ImageC2Gen: Customizing GenAI models to Conceal Commands in Images for Command and Control (C2) Attacks

Qian Feng [*1], Chris Navarrete [†1], Yanhui Jia [‡1], Qi Deng [§1], Durgesh Sangvikar [¶1], Iris Dai [‖1], Haozhe Zhang [**1], Mickey Zhang [††1], and Shengming Xu [‡‡1]

[1]Palo Alto Networks

March 24, 2025

Abstract

Command and Control (C2) attacks involve establishing an encrypted connection between victim machines and C2 servers. Utilizing Image-based C2 makes it more challenging for the network security and forensic analysis, even when firewalls have decryption capabilities enabled. This is because C2 images can be designed to resemble genuine pictures, causing the traffic generated by C2 requests to appear as normal image requests. Steganography techniques have been employed to embed C2 instructions within images, but such methods typically require a parser for decoding the commands. Antivirus software can easily detect this parser through code analysis, as the decoding logic is hard-coded.

In this paper, we propose to train a generative AI (GenAI) model to encode and decode C2 commands into or from arbitrary images. Instead of using a hard-coded parser, we train genAI ML models with arbitrary images. The output is an image with C2 commands hiddeing in it, and a decoder model which is used to decode the C2 commands from the image. This approach makes it more challenging for traditional anti-virus detection techniques. The malicious model used as a C2 parser is a legitimate model, and it is not code but a model file with weights. Therefore, traditional code analysis would be ineffective for detection. Additionally, signature-based detection is also difficult since the hash value of the model can be easily updated by the model retraining.

[*]qfeng@paloaltonetworks.com
[†]cnavarrete@paloaltonetworks.com
[‡]yjia@paloaltonetworks.com
[§]qdeng@paloaltonetworks.com
[¶]dsangvikar@paloaltonetworks.com
[‖]ydai@paloaltonetworks.com
[**]haozhang@paloaltonetworks.com
[††]mzhang@paloaltonetworks.com
[‡‡]sxu1@paloaltonetworks.com

1. Introduction

Generative AI is a specialized field of artificial intelligence that concentrates on generating novel and unique content. Unlike conventional AI models, which merely recognize patterns or make predictions, generative AI can produce entirely new content. It has been used for various applications, including text, image, and video generation. Numerous researchers concentrate on utilizing GenAI models to improve our lives or identifying weaknesses in GenAI models. In this paper, we investigate the potential application of GenAI for malicious purposes. Specifically, we aim to explore the feasibility of training a malicious generative AI model to serve attacks, specifically for C2 attacks.

Command and Control (C2) attacks function by establishing an encrypted connection between victim machines and C2 servers. If the firewall enables the decryption feature, security researchers still have the capability to extract features from the C2 traffic for detection. Image based C2 makes this analysis harder even if the firewall enables the decryption feature. This because the network traffic itself is legitimate, and all malicious contents are hidden in a actual picture. Identifying malicious code buried behind a actual picture is quite challenging and it is still an open research question.

Image based C2 could not be captured by the network detection, but the C2 decoding logic in its C2 stager still has some patterns can be used for detection at the endpoint. This is because that many of the existing Steganography techniques used by attackers are rule-based. They embedded the payload in the least significant bytes of an existing image so that it looks like an actual picture. It could be hard to detect an embedded image, but it would not be that hard to detect the stager used to decode and extract the malicious payload from the image. The decoding logic is hard coded in code, so it can be easily detected via antivirus software by using

code analysis.

In this paper, we propose training Generative AI (GenAI) models to encode and decode C2 commands into or from arbitrary images. Instead of using a hard-coded parser, we utilize ML models to encode and decode C2 commands. This approach makes it more challenging for traditional anti-virus detection techniques to identify malicious models via code analysis. The model-based C2 parser is a legitimate model, and it is not code but a model file with weights. Therefore, traditional code analysis would be ineffective for detection. Additionally, signature-based detection is also difficult since the hash value of the model could be changed due to the model-based C2 parser being easy to retrain. We investigate the feasibility of utilizing AI models to generate and decode C2 images, and we showcase its effectiveness by providing a live demo.

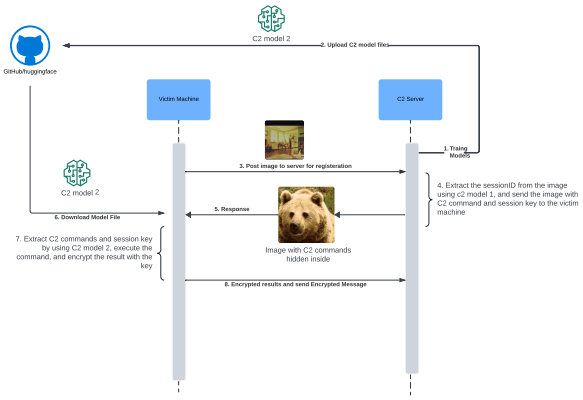

Figure 1. The AI enhanced C2 Framework work flow.

2. Overview

In this paper, we provide a novel AI enhanced image based C2 framework. Like the traditional C2 framework, the AI enhanced image based C2 framework will also set up listeners and generate stagers. The different part is that the framework will have the imageGen component. This imageGen component will be used to generate an actual image containing arbitrary C2 session keys or commands, and a corresponding ML model which is used to extract C2 essentials from the image. For each stager, the C2 server will generate new images and models for it. Then it will publish C2 models into huggingface or github. At the staging phase, the stager will retrieve the corresponding image from the C2 server, and download the C2 model from model repositories. The stager will use the C2 model to extract C2 keys and commands for execution. Finally, it will encrypt the execution results with the C2 key and send the result back to the C2 server. The figure 1 illustrates the whole C2 communication process.

2.1. The AI enhanced C2 Server

The AI-enhanced C2 server will incorporate an AI component, ImageGen, within its framework, which is responsible for generating C2 images for C2 communication. When attackers create a stager, the server initiates ImageGen to produce the corresponding C2 images and their associated C2 models. For each stager, the server maintains a staging image and its corresponding staging model, used for the handshake between the server and the stager. The server retains the staging model locally, and incorporates the staging image into the stager. Additionally, the server holds a session image and a variety of C2 images with different C2 commands along with a C2 model for C2 communication. After creating a stager, the C2 server publishes the C2 model for the stager to public repositories. When the C2 server receives a image, it will first extract the stager id from it using the staging model. If it can be decryped successfully, it will register the stager with the stager id and send the corresponding session image of this stager to the victim machine and wait for the response. If the stager responses 'ok', then the C2 server will send C2 images to the victim machine and wait the results. Finally, the results encrypted with the session key will be returned. The C2 server will decrypt the response with the session key and get the result.

2.2. The AI enhanced C2 Stager

The AI-enhanced C2 stager performs three main tasks. First, it establishes a connection with the C2 server for a "handshake." This handshake involves registering the stager on the server-side and requesting a session key. The stager sends the staging image embedded within itself back to the server, including a unique stager ID, and awaits the server's response. The server extracts the stager ID using the stager model stored on the server-side.

The server's response to the victim machine is a session image. The C2 stager downloads the stager model from the public repository and extracts the session key from the session image, marking the end of the handshake process. Following this, the C2 stager waits for C2 commands from the server.

When the server sends commands to the stager, it does so through a C2 image. The stager extracts the C2 commands from the image using the staging model and executes the command. Once the command execution is complete, the C2 stager encrypts the results with the session key and sends the encrypted results back to the server.

3. Generating images with C2 commands

Generative AI models have already explored to hide secrete data in actual pictures for the steganography and watermarking purpose. They usually train a general model that can extract any hidden data from arbitrary images. They shed a light on this direction, but it has false positives and false negatives. Therefore, it cannot be blindly reused in our C2 attack scenario. C2 attacker does not tolerate FP or FNs. We do not want to extract wrong session keys or unrecognizable commands from images for attack. In this case, we would like to customize existing AI models to adapt them into our use scenario. More specifically, we would like to customize the training process used by these models to provide the precise decoding capability and reduce the training time.

We train an encoder-decoder model jointly with an adversarial discriminator. Specifically, the encoder receives a cover image and a binary secret message and produces an encoded image. The decoder is designed to retrieve the message from an encoded image. The discriminator is trained to enable the encoder network to produce fake images (encoded images) that are indistinguishable from real images (cover images). We find that training a generic model for arbitrary commands often fails to achieve the accuracy level necessary to be applicable. Therefore, we propose to train models using pairs of images sampled from a finite subset, each paired with a predefined list of commands. We demonstrate that this produces images with the given commands which are indistinguishable to the human eye: as shown in example image1 and image2 below.

Figure 2 describes how our imageGen component works. We will provide a set of training images where we would like to hide C2 commands with a list of C2 secretes (C2 commands or C2 session keys). The model training process will take input C2 secretes and cover images, and outputs will be a visually indistinguishable encoded image, and a decoder which can recover the original message. In this paper, we adopted the model in () as our training model. In this model, both the encoder and decoder networks are trained simultaneously. Given an input message and a cover image, the encoder generates an encoded image that is visually indiscernible from the original. Subsequently, the decoder can extract and recover the original message from this encoded image. We will use the encoded image as our C2 or staging image, and the trained decoder model as our C2 or staging model. In the future, we can use more advanced latent diffusion models for our purpose.

As demonstrated by the two images generated by our model in Figure 3, they appear indistinguishable from conventional web images to the untrained eye. However, they are uniquely engineered to embed a C2 command ("whoami" in image1) and an encryption key ("62220ccff5"). The decoder model will be the part of the stager of the C2 at the client side,

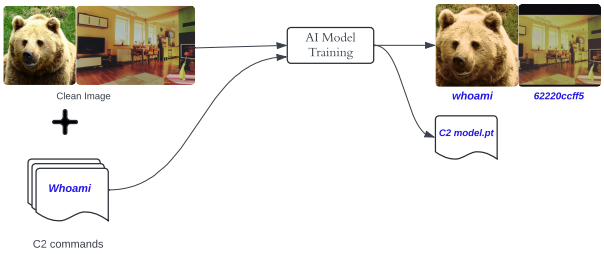

Figure 2. The overview of ImageGen component.

aiding in the retrieval of commands and encryption keys from images that are transmitted by the C2 server. The client will then execute these commands, encrypt the results of the execution using the encryption key, and subsequently send them back to the server.

Figure 3. The two demos of encoded images.

4. Evaluation

We evaluated our AI-enhanced C2 framework on one of the coco datasets(Lin et al., 2014). To evaluate the decoding efficiency of the AI-enhanced C2 framework, we randomly selected 10,000 images for testing purposes.

We conducted 2 sets of experiments. In the first setting, we train model for each image with arbitrary C2 commands. C2 commands include tasks, whoami, upload /opt/tools/bypassuac.exe etc.. For each pair of cover image and a C2 command, we train the model on it until that the bit error of the encoded image is zero. Then we will conduct the validation on the encoded image. For each encoded image, we will use the decoder model generated by the training process to extract the C2 command from the encoded image. If the extracted C2 command matches the original C2 command, then we think the cover image and the decoder model functions correctly. Otherwise, we think it fails. The output is 1000 decoder models for 1000 images.

In the second setting, we train the model for all of the images with a list of C2 commands in the single training process. In this case, we also reuse the same validation strategy used in the setting 1. The output is the only 1 decoder model, and it can decode C2

commands from 1000 images.

The experiments show that our model can successfully encode 15 C2 commands into 1000 images. The TPR is 100% in the experiment I. The 15 C2 commands can also be successfully extracted by using 1000 generated decoder models. The training time is around 5min per decoding model.

The results of Experiment II demonstrate that our model training requires one hour and is able to effectively encode 15 C2 commands into 1000 images. However, the True Positive Rate (TPR) for decoding is close to zero.

Overall, it means that in AI-enhanced C2 framework, it is more practical to train a C2 model per a cover image with C2 command.

5. Related Work

C2 attacks are increasingly using the steganography to conceal their activities. Invoke-PSImage (psimage) as one famous steganography tools, has been used by many attacks to hide information. Besides, some other attacks such as the OilRig-associated tool (link) to hide commands and data within bitmap images for C2 communication. However, the existing steganography technique in this attack merely involves hardcoding C2 command information into bitmap images, which makes them easy to detect. Recently, advanced generative AI techniques have revolutionized steganography, creating a fully trainable framework for data hiding. This presentation will introduce a new method by repurposing recent generative AI techniques for generating indiscernible images that can conceal arbitrary C2 commands.

There are existing efforts to embed arbitrary data into images using neural network models, such as the ECCV paper titled "HiDDeN: Hiding Data With Deep Networks" (Zhu et al., 2018). However, these methods cannot be directly applied to our C2 attacks because the methods lack sufficient accuracy for embedding and extracting messages for our C2 attacks. Building on existing work, we introduce new training techniques and model designs that significantly improve accuracy, making it suitable for our use scenario.Our current model design utilizes the encoder-decoder architecture; however, the method can be adapted to more recent generative techniques, such as latent diffusion models.

References

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll'a r, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. *CoRR*, abs/1807.09937, 2018. URL http://arxiv.org/abs/1807.09937.