase in a camera system, which involves transmitting the raw image signals generated by the image sensor to an image processor via a standard camera interface bus. The transmitted image signals will go through an image processing pipeline before becoming the final images or videos. Vulnerabilities of the image signal transmission have long been overlooked due to the seeming absence of an attack surface—external access to the image signals is disabled by camera design. Our work is motivated by the bizarre camera glitches, a phenomenon found in rare but benign scenarios on various types of cameras, including Raspberry Pi [20, 44], webcams [15, 30], and smartphones [1, 4]. As shown in Fig. 2, most of the camera glitches appeared as horizontal purple stripes overlapped across the entire image, significantly distorting the image content. Since no optical interference was present when camera glitches happened, electromagnetic interference (EMI) from the environment or other parts of the device is suspected to be one of the main culprits [7]. If true, we wonder if an attacker could falsify a camera’s output images by intentionally inducing similar glitches via IEMI. However, it is unclear whether such an attack is feasible or poses a genuine threat though this phenomenon has been known for almost a decade.

After studying a camera system’s workflow, we believe the camera interface bus used for image signal transmission is a significant target for EMI coupling. Our preliminary experiment demonstrates that a strong EMI can inject interference signals into the image transmission line despite the protocol’s anti-interference designs, causing random glitches similar to Fig. 2. To investigate the reason, we discover that the raw im-
age signals are designed to be transmitted by individual lines of the image sensor and are decoded according to the fixed color filter arrangement. Due to the alternating arrangement of color filters in consecutive lines, if a line missing in the transmission, e.g., caused by EMI, is not handled properly by the image processor, it can disrupt the color interpretation of the following lines during the image processing, thereby causing colorful ribbonings. In addition, we observe that if a large number of lines are missing in transmission, it will cause visible content stitching between consecutive frames.

Based on the discovered vulnerabilities of image signal transmission, we design GlitchHiker attack (GH attack in short) that can falsify a camera’s output images with EMI. Without wiring into the camera directly, GH attack can intentionally create camera glitches and manipulate glitch patterns’ position, width and number. We successfully launch the GH attack on 8 off-the-shelf camera systems in 5 categories in their original packages at a distance of up to 30 cm. Proof-of-concept experiments conducted on these cameras with 2 case studies involving 4 object detectors and 2 face detectors demonstrate that injecting one ribboning can effectively reduce the reliability and performance of object detection and face detection with a maximum success rate of 98.5% and 80.4%, respectively. We discuss real-world attack scenarios, build a miniaturized attack prototype for less than $40, and investigate the attack’s feasibility in a real-world scenario. Although our paper mainly studies untargeted attacks, we have performed additional preliminary investigations on targeted attacks that can hide, create and alter specific objects.

Understanding GH attack and analyzing the characteristics of camera glitches can help enhance future camera systems and expand the assumption on forms of image interference to prepare CV systems against a broader range of attack vectors. In summary, our contributions include:

- We discover a new class of camera vulnerabilities involving the previously overlooked image signal transmission phase in a camera system. We are the first to study and explain the camera glitch phenomenon.
- We design the GH attack, which can intentionally falsify a camera’s output images by adding controlled glitch patterns via IEMI injection. We validate the attack on 8 off-the-shelf cameras in 5 categories with 2 case studies involving 4 object detectors and 2 face detectors.
- We discuss real-world attack scenarios and perform preliminary investigations on the feasibility of targeted attacks that can hide, create or alter specific objects.
- We suggest hardware- and software-based countermeasures.

In summary, our contributions include:

- We design the GH attack, which can intentionally falsify a camera’s output images by adding controlled glitch patterns via IEMI injection. We validate the attack on 8 off-the-shelf cameras in 5 categories with 2 case studies involving 4 object detectors and 2 face detectors.
- We discuss real-world attack scenarios and perform preliminary investigations on the feasibility of targeted attacks that can hide, create or alter specific objects.
- We suggest hardware- and software-based countermeasures.

2 Camera System Preliminaries

In this section, we introduce camera systems and their known vulnerabilities. A camera system’s workflow includes three typical procedures: image capture, image signal transmission, and image processing. As shown in Fig. 3, an image sensor converts ambient light to digital signals of a pixel array, which are transmitted via a standard camera interface to an image signal processor (ISP) that produces the final image or video.

2.1 Image Capture

There are two main types of image sensors, the charge-coupled device (CCD) and the complementary metal-oxide-semiconductor (CMOS). CMOS sensors currently dominate consumer products due to their low cost and power consumption. As shown in Fig. 4(a), an image sensor consists of three parts: (1) an array of photodiodes that measures the light intensity and converts it to an electrical signal, (2) a color filter array (CFA) that only allows the light of specific wavelengths to reach the photodiodes, and (3) a microlens that increases the optical fill factor. This design enables the image sensor to measure different colors at various pixels. The Bayer filter mosaic [47], also known as BGGR, RGBG, GRBG, or RGGB, is the most popular pattern for color filter arrays. Inspired by the human visual system, a typical Bayer filter mosaic is arranged with red (R), green (G), and blue (B) color filters alternatively, with the number of green filters doubling others because human eyes are more sensitive to green light.

2.2 Image Signal Transmission

Several serial and parallel buses (a.k.a. the camera interface or CAMIF) have been proposed for public and proprietary use to connect an image sensor and an image processor. For example, the Digital Video Port (DVP), Low-voltage Differential Signaling (LVDS) [46], MIPI Camera Serial Interface 2 (MIPI CSI-2) [28], and High-speed Pixel Interface (HiSPI) [2]. With the rising demand for higher throughput and compatibility between hardware and software from different vendors, MIPI CSI-2 has become the de-facto standard for

Figure 2: Benign camera glitches in public reports.

Figure 3: A simplified camera system’s workflow, including image capture, image signal transmission, and image processing. Image capture and processing are typically implemented on separate hardware that transmits and receives image signals via standard camera interfaces, such as the MIPI CSI-2.
After receiving the raw image signals, the ISP or host processor performs image processing to recover viewable images that match human perception. Despite the manufacturer algorithm differences, a standard image processing pipeline typically includes demosaic, white balance, and post-processing.

Demosaic is a process of reconstructing a full-resolution color image from the sampled Bayer-encoded image signals. The sampled Bayer-encoded image is not viewable because each pixel only measures one color, and interpolation is required to estimate the full color of each pixel in the RGB color space using information from nearby pixels [35]. Standard algorithms include direct interpolation, nearest-neighbor interpolation, and bilinear interpolation.

White balance or color balance is a color correction mechanism that compensates for the unrealistic color biases caused by CFA and light sources. White balance can be achieved in various ways, and the gray world algorithm (GWA) [23] is utilized on most cameras.

Depending on the camera application, the ISP may perform further post-processing, such as noise reduction and image compression, to provide photos or videos in the required format and quality.

2.4 Known Camera System Vulnerabilities

Existing vulnerabilities of camera systems mostly reside on the sensors in the image capture phase, e.g., the saturation [34, 49], the fundamental architecture [18], and rolling shutter mechanism [19, 37, 50] of image sensors, the reflection of lenses [27], and acoustic susceptibility of the image sensor’s anti-shake head [16]. The vulnerability of image signal transmission has been rarely discovered and exploited yet. Oyama et al. [32, 33] proposed a fault injection attack on image transmission lines with the assumption of physically connecting to the MIPI CSI-2 lines. However, external access to the image sensor is disabled by the camera’s design in the real world. Our work parallels with [32, 33], and we can stealthily falsify a camera’s output image using IEMI without physically wiring into the image transmission lines.

3 Threat Model

3.1 Attacker’s Goal and Attack Scenarios

The attacker aims to reduce the reliability and performance of camera-based computer vision (CV) systems by stealthily falsifying a camera’s output images with IEMI. In particular, there may be two attack scenarios: (1) Mislead the object detection of autonomous driving. The attacker may fool autonomous vehicles by intentionally creating glitch patterns to hide existing objects or create fake objects in the image. Either case may cause erroneous vehicle behaviors and potentially lead to accidents. (2) Evade the face detection of a video surveillance system. Similarly, the attacker may avoid being detected if the captured image is falsified with IEMI.

We envision that the intentionally created glitch patterns may significantly disrupt or spoof CV systems, and we classify the attacks into three categories based on their outcomes: (1) Hiding Attack (HA): where an object/face in an original image disappears under the attack. (2) Creating Attack (CA): where a new object/face appears under the attack. (3) Altering Attack (AA): where an object/face in an original image is misdetected to another object/face under the attack. It is important to note that our attack is to hide, create or alter any objects/faces, i.e., untargeted attacks.

3.2 Attacker’s Capabilities

We make the following assumptions for the attacker to achieve the aforementioned attacks:

**Attack with EMI.** Considering the attacker’s goal and all possible options to trigger the camera glitches, we assume the attacker can only falsify the camera’s output image by intentionally generating EMI. That is, she cannot take apart or wire into the camera, physically change objects in the camera’s field of view, emit visible light and sound, or digitally hijack the camera’s image transmission to the CV system, e.g., using malware.

**Knowledge of the Target Camera.** The attacker knows the target camera’s model, and she may obtain a similar camera for assessment beforehand. For example, she may learn the design of the camera’s frame rate from public documents or by reverse engineering. However, during the attack, the target camera remains as a black box, and there is no regular access to the camera’s output images or inner signals.

**Proximity Access to the Target Camera.** We assume the attacker can physically get close to the target camera and
quickly place an attack device on/near the camera to prepare for the attack. This can happen when vehicles and surveillance cameras are temporarily unattended. We envision two types of potential real-world attack scenarios: contact attacks and contactless proximity attacks. Nonetheless, the attacker does not have the time to dismantle the camera and tamper with its hardware or software settings.

4 Feasibility and Vulnerability Investigation

In this section, we first investigate the feasibility of intentionally generating camera glitches and the reasons for glitch patterns. Then, we illustrate how the investigation facilitates us in identifying vulnerabilities of image signal transmission. Our investigations and experiments are conducted based on a standalone OmniVision OV5647 camera module attached to a Raspberry Pi 3B+ board through the standard MIPI CSI-2 camera interface.

4.1 Generating Camera Glitches with IEMI

Several online discussions suspected EMI as the cause of camera glitches in certain circumstances [4, 20]. These observations motivate us to investigate the feasibility of generating glitch patterns using IEMI. As a preliminary experiment, we utilize an electrostatic gun to generate high-intensity electric charge pulses at the image transmission line between the OmniVision OV5647 camera module and the Raspberry Pi 3B+ board. We report the results in Figs. 5(a) and 5(b). Also, we successfully inject similar glitch patterns in Fig. 5(c) by employing a signal generator, a power amplifier, and a magnetic probe to generating a 30 MHz sine wave. The glitch images we generated are similar to the benign cases in Fig. 2, showing purple horizontal ribboning and serious content stitching. However, these glitch patterns appeared sporadically and traversed the image.

4.2 What Goes Wrong with the Camera Systems?

After verifying the feasibility of generating camera glitches with IEMI, we are inspired to investigate the cause of glitches and what goes wrong with image signal transmission.

Investigate the Cause of Camera Glitches. Data buses are well-known to be susceptible to EMI disruption, which couples an electric potential into the wires and affects the voltages of the transmitted signals. We use a 30 cm sunny connector and conduct an experiment to demonstrate the cause of camera glitches by moving the magnetic probe to the three injection points indicated in Fig. 6(a). The other two injection points are shielded when the probe is in one of the three points. The result shows that the glitch patterns can be induced only when the probe is placed over the transmission line at the injection point 2, implying that the glitch is caused by interfering with the image signal transmission. After confirming that the image transmission phase is disrupted, we address the following two questions.

What Goes Wrong with the Transmission Lines? MIPI CSI-2 is designed with preparation for environmental interference, and the MIPI D-PHY [29] physical layer carries differential signals on several pairs of DATA+ and DATA− lines. This design is capable of rejecting common-mode interference in most cases. We wonder what goes wrong. During the injection, we monitor the DATA+ and DATA− lines of the image transmission with an oscilloscope. Fig. 6(b) shows the signals measured on the DATA+ and DATA− lines, as well as the calculated differential signal when a single line of the image is transmitted under EMI. Both data channels are affected by different levels of EMI interference, and there is still interference in the differential signal. In other words, EMI introduces differential-mode interference into the transmission lines, causing some image signal transmission errors. In addition, we use the commands “vcgencmd set_logging level=0xc00” and “sudo vcdbg log msg” to capture the firmware log of Raspberry Pi’s SoC during the injection. The result shows that there are numerous error messages in the log when the glitches appeared, indicating that various transmission errors occurred.

How Does the Transmission Protocol Respond to the Errors? We carefully review the source code (bcm2835-unicam.c and bcm2835-unicam.h) of the Raspberry Pi’s CSI driver and find a status register Unicam_STA defined to report transmission errors. Raspberry Pi implements the Checksum and ECC checking in hardware block, and errors are indicated via the Unicam_STA register. The Pi SoC discards the lines that encounter transmission errors1, and sets a status bit in the Unicam_STA register. This does not

---

1 This is confirmed by the Raspberry Pi engineers who write this driver.
increment the write pointer, so the next line received without multiple bit errors will be written to that point in the image buffer. We experimentally confirm this by analyzing the size of the captured raw data file. We use the command line “libcamera-raw -t 1000 –segment 2 -o test%03d.raw” to capture the raw frames with the resolution of 640*480 for 2 seconds. We receive a .raw file with a size of 384,000 bytes under normal circumstances. However, the size of a .raw file we receive during the attack is less than 384,000 bytes (for example, 380,928 bytes), indicating that several lines are missing.

4.3 Why Glitches Appear in Specific Patterns?

After knowing what goes wrong with the image signal transmission under EMI interference, the following question is why a few discarded horizontal lines can significantly alter the color and content of the other parts of the image. We first study the standard image processing pipeline on Raspberry Pi, and then we investigate the causes of image color and content changes, respectively.

4.3.1 The Normal Image Processing Pipeline

Raspberry Pi has two independent hardware blocks to handle the received image data packet: Unicam (CSI receiver) and ISP. The processing pipeline can be briefly summarized as follows. A buffer in Pi SoC’s SDRAM is allocated for the captured image, and the start/end address of the buffer and the line pitch are passed to the Unicam. Unicam waits for the FS interrupt to write all subsequent data into this buffer. When the FE interrupt is received, the buffer is signaled as “complete” and passed to the ISP. Then, the ISP process the defined image buffer (demosaic, white balance, etc.) with whatever data is there and output the final image. However, the missing lines disrupt this pipeline, changing the captured image’s color and content.

4.3.2 Investigating the Color Change

As shown in Fig. 4(b), the blue-green (B/G) and green-red (G/R) filters in the Bayer filter array are arranged in a line-alternating fashion. Half of the lines contain blue-green colors, and the other half contains green-red colors. The missing of one line may desynchronize the Bayer pattern and significantly affect the color cognition of subsequent lines. We illustrate this effect with two cases in Fig. 7.

Case 1: An image data packet is discarded. When line 2 in Fig. 4(b) is discarded, the demosaic algorithm does not know that a line is missing and works as usual. Thus, the Bayer order below line 2 swaps and becomes Fig. 7(a). The red and blue filters in Bayer are decoded as green filters, and the green filters are decoded as red or blue filters. Due to the architecture of the RGGB CFA, green values are usually higher than red and blue under normal circumstances. Therefore, after color swapping, the values of blue and red increase and the green values decrease in most cases, making the image to be purple or magenta after the process of demosaicing and white balance. In addition, discarding an odd number of lines can achieve the same outcome, but discarding an even number of lines cannot.

Case 2: Another image data packet is dropped after Case 1. As illustrated in Fig. 7(b), a second line (line 5) is discarded in the same manner, the Bayer order is swapped again, and the subsequent lines are returned to their previous state, resulting in a color stripe overlapping on the original image. In this paper, the phenomenon associated with Case 1&2 is referred to as intra-frame colorful ribboning.

![Figure 7: Two examples illustrate why intra-frame colorful ribbonings appear.](image)

Figure 7: Two examples illustrate why intra-frame colorful ribbonings appear. Discarding a line in (a) leads all subsequent green filters to be decoded as red and blue, and all red/blue filters to be decoded as green. In (b), removing another line can end the color discrepancy.

**Insight 1:** Discarding \((2n - 1)\) lines \((n \in \mathbb{N}^+)\) continuously can swap the Bayer order and disrupt the color interpretation of subsequent lines and discarding \((2n - 1)\) lines again ends this form of color discrepancy.

Depending on the RGB values of the source image, numerous colorful ribbonings may be induced. Fig. 8 depicts the transition of the color space resulting from the missing lines on a ColorChecker Classic. The discarding of lines converts an image’s color space from a standard color space to a purple-green color space. For example, the green region in Fig. 8(a) converts into purple in Fig. 8(b), while the red region becomes green. Moreover, after color space shifting, the fourth row of six grayscale colors, including white \((255, 255, 255)\) and black \((0, 0, 0)\), remains unchanged. This is because the RGB values of grayscale colors are identical in all three channels and this form of swapping has no effect on them.
5 Attack Design

In this section, we present GlitchHiker, the first attack that can intentionally induce controlled camera glitches via IEMI. The attacker’s goal is to inject disruption to falsify a camera’s output images to glitch patterns at a variety of positions, widths, and numbers. The attack workflow is summarized in Fig. 10. We first endeavor to find the appropriate injection frequency with the proper transmitting antenna to perform efficient glitch injection. Second, we synchronize the injection frequency with the proper transmitting antenna to perform efficient glitch injection. Then, we synchronize the injection with the designed signal transmission and design the injection signal to manipulate the position, width, and number of glitch patterns.

5.1 Efficient Glitch Injection

Though we have verified the possibility of generating camera glitches with an electrostatic gun in Section 4.1, these glitch patterns only appeared occasionally. The effectiveness of the glitch injection is dependent on the strength of the coupled signal in the victim camera system. To increase the efficiency of glitch injection, we first consider the generation and transmission of the EM signal.

EM Signal Generation. The coupling efficiency between a victim circuit and an EM signal is determined by the frequency and amplitude of the EM signal. (1) Signal frequency \( f \). We can increase the strength of the coupled signal by determining the frequency at which the victim circuit is most susceptible. Since it is difficult to examine the victim circuit’s resonant frequency, we conduct a frequency sweeping test to determine the effective frequency range. (2) Signal amplitude
5.2 Controlled Glitch Injection

After improving the injection efficiency, the following task is to make the random glitches controlled. The attacker can accomplish this by (1) synchronizing the injection with image signal transmission and (2) designing the controlled injection signal to manipulate the position, width, and number of the glitch patterns.

5.2.1 Synchronization with Image Signal Transmission

As shown in Fig. 12, we synchronize the injection with image signal transmission by EM leakage detection, signal processing, and injection signal transmission.

**EM Leakage Detection.** The voltage of the victim camera’s data lines changes from “high” to “low” or reverses during image signal transmission, creating a sudden current in the line. This sudden current creates a magnetic field that an attacker can use to eavesdrop the timing of the image signal transmission for synchronization. We utilize an oscilloscope, a magnetic probe, and a low noise amplifier to monitor the EM leakage and Fig. 13(a) illustrates the relationship between the EM leakage and the original signal. The result indicates that synchronizing injection timing with the image signal transmission is feasible.

**Signal Processing and Injection Transmission.** Our strategy of synchronization with image signal transmission is depicted in Fig. 13(b). After detecting the frame start interrupt of $Frame_i$, the arbitrary signal generator is triggered to output the injection signal into $Frame_{i+1}$ after a latency of $\Delta t$. The signal generator’s latency is less than 100 ns, which is substantially shorter than the period of line transmission and can be considered constant. Furthermore, we can calculate the frame period according to the camera’s frame rate to estimate the arrival of $Frame_{i+1}$ and adjust the timing of the injection signal. Thus, we can execute steady injections across continuous video frames.
5.2.2 Manipulation of Glitch Patterns

We design the controlled injection signal to manipulate the frame’s position, width, and number of ribbonings. Fig. 14 shows the schematic diagram of the signal we designed.

Controlled Injection Signal Designing. The controlled injection signal has five configurable parameters: signal frequency \( f \), signal amplitude \( A \), pulse width \( W \), signal delay \( \Delta t \), and signal interval \( \Delta T \). A controlled injection signal contains \( W \) width of the pulse signal, and \( \Delta T \) denotes the time interval between two consecutive injection signals. \( f, A, \) and \( \Delta t \) represent the sine wave’s frequency, amplitude, and delay time, respectively. We can perform various glitch injections by varying the values of these five parameters: (1) We should choose the adequate combination of \( f \) and \( A \) determined in Section 5.1 to realize the efficient glitch injection. (2) We can change \( \Delta t \) to do synchronization and manipulate the position of the ribboning. (3) When \( W \) is increased, more lines are discarded, resulting in more severe visible inter-frame stitching. We set \( W \) to be 100 us based on empirical value to achieve a high injection success rate while avoiding visible stitching. In addition, we evaluate the effect of the value of \( W \) on stitching in Appendix F. (4) We can change the value of \( \Delta T \) to adjust the intervals between adjacent ribbonings and inject ribbonings across frames according to the camera’s frame rate. \( \Delta T \) can be a constant or a variable, and we show examples of ribbonings with equal and unequal intervals in Appendix B. The condition of ribbonings with unequal intervals is complex, and we set \( \Delta T \) to be a constant to inject ribbonings with equal intervals to demonstrate the ability of \( \text{GH} \) attack.

Manipulation of the Position, Width, and Number of Purple Ribbonings. (1) Position. After synchronization with image signal transmission, the start position of the ribboning can be represented as \( \frac{\Delta t}{F} \), where \( F \) is the frame rate of the camera system. We can adjust the signal delay value \( \Delta t \) to change the position of the glitch patterns. Fig. 15(a) shows that the ribboning gradually move downward (from left to right) with the increase of the value of \( \Delta t \). (2) Width. Given the start position of the glitch pattern, we can manipulate a ribboning’s width by adjusting the value of signal interval \( \Delta T \) according to Insight 2. Fig. 15(b) shows that the ribboning gets wider with the increase of the signal interval \( \Delta T \). (3) Number. We can adjust the settings of signal delay \( \Delta t \) and signal interval \( \Delta T \) to adjust the value of \( N \) to manipulate the number of purple ribbonings. If \( \Delta t = 0 \), we can set \( \Delta T = \frac{1}{F} \cdot \frac{1}{P} \) to inject \( N \) ribbonings, and the smaller the \( \Delta T \), the denser the color stripes. Fig. 15(c) shows examples of 1, 2, 3 purple ribbonings overlapping on the image with different \( \Delta T \) settings of 16.66 ms, 6.66 ms, and 4.76 ms when the frame rate is 30 fps. More examples can be found in Appendix B.

6 Evaluation

We perform \( \text{GH} \) attack on 8 camera systems to validate the effectiveness of the attack. Then, we evaluate the untargeted attacks on these cameras with 2 case studies involving 4 object detectors and 2 face detectors. Next, we evaluate the attack’s performance regarding the ribboning number and attack distance. Finally, we discuss real-world scenarios, build a miniaturized attack prototype, and briefly investigate the attack’s feasibility in a real-world scenario.

6.1 Experiment Setup

The experimental setup is shown in Fig. 16. The attack devices used in a laboratory setting include a Keysight N5712b vector signal generator for EMI signal generation, a Mini-Circuits ZHL-100W-GAN+ power amplifier or a Mini-Circuits ZHL-50W-63+ power amplifier for EMI signal amplifying, a convenient magnetic field probe (with a radius of 45mm and operating frequencies ranging from 9 kHz to 3 GHz) for EMI signal generation transmission. A RIGOL MSO8104 digital oscilloscope, a near-field probe, and a low noise amplifier (LNA) are used to monitor the EM leakage to synchronize with image signal transmission. Thus, if not otherwise specified, we maintain the experiment setup like this.
6.2 Attack on Eight Camera Systems

We evaluate our GH attack on eight off-the-shelf camera systems in five categories, including five physically sealed consumer camera systems and three latest IoT Kits with MIPI cameras. The camera systems we tested are shown in Fig. 23, and their specs and attack results are reported in Table 1. Attaching the attack antenna to the camera shell in preparation, we run a frequency sweeping test on these cameras from 20 MHz to 1 GHz with a 10 MHz step and -5 dBm amplitude. We attach the attack antenna next to the camera’s shell in advance and conduct a frequency sweeping test on these cameras from 20 MHz to 1 GHz with a step of 10 MHz at an amplitude of -5 dBm. The results show that all eight cameras are subject to our GH attack whether the camera system is physically sealed. We can induce both intra-frame colorful ribboning and inter-frame content stitching glitch patterns. During the frequency sweep test, we can achieve successful attacks at multiple frequencies, and the minimum effective injection frequency varies among camera systems due to their different circuits. Under the guidance of Section 5.2.2, we can manipulate the glitch patterns at different position, width and number by modifying the value of $\Delta T$ and $\Delta T'$ according to each camera’s frame rate. For example, to induce one ribboning into a Blackview Dacshcam and a Xiaomi Dafang IP camera, we should set the value of signal interval $\Delta T'$ to be 16.6 ms and 12.5 ms, respectively.

6.3 Case Studies

We conduct the GlitchHiker attack as a proof of concept to demonstrate the vulnerability’s security impact. We place the victim camera systems in front of a monitor displaying images from a public dataset.

**Attack Strategy.** The attacker’s goal is to reduce the reliability of camera-based computer vision systems. Unlike the works on generating adversarial examples, we do not focus on targeted adversarial attacks, which require obtaining the real-time camera images for optimization and synchronizing to precisely controlled injection, and are too overwhelming to fit our threat model. During the attack, we use the same set of signal frequency, signal amplitude, and pulse width (1 GHz, -5 dBm, and 100 us) and customize the value of signal interval $\Delta T'$ based on each camera’s frame rate. To increase the variety of injected glitch patterns, for each experiment, our attack strategy is to inject a ribboning with various positions inside the images and widths to cover more cases.

**Evaluation Metric.** We classify the fine-grained attack outcomes into three categories described in Section 3 based on the intersection over union (IoU). The details of IoU can be found in Appendix D. We use attack success rate (SR) as the evaluation metric, which is the ratio between the number of successful attacks (at least one object/face is hidden, created, or altered in the detector’s output) over the total number of conducted attacks.

6.3.1 Case Study I: Object Detection

We display the 100 images selected from the BDD100K dataset [52] and send the images captured by 3 cameras (Raspberry Pi 3B+, 360 M320 Dashcam, and Blackview Dashcam) to 4 academic object detectors, including 3 one-stage object detectors (YOLO v3/v4/v5 [3,5,36]) and a two-stage object detector (Faster R-CNN [6]).

The attack’s overall SRs are reported in Table 2, and we summarize the results regarding CV systems, attack outcomes, and camera systems. 1) **CV systems:** our attack achieves an average attack SR of 82.7%, 85.7%, 93.4%, and 98.5% against YOLO v3/v4/v5 and Faster R-CNN, where Faster R-CNN is the most fragile of the four models. 2) **Attack outcomes:** our attack achieves an average attack SR of 77.2%, 60.0%, and 27.8% for HA, CA, and AA. We can see that it is easier to achieve HA and CA than AA for these detectors, which may be because AA needs to make more fine-grained changes to the image. 3) **Camera systems:** the attack’s average SRs against Raspberry Pi 3B+, 360 M320 Dashcam, and Blackview Dashcam are 90.0%, 89.7%, and 90.6%. The performance of these three camera systems appears to be very similar, although the Blackview Dashcam is slightly more vulnerable. HA may mislead autonomous vehicles to unintended operations. For example, the attacker can hide all the objects in the original image in Fig. 17(a), resulting in severe consequences such as hitting the car. CA may create a non-existent object that can lead to malicious driving behaviors. As shown in Fig. 17(b), the attacker can create a person, resulting in emergency brakes or detours. AA may cause severe traffic accidents as HA and CA do; for example, the cars in the original image may be altered to a train in Fig. 17(c).

6.3.2 Case Study II: Face Detection

We evaluate the attacks on Facenet [38] and RetinaFace [10] using ASUS Tinker Board 2, Xiaomi Dafang IP Camera, and Google Pixel One. The 100 images are selected from the WIDER dataset [51] and real-world surveillance videos. Two attack outcomes are detected: fewer people (HA) or more people (CA). Altering attack (AA) is not available.
We report the attack’s overall success rates on face detection in Table 3 and summarize the results regarding CV systems, attack outcomes, and camera systems. 1) **CV systems:** our attack achieved an average attack SR of 80.4% and 76.3% against Facenet and RetinaFace, which appear to be similar. 2) **Attack outcomes:** the average attack SRs of HA and CA are 63.4% and 15.0%, indicating that it is comparatively easier to hide a person from being detected than to create a new person. 3) **Camera systems:** the attack’s average SRs against ASUS Tinker Board 2, Xiaomi Dafang IP Camera, and Google Pixel One are 74.6%, 83.9%, and 76.8%. The Xiaomi Dafang IP camera is the most fragile of the three cameras. For example, 6 persons are hidden in Fig. 17(d) and a non-existing person is created in Fig. 17(e) after GH attack. **Summary.** The following observations are made as a conclusion: (1) End-to-end GlitchHiker attacks can significantly disrupt object detection and face detection even if only one ribboning is injected into the captured image. (2) HA and CA are easier than AA, and AA requires a more fine-grained attack design. **6.4 Impact Quantification.** We quantify the attack’s capability under the impact of the ribboning number and attack distance. The experiment is carried out on a standalone OmniVision OV5647 camera model attached to a Raspberry Pi 3B+.

**Impact of Ribboning Number.** We vary the settings of the controlled injection signal for each image to inject a different number of ribbonings ranging from 1 to 10 at equal intervals. The result of attack SRs on face detection are shown in Fig. 18(a), and we can see that the attack SR grows with the increment of ribbonings. For example, we achieve an attack SR of 93.1% in a ribboning, while 97.3% in ten ribbonings, indicating that increasing the number of ribbonings results in more modifications to the image content and a greater likelihood of modifying the IoU in the images. There is a trade-off between the attack overhead and ribboning number.

**Impact of Attack Distance.** We evaluate the relationship between the power level and attack distance and calculate the injection success rate as the number of images that are successfully injected over the total number of images collected.
The objects are hidden

The person is created

The car is altered into the train

6 of 8 persons are hidden

A person is created

Figure 17: Illustrations of real-world attacks on object detection (a-c) and face detection (d-e).

Figure 18: Performance of the attack in terms of ribboning number and attack distance.

We envision that there could be two types of potential real-world attack scenarios: (1) **Contact attacks**: the attacker can physically contact the target camera system and place the attack devices on/near the camera in advance to the attack opportunistic, e.g., during car wash or car maintenance; (2) **Contactless proximity attacks**: the attacker may remotely control the device and launch the attack. The attack device can also benefit from a low cost and minimal power consumption. In the second contactless proximity attack scenario, the attack devices need to support attacking from a reasonable distance, which requires the transmitted EMI signal to be powerful and directional so that it can attack successfully and avoid interfering with other electronic devices around the camera system.

**Methods to Overcome the Challenges.** We discuss our efforts in overcoming the above challenges of the two attack scenarios. For contact attacks, we build a miniaturized attack prototype shown in Fig. 19(a), which is less than 15 cm long and 6 cm wide and costs less than $40. The prototype consists of an antenna and an electromagnetic pulse (EMP) generator made by a microcontroller (Seeeduino XIAO) and an EMP generation circuit. The microcontroller is responsible for controlling the EMP generator to generate pulses of width \( \Delta W \) at the set interval \( \Delta T \) and can be controlled remotely via Bluetooth. The EMP generation circuit consists of an N-channel MOSFET, a triode, and an amplifier. As shown in Fig. 19(a), the attacker can stick the battery-powered prototype next to the target camera when it is unattended, and launch the attack remotely via the Bluetooth connection to the prototype. For contactless proximity attacks, we have managed to boost the attack distance to more than 30 cm using a basic directional antenna. To further increase the attack distance, we can implement high-performance equipment (e.g., a high-power EM amplifier, a high-directional antenna, etc.) or inject at the resonant frequency of the victim device, which will achieve the maximum injection efficiency.

**Experiment in a Real-world Scenario.** To investigate the attack’s feasibility in a real-world scenario, we test the contact attack on a dashboard camera mounted in a vehicle as a proof-of-concept experiment. We envision that attackers may quickly enter the target vehicle during a car wash or car maintenance and attach the attack prototype to the back of the camera, as shown in Fig. 19(b). Under this setup, we perform GlitchHiker attack on a Blackview dashcam and send the captured images to Faster R-CNN. The experimental results demonstrate successful end-to-end attacks that achieve all of the hiding (HA), creating (CA), and altering (AA) in this scenario. In Fig. 19(c), the cars in both images are hidden.
Figure 19: Illustrations of the attack prototype and the attack results on a Blackview Dashcam in a vehicle: (a) the self-built portable equipment for GH attack, (b) implementation for experimental setup, (c-e) representations of three attack outcomes. after the attack. In Fig. 19(d), a person is created in the upper image, and a train is created in the lower image. In Fig. 19(e), a car is altered into a truck in the upper image and a boat in the lower image. We envision that the above attack outcomes may mislead an autonomous vehicle and cause severe accidents, e.g., crashing into the cars hidden in Fig. 19(c).

7 The Feasibility of Targeted Attacks

In this section, we briefly investigate the feasibility of targeted attacks that can hide, create, or alter specific objects following similar assumptions and procedures of prior works [16, 50].

Stage 1: Emulating Glitch Patterns. We develop a GH attack emulation model that takes a raw image as input and outputs an image with simulated glitch patterns. There are two steps to simulate the glitch patterns: image processing modeling and IEMI modeling. We use the high-quality linear interpolation demosaicing method proposed in [26] to transform the CMOS sensor’s raw images into .jpg images and further correct the image based on the gray world algorithm (GWA) [23]. We leave the mathematical details to Appendix E. Under the guidance of Section 4.3, we can remove lines to induce intra-frame colorful ribbonings and add lines from the preceding or subsequent frame to simulate inter-frame content stitching.

Stage 2: Searching Attack Parameters. Using various combinations of injection signal parameters, we can simulate glitch patterns on prerecorded images of the target object based on the glitch pattern emulation modeling. The optimization parameters include the start position $S_i$ and width $W_i$ of the $i$-th glitch pattern, the total number of glitch patterns $N$, and the number of discarding lines $(2n - 1)$ respectively. We seek the most effective combinations of attack parameters using a Bayesian optimization method.

Stage 3: Generating EMI Injection Signals. We map the attack parameters from our simulation to adjustable injection parameters of the EMI injection signals, including: (1) Signal Delay $\Delta T$. The signal delay determines the start position $S_i$ of the ribboning in the model. (2) Signal interval $\Delta T$. We can change the ribboning’s width $W_i$ by modifying the value of signal Interval $\Delta T$, and the maximum number of glitch patterns $N$ can be expressed as $N = \left\lceil \frac{T_e - T_s - 2\Delta T}{T_e - T_s} \right\rceil$, where $T_s$ and $T_e$ are the start and end time of a frame. (3) Pulse Width $W$. We can increase the pulse width to discard more lines, and the relationship between the pulse width and the number of discarding lines is discussed in Appendix F.

Experiment of Targeted Attacks. Fig. 20 presents a comparison between the original, simulated attack and the real GlitchHiker attack images. Each group of images shows similar glitch patterns visually and has the same detection results from Fast-RCNN. For example, the truck in Fig. 20(a) is targeted to be hidden. A non-existing person is detected in Fig. 20(b) mistakenly. The traffic light in Fig. 20(c) is incorrectly detected as a stop sign. The results suggest that targeted attacks are feasible in an ideal experimental setup.

Practical Challenges of Targeted Attacks. Though the simulated images may look similar to those created in real attacks, there is sometimes a color difference between the real attack image and the simulated image, which is caused by the dynamically changing camera parameters such as ISO.
and exposure time. We found that the detector is sensitive to such color differences and may lead to unexpected attack results. In addition, the attacker will face several significant challenges for targeted attacks in practice: (1) obtaining the camera system’s output images, (2) real-time generation of adversarial examples according to the output images, and (3) synchronizing with the camera’s image signal transmission for precise control of the injected glitch patterns. We leave the exploration of targeted attacks to future work.

8 Discussion

Explore the Inter-frame Content Stitching. Due to space limits, we mainly focus on the intra-frame colorful ribboning and do not discuss the inter-frame content stitching in detail. We conduct additional experiments to investigate the relationship between the pulse width and the inter-frame content stitching patterns. We also explore the attacks exploiting inter-frame content stitching in Appendix F.

Countermeasures. To alleviate the vulnerabilities, we propose hardware- and software-based countermeasures. (1) The image transmission lines should have an electromagnetic shield structure, which is the first and most effective step in preventing a malicious signal from coupling to the image transmission lines. Unfortunately, the electromagnetic shielding in these off-the-shelf camera systems is not as effective as anticipated. (2) We suggest that the camera system manufacturers utilizing the MIPI CSI-2 protocol should strictly follow MIPI Alliance Camera Working Groups’ recommended approach for handling error conditions. We believe developing error handling mechanisms can deal with packet loss and prevent GH attacks. (3) The CV models can be protected from GH attacks by detecting these glitch patterns. We propose and evaluate an attack detection method by calculating the color difference between images in Appendix G.

Limitations and Future Work. The frequency step of the frequency sweeping test is so large that it is difficult to determine the resonant frequency at which injection efficiency is at its highest. We can use fine-grained frequency steps in the effective frequency range to identify the resonant frequency to improve attack performance. Furthermore, a more effective targeted attack requires further investigations and improvements. We designate these issues as our future work.

9 Related work

In the following, we summarize the existing attacks utilizing EMI and real-world adversarial attacks on CV systems.

EMI Attacks. EMI is known to jeopardize the integrity and reliability of analog sensor outputs, communication signal transmissions, and actuator behaviors. (1) Analog sensor’s outputs. Several existing IEMI attacks target at sensors, such as microphones [21, 48], implantable cardiac devices [21], magnetic speed sensors [42], smartphones [17, 48], light sensors [39], temperature sensors [43], CCD sensors [18] and touchscreens [40, 45]. (2) Communication signal transmissions. Selvaraj et al. [39] presented the IEMI attack that an attacker may induce bit flips in serial communications, and Ogura et al. [31] falsified a CAN frame by EM transient pulse injection to the CAN bus. (3) Actuator behaviors. Dayanikli et al. [8, 9] demonstrated modulated IEMI attacks to compromise the power switches and the servo motors. Our GH attack focuses on the image signal transmission phase and can actively induce controlled glitch images of a camera.

Real-world Adversarial Attacks on Computer Vision Systems. Existing real-world attacks can be categorized primarily as those that interfere with the camera sensor’s operation and purposely designed perturbations on the object in a special shape or color. Works in [11, 34, 49] demonstrated that strong lights or lasers could blind a camera or cause a rolling color effect, and works in [13, 16, 19, 24, 27, 37, 50, 53] showed the ability to bypass and misleading attacks against CV systems by illuminating the camera or object with modified infrared, LED, laser beams and acoustic signals. Oyama et al. [32, 33] proposed a fault injection attack by physically wiring into the camera’s MIPI lines. Works in [12, 14, 22, 25] demonstrated the existence of physical adversarial examples by printing hostile perturbations as a “adversarial patch”. Previous research has primarily focused on the object or the image capturing phase, whereas our GH attack emphasizes the security of image signal transmission. Our work highlights this new vulnerability, sheds light on its fundamental causes, demonstrates its severity impact and offers countermeasures.

10 Conclusion

In this paper, we uncover a new class of vulnerabilities in image signal transmission phase and explain the underlying principles of camera glitches for the first time. We design GlitchHiker, a unique attack that can actively falsify a camera’s output images through EMI. We demonstrate how the attack can disrupt off-the-shelf camera systems and deceive CV systems, discuss real-world attack scenarios and investigate the feasibility of targeted attacks. We propose both hardware- and software-based countermeasures to alleviate the vulnerability’s threat.

Acknowledgments

We appreciate the anonymous reviewers’ valuable comments and the Raspberry Pi engineers’ patience responses. This work is supported by China NSFC Grant 62222114, 62201503, 61925109, and 62071428.
References


Appendix

A Responsible Disclosure

We have reported our findings, including the vulnerable devices, a thorough description of the vulnerability, and proof of concept to the MIPI Alliance and the camera system manufacturers. We presented the practical steps required to reproduce a simplified attack version. In addition, we provided our hardware- and software-based countermeasures.

B Glitch Patterns

Ribbons with both equal and unequal intervals. When $\Delta T$ is a constant, the ribbonings are evenly spaced throughout the image in Fig. 21(a), whereas when $\Delta T$ is a variable, the stripes overlap at unequal intervals in Fig. 21(b).

(a) Injection with equal interval (b) Injection with unequal interval

Figure 21: Colorful ribbonings injected with (a) equal intervals when $\Delta T$ is a constant and (b) unequal intervals when $\Delta T$ is a variable on an OmniVision OV5647 camera.

Ribbons at Different Number. We can modify the value of $\Delta T$ to induce different numbers of purple ribbonings overlapping on the same image. Fig. 22 shows the examples, we configure $\Delta T$ as 16.6 ms, 8.31 ms, 5.56 ms, 4.15 ms, 3.32 ms, 2.77 ms, 2.38 ms, 2.09 ms, and 1.86 ms to induce purple ribbonings range from one to nine separately. The frequency, amplitude, cycles and delay of the signal are configured as 100 MHz, 5 Vpp, 3000 and 0, respectively.

C Evaluation of GH attack.

The camera systems we evaluate are shown in Fig. 23, including 5 physically sealed consumer camera systems and 3 latest IoT Kits in their original packages. Our experiments show that EMI attacks can successfully inject glitch patterns, including intra-frame colorful ribboning and inter-frame content stitching, into all 8 off-the-shelf camera systems. Note
that all camera systems except the 3 IoT Kits are physically sealed, and we do not make any modifications to these devices. We have learned their sensor type, maximum resolution, and frame rate from public documents.

![Image](57x621 to 128x661)

**Figure 23:** The 8 camera systems tested in the evaluation.

## D Explanations of IoU.

IoU is commonly used to evaluate a box’s prediction accuracy compared to an original or ground truth box, which is a measure of the area of overlap between two bounding boxes, as a proportion of area of overlap, area of union. If IoU \( \left( b_1^{(l_i)}, b_2^{(l_i)} \right) > 0.5 \), where \( l_i \) is the label of the bounding box \( b_i \), we suppose \( b_1 \) and \( b_2 \) are at the same position. We use \( b_0^{(l_o)} \) and \( b_0^{(l_a)} \) to represent the box before and after \( \text{GH} \) attack. We measure fine-grained misleading attacks in three categories: (1) **Hiding Attack:** there is a bounding box \( b_0^{(l_o)} \) in an original image, but the box at the same position disappears after the \( \text{GH} \) attack. (2) **Creating Attack:** there isn’t a bounding box \( b_0^{(l_a)} \) in an original image, but the box \( b_0^{(l_a)} \) appears after the \( \text{GH} \) attack. (3) **Altering Attack:** there is a bounding box \( b_0^{(l_a)} \) in an original image and a bounding box \( b_0^{(l_o)} \) at the same position in a disputed image after the \( \text{GH} \) attack, where \( l_o \neq l_a \).

We can summarize from the evaluation results of \( \text{GH} \) attack on face recognition as followings: (1) \( \text{GH} \) attack primarily affects face detection process when compared to face recognition, and our attack can hide a person from detection or create a fake face in the non-face area, which is then recognized as an enrolled person. (2) To improve the attack SR, the attacker should inject ribbonings into the area around a person’s eyes, nose, and mouth.

## E GlitchHiker Attack Patterns Emulation

To estimate the red value \( \hat{R}(i, j) \) or blue value \( \hat{B}(i, j) \) at a green pixel in the position \((i, j)\), we can express as \( \hat{X}(i, j) = X(i, j) + \alpha \Delta G(i, j) \), where \( X(i, j) = \frac{1}{k} \sum X(i + m, j + n) \), \( X \in \{ R, B \} \), \((i + m, j + n) \) is the position of neighboring red/blue pixels in a cross pattern, and \( k \) is the number of the red/blue pixels. \( \Delta G(i, j) \) is the gradient of \( G \) at that location and \( \alpha \) is a gain parameter to control the applied correction. The gradient of the pixel at location \((i, j)\) is \( \Delta X(i, j) = X(i, j) - \sum X(i + m, j + n), X \in \{ R, G, B \} \). After demosaicing, the red, green and blue value of a pixel in the position \((i, j)\) will be further corrected by the white-balance diagonal matrix. So the green values of pixels in the image can be expressed as \( G'(i, j) = \hat{G}(i, j) \), the red and blue values are \( R'(i, j) = \hat{R}(i, j) \times (K_r/K_g), B'(i, j) = \hat{B}(i, j) \times (K_g/K_b) \) respectively.

## F Inter-frame Stitching

We investigate the relationship between the attack signal’s pulse width \( W \) and the inter-frame content stitching pattern. As shown in Fig. 24, the inter-frame content stitching pattern becomes more prominent as the pulse width \( W \) increases (Figs. 24(a) to 24(c)). As shown in Figs. 24(b) and 24(d) to 24(f), when the pulse width \( W \) is constant, a different start position of the glitch pattern will result in different parts of the object being dropped.

![Image](321x190 to 391x229)

**Figure 24:** Illustration of inter-frame content stitching images with varied attack parameters.

We also investigated methods that use inter-frame content stitching patterns to attack the face detection system, with some representative examples shown in Fig. 25. Inter-frame content stitching patterns can cause persons in an image to both disappear (Figs. 25(a) to 25(c)) and reappear (Figs. 25(d) to 25(f)).
Figure 25: Illustration of a face detection attack that exploits inter-frame content stitching.

G Detecting the Images Under the Attack

We use the method in [41] to transform the image from RGB to CIE Lab color space to compute the color difference between the two images. We select six groups of images for color difference calculation, with each group containing 30 normal images and 30 images under our GlitchHiker attack. The results are shown in Fig. 26(a). The vertical red dash line in the center of Fig. 26(a) separates the normal images from the attacked images, and it is evident that the color difference value of the normal image in Fig. 26(b) is only affected by the environment and the camera. While the color difference value of the attacked image increases or falls dramatically and changes with the injected glitches in Fig. 26(c). For example, the average value of color difference for normal images in the fourth group is 1.12, but the maximum value under the attack might reach 3.97.

Figure 26: Results of the color difference between images.