My Project

Generated by Doxygen 1.8.13

Contents

1 Covert-Channel

Installation

Victim Box

1. On Ubuntu 18.04, apt install Python 3.6 and python-pip3
   (a) sudo apt install python3.6 python-pip3
2. Use pip3 to install opencv, bitstring, and numpy
   (a) sudo pip3 install opencv-python bitstring numpy
3. Ensure the global variables in CovertClient.py identify the CovertServer IP address and the paths for the data file to hide, the image to hide it in, and the path to write the new image containing the hidden data to.

Personal Box

1. On Ubuntu 18.04, apt install Python 3.6 and python-pip3
   (a) sudo apt install python3.6 python-pip3
2. Use pip3 to install flask, opencv, bitstring, and numpy
   (a) sudo pip3 install flask opencv-python bitstring numpy
3. Ensure the global variables in CovertServer.py identify the path to write the exfiltrated image and extracted data to.

Exfiltrate File

1. Run CovertServer.py on personal box.
   (a) export FLASK_APP=CovertServer.py
   (b) run flask –host=0.0.0.0
2. Run CovertClient.py on victim box to exfiltrate data to CovertServer.
   (a) ./CovertServer.py

2 Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

3 Namespace Documentation

3.1 CovertClient Namespace Reference

Functions

- def print_usage ()
  Print usage information if user provides help arguments or provides too many arguments.
- def check_args ()
  Parse command line arguments.
- def check_image_validity (path_to_image)
  Check if the image exists and can be read from.
- def check_file_exists_to_read (path_to_data)
  Check if a file exists on the disk and can be opened.
- def check_data_validity (path_to_data)
  Check if the data file exists and can be read from.
- def check_size_validity (image, file_bits)
  Checks if image contains enough least significant bits to store data from file.
- def check_validity_and_read (path_to_image, path_to_data)
  Check if the image and data files exist and can be read from.
- def store_data_in_image (image, file_data)
  Stores file data into the two least significat bits of the three color components (Red, Green, Blue) of the pixels in an image.
- def send_image ()
  Sends image containing file data to flask server listening for HTTP POST requests.
- def main ()
  Orchestrates hiding file data in an image and sending that image to server

Variables

- string **default_path_to_image** = "image.png"
- string **default_path_to_data** = "data.txt"
- string **default_path_to_write_image** = "hidden.png"
- string **serverIP** = "127.0.0.1"

3.1.1 Detailed Description

```
@package CovertClient covert channel to exfiltrate data
```

3.1.2 Function Documentation

3.1.2.1 check_args()

```
def CovertClient.check_args ( )
```

Parse command line arguments.

Print usage information and exit if user inputs help arguments or inputs too many arguments. Declare path_to_image and path_to_data variables based on provided arguments.

3.1.2.2 check_data_validity()

```
def CovertClient.check_data_validity (
             path_to_data )
```

Check if the data file exists and can be read from.

Parameters

path_to_data	Relative path to data file to check

Returns

bitarray of data file

3.1.2.3 check_file_exists_to_read()

```
def CovertClient.check_file_exists_to_read (
             path_to_data )
```

Check if a file exists on the disk and can be opened.

On error, print error message and exit.

Parameters

path_to_data	Relative path to data file to check

Returns

file descriptor to file at path_to_data

3.1.2.4 check_image_validity()

```
def CovertClient.check_image_validity (
              path_to_image )
```

Check if the image exists and can be read from.

Parameters

path_to_image	Relative path to image to check

Returns

2D numpy array representation of every pixel in image

3.1.2.5 check_size_validity()

```
def CovertClient.check_size_validity (
              image,
              file_bits )
```

Checks if image contains enough least significant bits to store data from file.

Parameters

image	Image object to check
file_bits	BitArray of file data to check

3.1.2.6 check_validity_and_read()

```
def CovertClient.check_validity_and_read (
              path_to_image,
              path_to_data )
```

Check if the image and data files exist and can be read from.

Reads data into objects if possible and returns them.

Parameters

path_to_image	Relative path to image to check and read
path_to_data	Relative path to data file to check and read

Returns

Image object created from image at path_to_image
BitArray read in from data file at path_to_file

3.1.2.7 store_data_in_image()

```
def CovertClient.store_data_in_image (
              image,
              file_data )
```

Stores file data into the two least significat bits of the three color components (Red, Green, Blue) of the pixels in an image.

The image is modified in place, and any pixels beyond the amount required to store the file data are unmodified.

Parameters

image	Image object to hide data file in
file_data	Data from data file to hide in image

3.2 CovertServer Namespace Reference

Functions

- def post_image ()
  Parses image file from POST request and writes it to disk.
- def extract_data (path_to_image, path_to_file)
  Extracts a hidden file from the two lowest significant bits of color components (Red, Green, Blue) in pixels of the image.

Variables

- string **default_path_to_image** = "exfiltrated.png"
- string **default_path_to_data** = "extracted.txt"
- **app** = Flask(__name__)
- **methods**

3.2.1 Detailed Description

```
@package CovertServer covert server to collect exfiltrated data
```

3.2.2 Function Documentation

3.2.2.1 extract_data()

```
def CovertServer.extract_data (
              path_to_image,
              path_to_file )
```

Extracts a hidden file from the two lowest significant bits of color components (Red, Green, Blue) in pixels of the image.

Writes extracted file to the disk.

Parameters

path_to_image	Relative path to image to extract hidden file from
data_file	Relative path to file to write extracted data to

3.2.2.2 post_image()

```
def CovertServer.post_image ( )
```

Parses image file from POST request and writes it to disk.

The flask server calls this function when it recieves a post request to the root directory.

Returns

Always returns 200 OK HTTP reponse

Index