Stegosploit

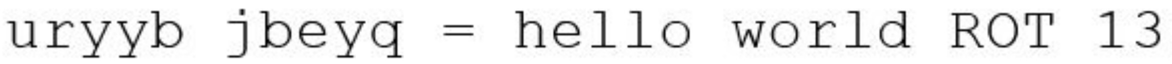
By Arjun and Brandon

Outline

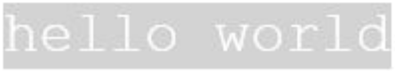
1. Terminology
2. Background
3. Vulnerability + Exploits
4. Impact
5. Mitigation

What is **Steganography**?

- Comes from the Greek words steganos (to conceal) and graphein (writing)
- Hiding a secret message within a medium such as a picture, document or video
- How is it different from Cryptography?
  - Cryptography attempts to hide the meaning of a message
  - Steganography attempts to hide the entire message in plain sight



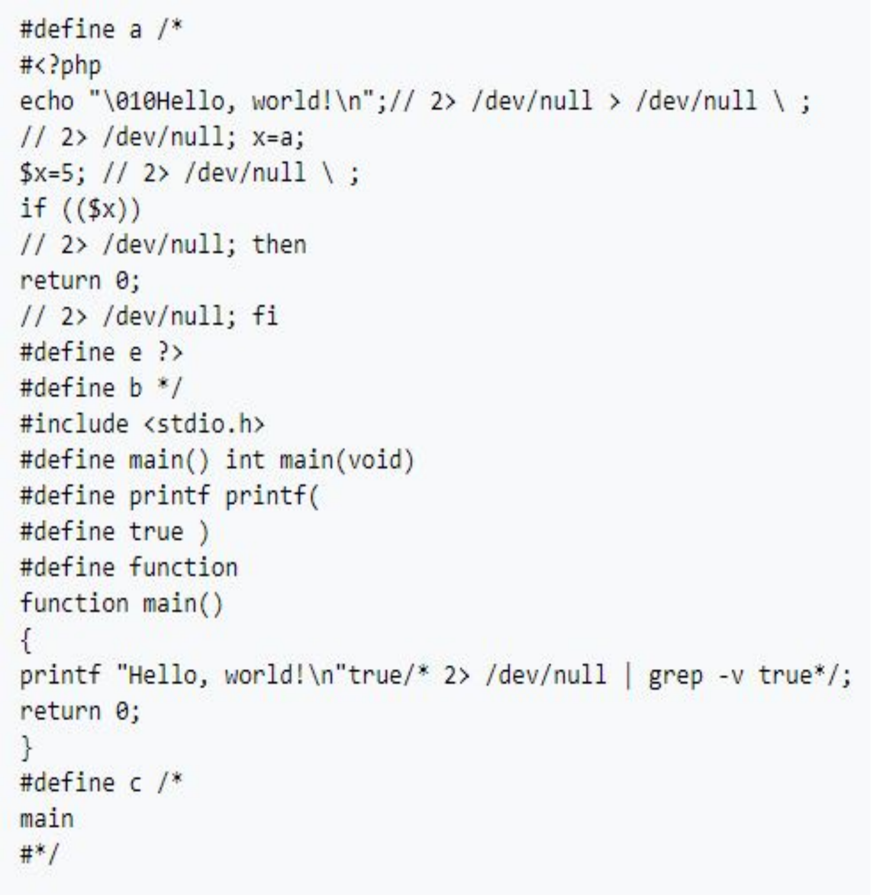
uryyb jbeyq = hello world ROT 13



hello world

What is a **Polyglot**?

- Two or more data formats in a single container that coexist without breaking each others syntax or specifications
- Code written that can be interpreted as C, Bash script, and PHP
- Bash returns “\010Hello, world!\n Line 5: a=5: command not found Hello, world!”
- PHP returns “#define a /* Hello, world!”
- C returns “Hello, world!”

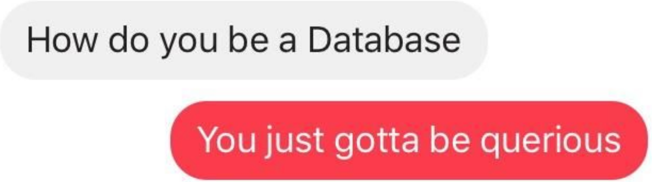


```
#define a /*
#<?php
echo "\010Hello, world!\n";// 2> /dev/null > /dev/null \ ;
// 2> /dev/null; x=a;
$x=5; // 2> /dev/null \ ;
if (($x))
// 2> /dev/null; then
return 0;
// 2> /dev/null; fi
#define e ?>
#define b */
#include <stdio.h>
#define main() int main(void)
#define printf printf(
#define true )
#define function
function main()
{
printf "Hello, world!\n"true/* 2> /dev/null | grep -v true*/;
return 0;
}
#define c /*
main
#*/
```

Images are Innocent?

Me: Makes a small CSS change

My Site:



How do you be a Database

You just gotta be querious

“A good exploit is one that’s delivered with style.” - Saumil Shah

Hiding In Plain Sight

can't stop what you can't see

Stegosploit = Steganography + Polyglot

Stegosploit is …

- NOT an exploit
- NOT an XSS attack or webshell
- NOT a manipulation of EXIF data

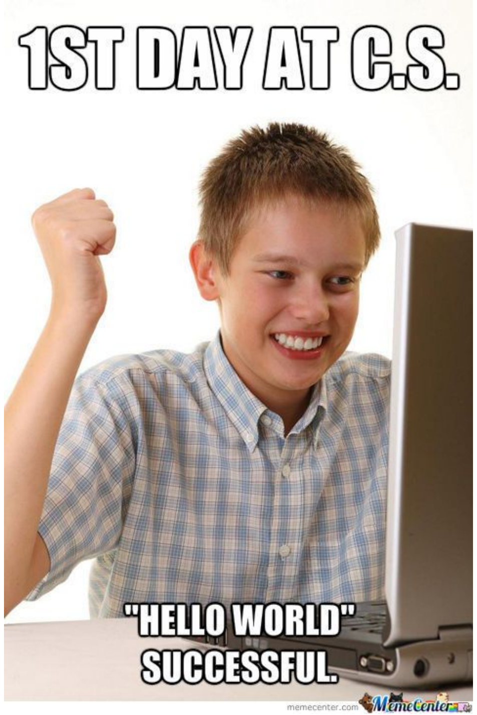
BUT...

Stegosploit is a way to deliver web browser exploits using pictures.

Stegosploit goals

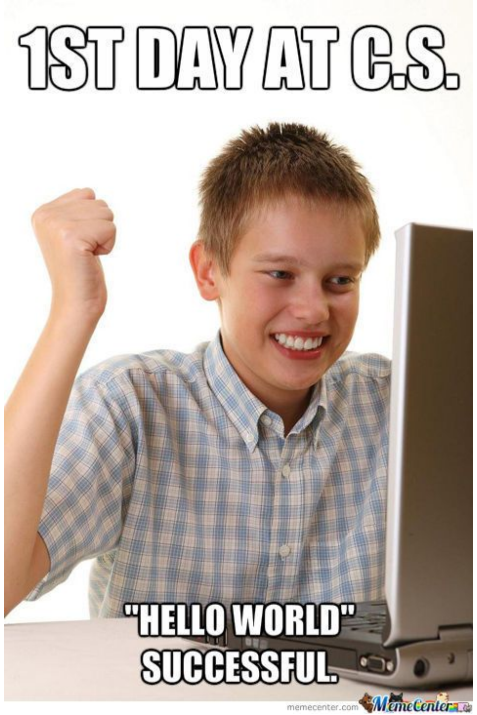
1. No extra data sent over the network
2. The image looks “normal” and not deformed or distorted
3. The exploit code within should not seen within the image file as a string
4. The image decodes itself and executes the exploit without ANY user interaction aka “autorun”

What’s the difference?



<script src=”boy.jpg”></script>

- Sees and interprets jpg as a bunch of Code



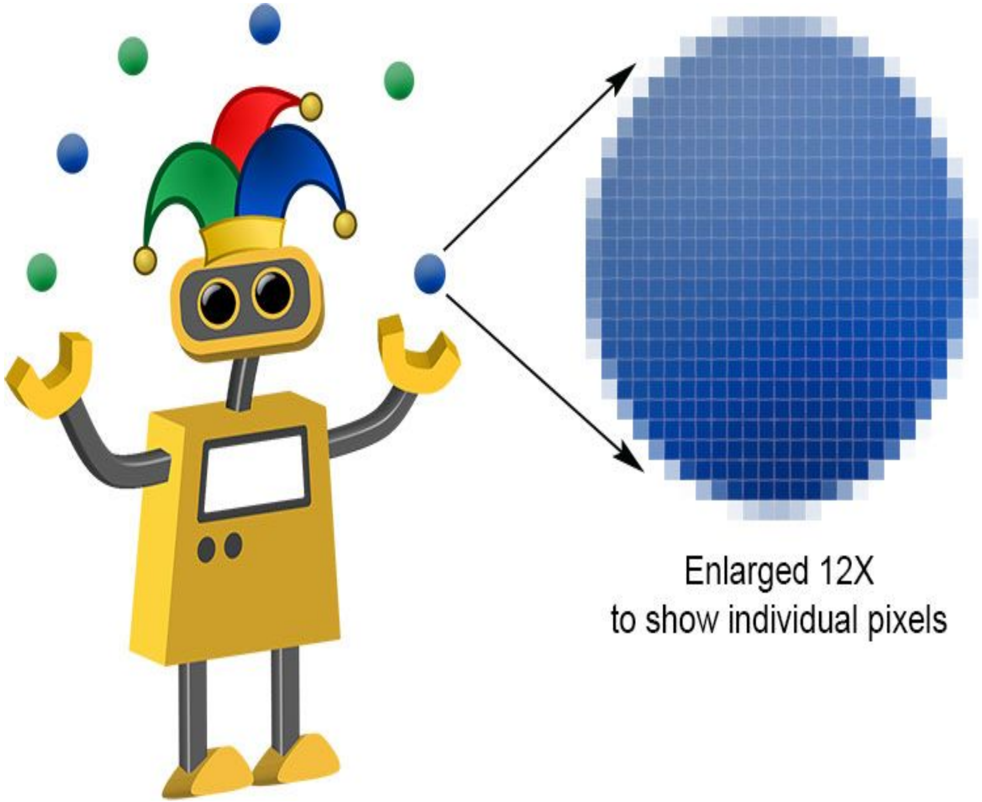
<img src=”boy.jpg”>

- Sees and interprets jpg as a bunch of Pixels

Let’s start the Stegosploit!

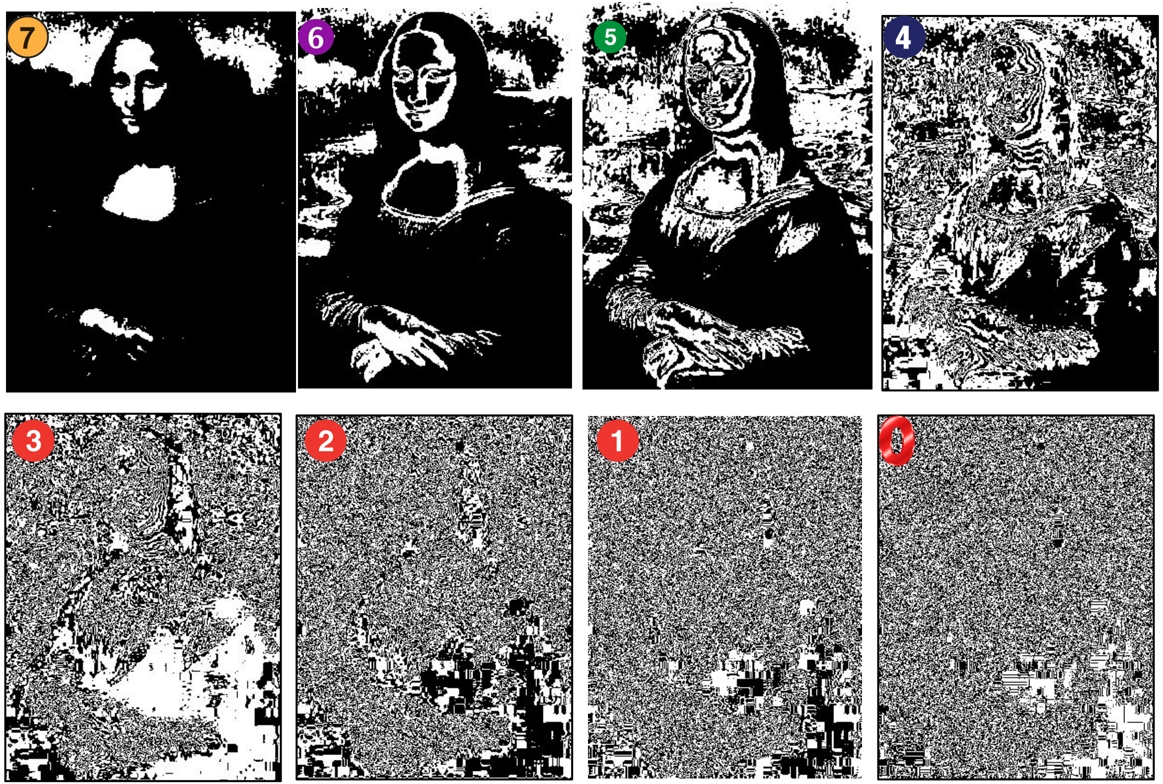
Closer Look at **Pixels**

- **Pixel** = specific number converted into bits (bitmap) that tell computer what colour the pixel should be
- **Bitmap** = sequence of bits defining the colour of each pixel
- More bits = more possible tone of colour



Enlarged 12X
to show individual pixels

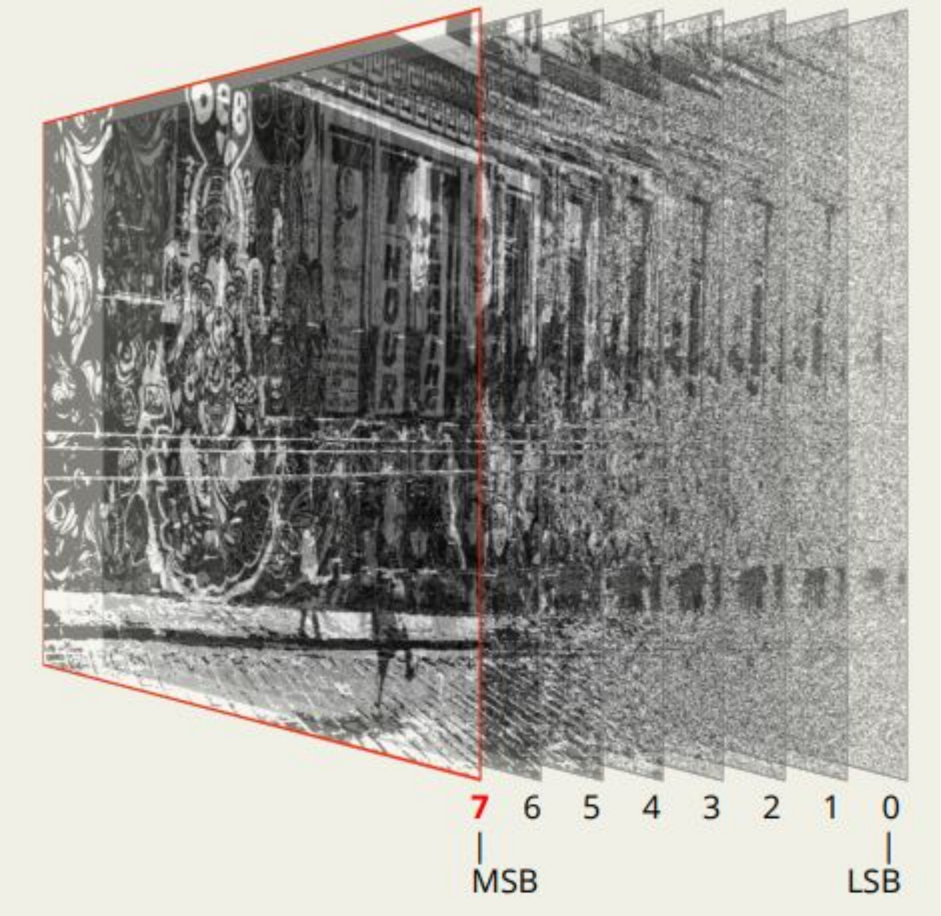
Closer Look at Pixels



- Images are a **composition of bits** and made with a bitmap
- Top left bit (Bit 7) = More Shape, Less Detail
- Bottom Right Bit (Bit 0)= More Detail, Less Shape

Encoding Exploit Code

- Exploit code converted to bitstream
- Pixel bits of layer *your choice* are overwritten with exploit bitstream



DEMO #1: Encoding Exploit into pixel data of JPG and PNG

PNG Encoding vs JPG Encoding

PNG = **Lossless compression**

- Picture Is saved at best quality with all pixels intact
- Only need to encode it with exploit once

JPG = **Lossy Compression**, meaning quality of pic is reduced

- Pixels are approximated to nearest neighbours

JPG Exploit Fix…. **Iterative Encoding**!

- Lets just keep on encoding the image until our exploit code sticks and is not messed up by the nearest neighbours loss

But wait how do we decode?!?

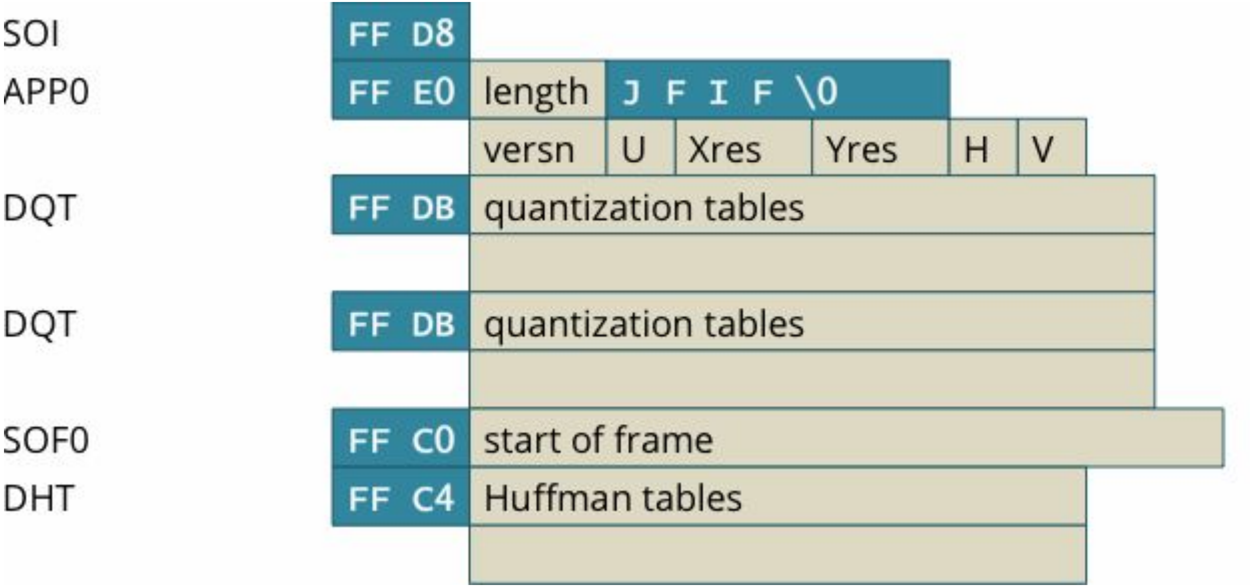
HTML5 Canvas to the Rescue!

- Read image pixel data using JS
- In-Browser decoding of steganographically encoded images
- Rebuild JS exploit code from pixel data, in memory
- Simple array and bit manipulation operations

IMAJs = Images + Javascript

- We can make polyglot pictures by encoding them with Javascript or Actionscript code
- By taking advantage of that vulnerability, we are able to deliver exploits via images like bmp, jpeg, and png's
- First coined by Saumil Shah at SyScan 2015, Singapore

JPG/PNG Headers



SOI FF D8

APP0 FF E0 length J F I F \0

versn U Xres Yres H V

DQT FF DB quantization tables

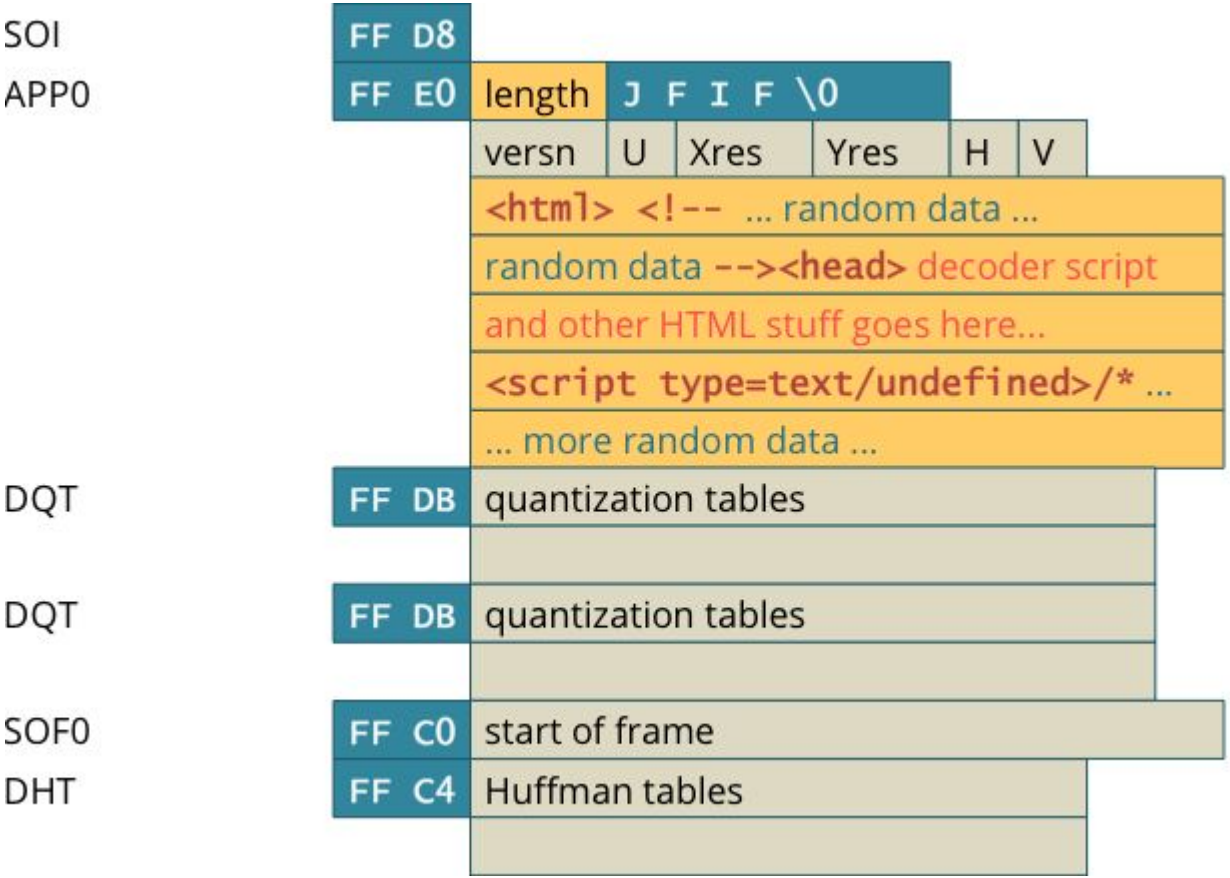
DQT FF DB quantization tables

SOF0 FF C0 start of frame

DHT FF C4 Huffman tables

- A JPG is composed of many different headers
- Interested in how we can put our decoder code into JPG
- JPG will decode the exploit and run it!

Lets Just Change the Length and Add Our Code!



SOI FF D8

APP0 FF E0 length J F I F \0

versn U Xres Yres H V

<html> <!-- ... random data ...

random data --><head> decoder script

and other HTML stuff goes here...

<script type=text/undefined>/* ...

... more random data ...

DQT FF DB quantization tables

DQT FF DB quantization tables

SOF0 FF C0 start of frame

DHT FF C4 Huffman tables

Demo 2: Decoding, creating Polyglot and Running Exploit

Putting it all Together

STEP 1) Have some exploit code you want to run and any image of your choice

STEP 2) Encode the code into the image. Using Steganography!

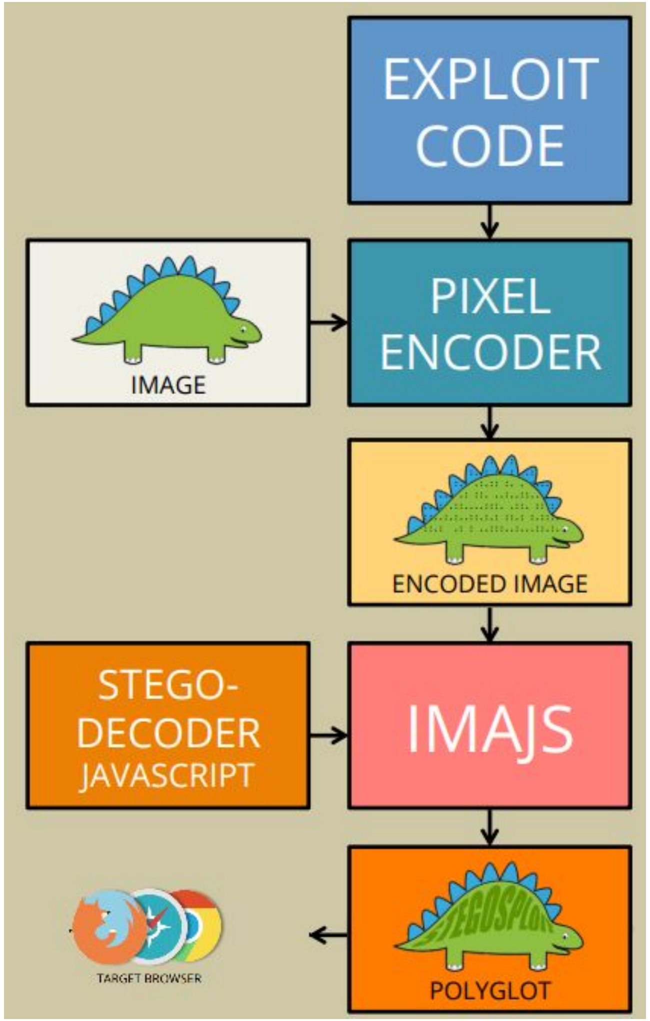
STEP 3) Turn encoded exploit image into a polyglot by changing image headers length + add decoder code

STEP 4) Send the polyglot image through the network then into the browser. Make sure that you are only sending the image data and no extra “baggage”!

STEP 5) Have the image load into the vulnerable browser

STEP 6) Image now decodes itself and is executed as javascript

STEP 7) $$$



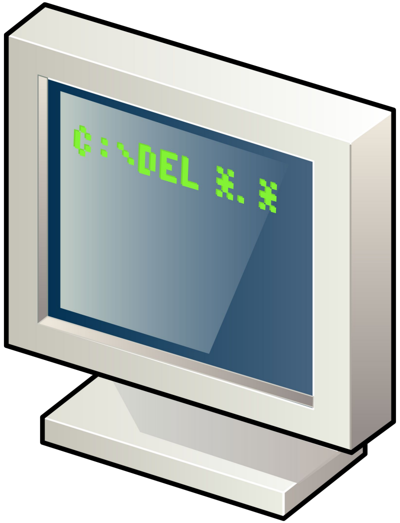
Reaching the target browser

As an attacker, we have the following options for sending the HTML+Image polyglot to the victim's browser:

- Host the image on an attacker controlled web server and send its URL link to the victim.
- Host the entire exploit on a URL shortener.
- Upload the image on 3rd party websites and provide direct links.

Real World Impact

- As of now, there are no reports of Stegosploit affecting anyone at a large scale
  - Fair new and undiscovered potential
- Current implementation of Stegosploit is difficult to use
  - A lot of things need to align for Stegosploit to work!
    - Target website needs to allow images to be uploaded by users
    - The website does not corrupt the data inside the image
    - The website loads the uncorrupted image uploaded by user
    - Different browsers read images differently



Potential Exploits Run with Stegosploit

- Firefox 3.5 Font Tags Buffer Overflow (CVE-2009-2478)
  - Exploit: https://www.exploit-db.com/exploits/9137/
  - Heap spray vulnerability
  - Remote code execution
  - This exploit tries to run shell
- Microsoft Internet Explorer 8/9/10 - 'CInput' Use-After-Free Crash
  - Exploit: https://www.exploit-db.com/exploits/33860
  - Use after Free crash vulnerability
  - The use of heap allocated memory after it has been freed or deleted leads to undefined system behavior

Demo 3: The True Power of Stegosploit

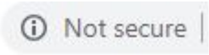
Full Demo by Saumil Shah - displaying the full potential of Stegosploit

https://youtu.be/6lYUtIZHlJA?t=1815 (30:15 - 37:15)

Mitigation

- Detection can be difficult since Stegosploit is a new vector for attacking
- Anti-viruses need to analyze images / videos outside of the normal means of detection
- Sanitize user input: Change file formats from JPG to PNG then back to JPG
  - This lowers the quality of image
  - Damages potential steganographic contents
- Resize the image
  - Same idea!
- Is the original size of image / video smaller than the image provided
  - Malicious code, be careful!
- **Re-encode the whole image! (Best Solution)**

Mitigation - Browsers, W3C and Users

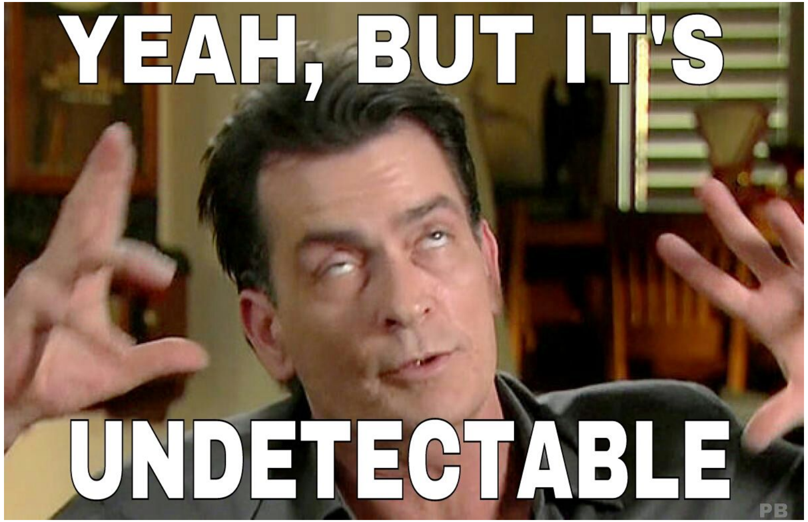


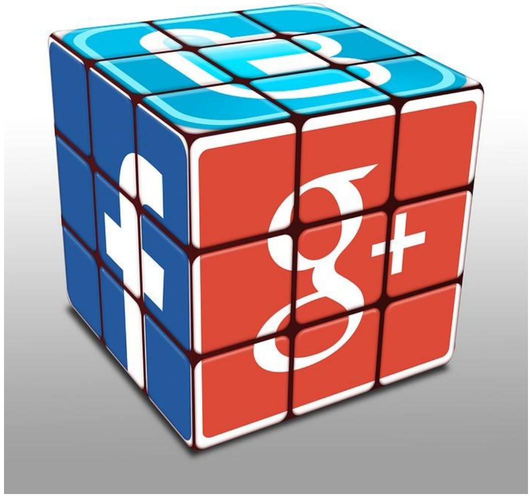
ⓘ Not secure

- W3C need to be strict like compilers
  - For example, if you miss a semicolon, the compiler says too bad!
  - Browsers on the other hand... Don’t just ignore improper syntax!!
- Browsers need to reject things that do not follow strict standards
- As a user, avoid going on unsecure websites
- Have an updated browser that has MIME type checking

On the Bright Side

- Many of the main websites reencode images
  - This helps prevent any Stegosploit since it corrupts the data
- Websites like Google, FaceBook, and Twitter all reencode of the images





References

http://stegosploit.info/

https://www.alchemistowl.org/pocorgtfo/pocorgtfo08.pdft

https://youtu.be/np0mPy-EHII

https://www.sans.org/reading-room/whitepapers/stenganography/steganalysis-detecting-hidden-information-computer-forensic-analysis-1014

https://www.blackhat.com/docs/eu-15/materials/eu-15-Shah-Stegosploit-Exploit-Delivery-With-Steganography-And-Polyglots.pdf

https://www.slideshare.net/saumilshah/stegosploit-hacking-with-pictures

https://www.youtube.com/watch?v=6lYUtIZHlJA

Questions?