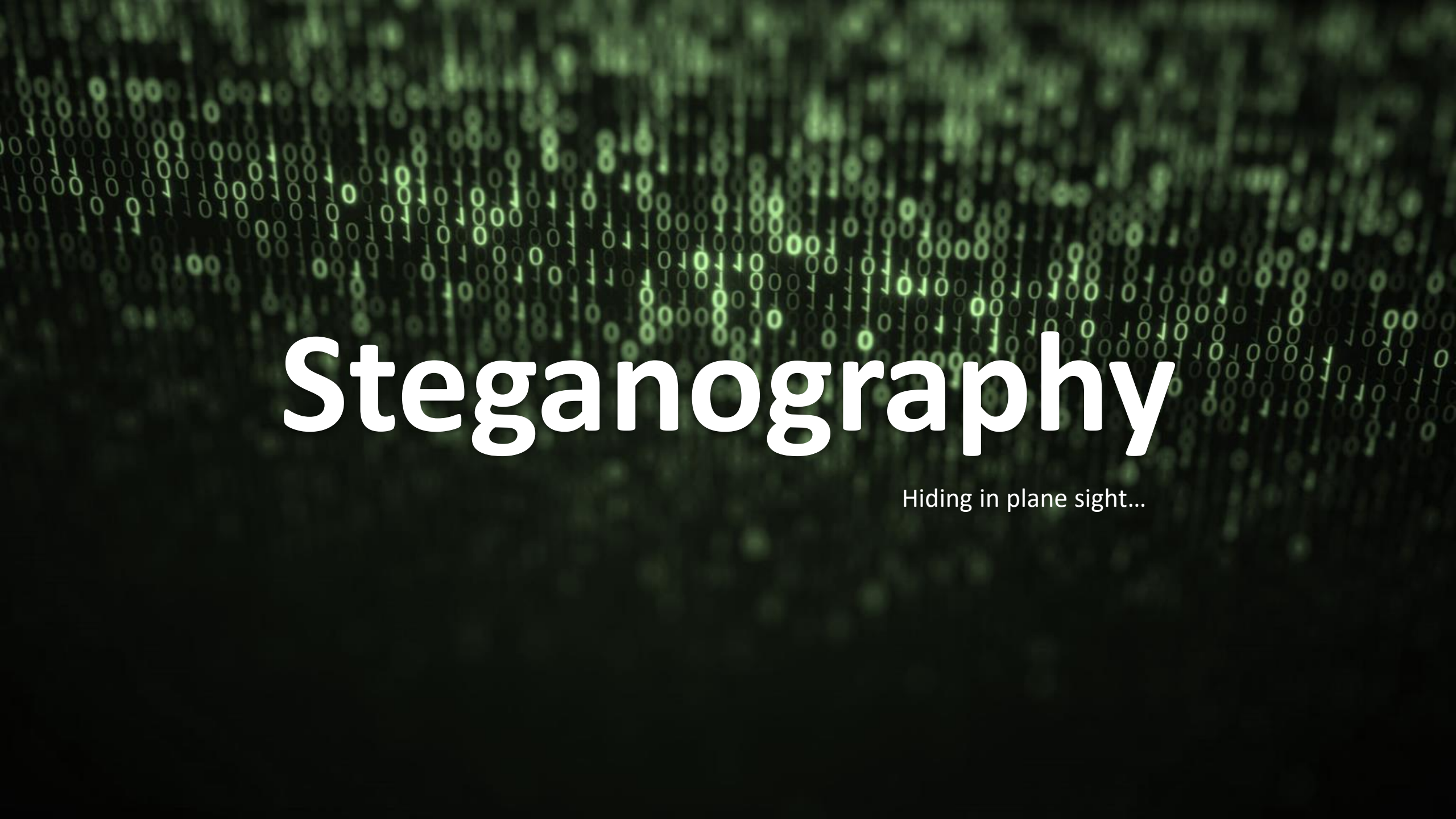
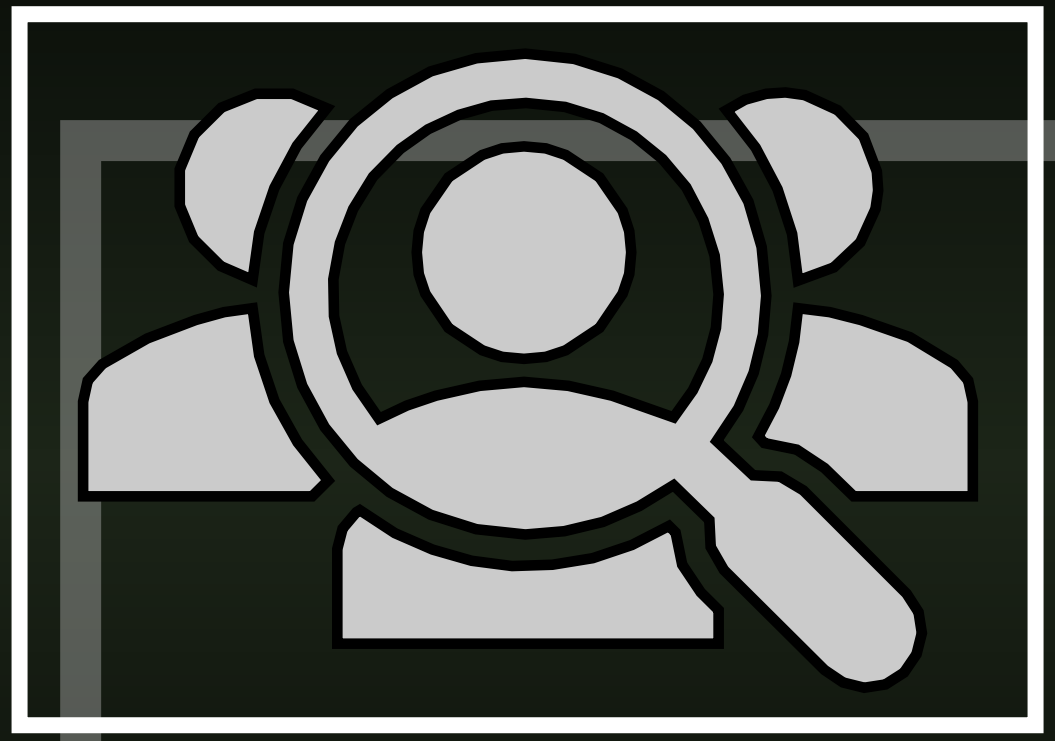
What do these have in common?

Since Everyone Can Read, Encoding Text In Neutral Sentences Is Doubtfully Effective


Steganography
Hiding in plane sight…



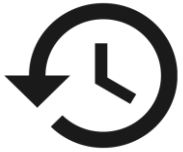
Kc Udonsi

Engineer I

Kc.Udonsi@td.com

4163072532

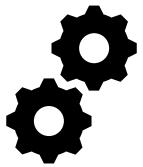
0x0



History

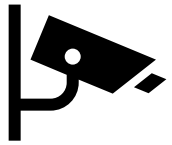
Usages

Tactics



Techniques - *

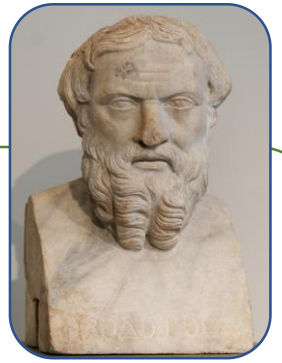
Tools



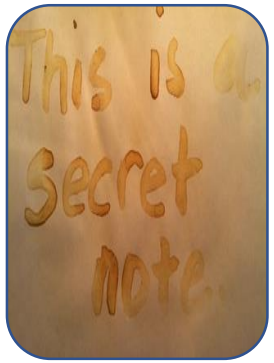
Detection

0x1

History



- Earliest known record was from 440 BC. Herodotus mentioned two (2) examples of steganography
- The Greeks commonly communicated over wax tablets. Cruel Xerses once wrote a message on the tablet before applying the wax



- Ancient Chinese messengers swallowed wax coated silk tiny balls with messages written in the silk
- Special Inks that could be revealed under certain conditions like blue-light or heat.
- The use of Frequency Domains, Bit/Byte patterns, extended character set/encodings

0x2

Usages

Finger-Printing

- Watermarking Digital documents
- Can mark each digital copy with signature for intended recipient
- Decode document for signature after leak to discover mole
- E.g FBI VS Reality Leigh Winner 2017

Smuggling Data

- Embedding a file or data into another file or data of different MIME type to bypass content filters
- Can hide binary in Text files using zero-width encoding or LSB with image hosts
- APT 32 a.k.a OceanLotus smuggled backdoors using steganography

Phishing

- Using homoglyphic characters to construct seemingly trusted URL domains
- Homoglyphic characters are nearly impossible to tell apart
- Detection usually involves certificate analysis
- E.g these are not the same sites: https://epic.com/ and https://epic.com

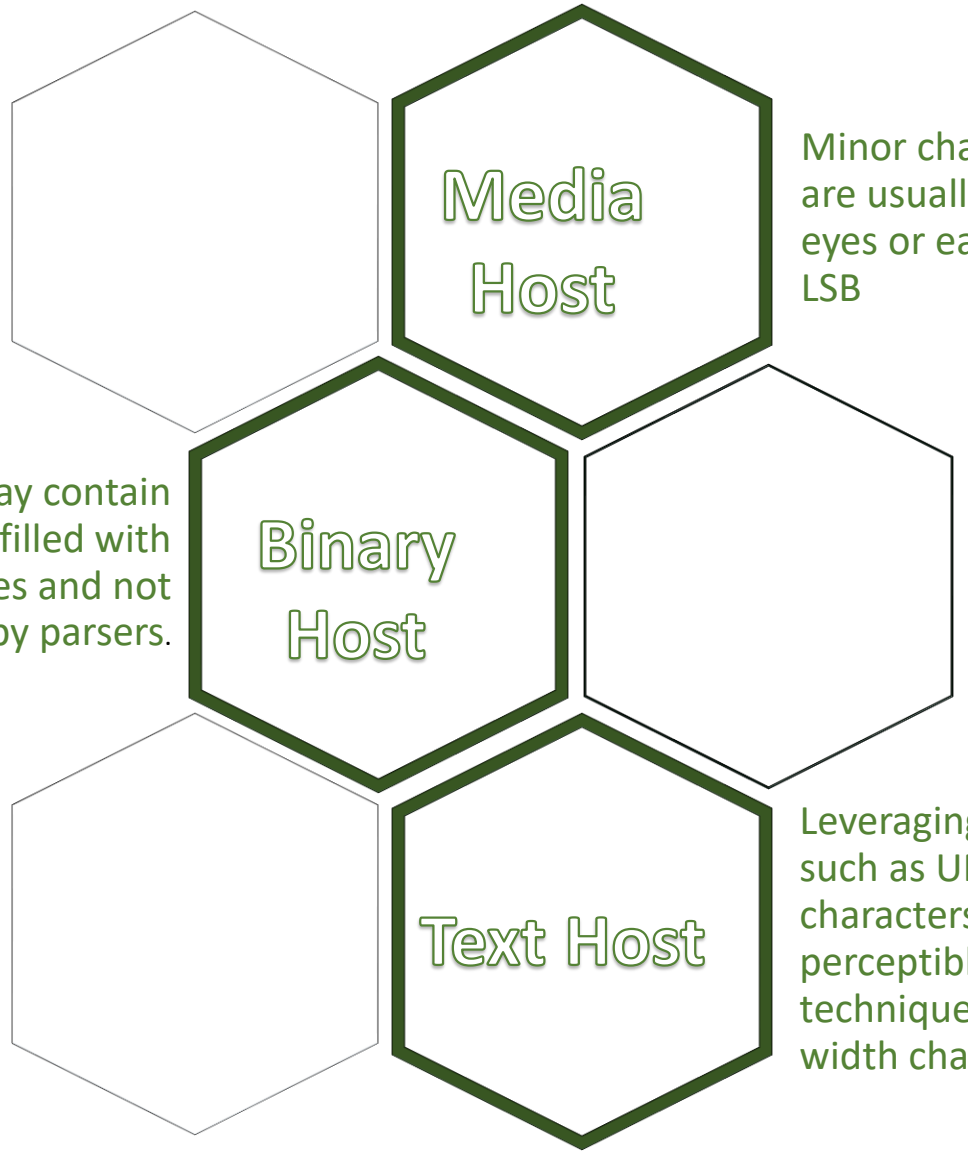
Secret Messaging

- Security through obscurity but can be encrypted as well
- Group messaging using zero-width chars to chat with specific recipient
- Memes/GIFs with encoded data for coolness points
- Fingerprinting messages leaked from chats

Things are not always what they seem; the first appearance deceives many --Phaedrus

0x3

Tactics



Minor changes in bits of media files are usually not perceptible by human eyes or ears. Example algorithms in LSB

Binary files may contain caves; regions filled with NULL bytes and not interpreted by parsers.

Leveraging encodings other than ASCII such as UNICODE to (mis)represent characters. Changes are also not easily perceptible to the eyes. Example techniques include homoglyphs and zero-width characters

Things are not always what they seem; the first appearance deceives many --Phaedrus

0x4

Techniques – Zero-Width Characters

Zero-Width Characters: Non-printing Unicode characters such as zero-width space (U+200B), zero-width non-joiner (U+200C)

Decode Me!

- Convert each character to be hidden into its binary representation

  1. “A” => 65 => 0x41 => “01000001”

- Convert to zero-width by iterating through the binary string, converting each 1 to zero-width space and each 0 to zero-width non-joiner. Delimit binary strings with zero-width joiner.
  - “0100” => U+200CU+200BU+200CU+200C

- Insert block of hidden text into host text.

Things are not always what they seem; the first appearance deceives many --Phaedrus

```python
#!/usr/bin/python3

def hide(decoy: str, secret: str) -> str:
    '''
    Embed secret as a zero-width string within decoy
    '''
    # split secret by spaces
    secret = secret.split()

    lookup = {}
    lookup["1"] = '\u200b'
    lookup["0"] = '\u200c'
    # split each word into binary repr of its characters and convert binary repr to zero-wdith characters
    zwsecret=[]
    for word in secret:
        # get the letter list
        letterlist = list(word)
        # convert the letters from ASCII to binary
        binletterlist = map(lambda letter: format(ord(letter), 'b').zfill(8), letterlist)
        # convert the letters in binary to zero-width letterlist
        zwletterlist = map(lambda binletter: "".join([lookup[c] for c in binletter]), binletterlist)
        # re-assemble the word as zero-width
        zwword = "\u200d".join(zwletterlist)
        zwsecret += [zwword]

    return "{decoy}{zws}".format(decoy=decoy, zws="\uFEFF".join(zwsecret))


```

```python
#!/usr/bin/python3

def unhide(decoy_with_secret: str) -> str:
    '''
    Reveal the secret within a text

    '''
    # Ignore all characters but the ones in our encoding
    secret = "".join([ '' if c not in ['\u200b', '\u200c', '\u200d', '\uFEFF'] else c for c in decoy_with_secret])

    # split by non-breaking space
    zerowidthwordlist = secret.split('\uFEFF')
    lookup = {}
    lookup['\u200b'] = "1"
    lookup['\u200c'] = "0"

    # split each word into binary repr of its characters and convert binary repr to char
    sentence=""
    for zwword in zerowidthwordlist:
        # get the letter list
        zwletterlist = zwword.split('\u200d') # [01000001, 01010111] but as zero-width
        # conver the letters from zero-width to ASCII and re-concatenate
        binletterlist = map(lambda zwletter: "".join([lookup[zw] for zw in zwletter]), zwletterlist) # [01000001,
01010111]
        asciiword = "".join(map(lambda bin: chr(int(bin, 2)) if not (bin == "") else '', binletterlist))
        sentence = sentence + " " + asciiword
    return sentence.strip()


```

0x5

Techniques – Least Significant Bit

The simplest form of media steganography whereby the LSB of image bytes are replaced with the bits of the message to be hidden

- Example media host is a 24-bit image
  - 10001010 01011010 01110010
    01010001 10100001 01011101
    00010100 00100011 11100011
- Example number to hide is 255. In binary string
  - 255 => 11111111

- Output
  - 10001011 01011011 01110011
    01010001 10100001 01011101
    00010101 00100011 11100011

```python
#!/usr/bin/python3

from PIL import Image

def encode(rgb: tuple, data: list) -> tuple:
    oldrgbasbin = list(map(lambda code: format(code, 'b').zfill(8), rgb))
    for i in range(len(oldrgbasbin)):
        # modify the LSB of every byte
        oldrgbasbin[i] = oldrgbasbin[i][:-1] + (data[i] if i < len(data) else oldrgbasbin[i][-1])
    return tuple(map(lambda bin: int(bin, 2), oldrgbasbin))

def hide(filename: str, message: str) -> bool:
    img = Image.open(filename)
    # add a MARKER so we know when to stop looking
    binary = '{}0000111100001110'.format(''.join(format(ord(c), 'b').zfill(8) for c in message))
    if img.mode in ('RGBA'):
        # forcefully make it Red, Green, Blue, Alpha
        img = img.convert('RGBA')
        # get all the pixels
        pixels = img.getdata()

        newpixels = []
        digit = 0
        temp = ''
        for pixel in pixels:
            if (digit < len(binary)):
                r,g,b= encode((pixel[0],pixel[1],pixel[2]),binary[digit:digit+3])
                        # give the new pixel Alpha of 255 and add to our new image
                newpixels.append((r,g,b,255))
                digit += 3
            else:
                newpixels.append(pixel)
        img.putdata(newpixels)
        img.save(filename, "PNG")
        return True
    return False
```

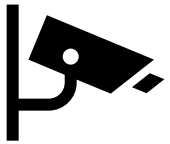
```python
#!/usr/bin/python3

from PIL import Image

def decode(rgb: tuple) -> tuple:
    return "".join(map(lambda code: format(code, 'b')[-1], rgb))

def binarytoascii(binary: str) -> str:
    return "".join([chr(int(binary[i:8+i],2)) for i in range(0, len(binary), 8)])

def unhide(filename: str) -> tuple:
    img = Image.open(filename)
    binary = ''

    if img.mode in ('RGBA'):
        img = img.convert('RGBA')
        pixels = img.getdata()
        for pixel in pixels:
            extract = decode((pixel[0],pixel[1],pixel[2]))
            binary = binary + extract
            # stop as soon as we recognize our MARKER
            if ('0000111100001110' in binary):
                return True, binarytoascii(binary)
        # we didn't find MARKER but we try anyway
        return False, binarytoascii(binary)
    return False, ""

```

0x6

Tools

- StegHide (WAV, BMP)
- Steganos
- Invisible Secrets
- https://330k.github.io/misc_tools/unicode_steganography.html - Zero-Width Characters
- https://stylesuxx.github.io/steganography/
- Build Your Own Tools

0x7



Detection

- Need to be aware steganography has been deployed on host or medium
- Can be challenging to find embedded data especially if custom algorithms were deployed
- Steganography may be used alongside cryptography making the detection/data retrieval even more challenging
- Zero-width encoding may be detected by character count where present
- Images may be brute-forced for detection by trying every common decoding scheme


Questions


Thank You

References

- https://www.slideshare.net/UttamJain/steganography-14902856
- https://publications.computer.org/computer-magazine/2018/11/15/how-steganography-works/
- https://medium.com/@umpox/be-careful-what-you-copy-invisibly-inserting-usernames-into-text-with-zero-width-characters-18b4e6f17b66
- https://blog.fastforwardlabs.com/2017/06/23/fingerprinting-documents-with-steganography.html
- http://csis.pace.edu/~ctappert/srd2005/d1.pdf
- https://www.ptiglobal.com/2018/04/26/the-beauty-of-unicode-zero-width-characters/