Steganography

Artem Albert and Ju Hong Kim

Steganography

Definition: the practice of concealing messages or information within other non secret text or data

- Invisible Ink
- Can be done via many forms such as pictures, videos, and audio

Digital Example - Hiding Images

Image

Hidden Image

Linux

```
zaku@zaku-laptop:~/Pictures/stegasploit$ cmp windows.jpg windows_orig.jpg
cmp: EOF on windows_orig.jpg after byte 3779, in line 13
```

```
zaku@zaku-laptop:~/Pictures/stegasploit$ stat windows_orig.jpg | grep Size
  Size: 3779            Blocks: 8          IO Block: 4096   regular file
```

How is it different from Cryptography?

- Cryptography intends to make messages unreadable to protect the content of the message
- Steganography purpose is to make messages not appear in plain sight

What is the advantage of steganography?

- Message not in plain sight and the medium seems innocent
- Can be employed in any medium unlike cryptography where it can only be in the form of encrypted text

Hello in Morse Code through knitting

.... . .-.. .-.. ---

What about hiding text

Original Image md5sum: 74331ad263c3948a5350c0ad49be8f96

New Image md5sum: 877070cf2abd3e390c581035d8a6b723

Image

Raw Data

```
%^N^HpA^H^P <82>^@<86> <82>^B¡^X <80><98><93>^D^P^HÄ<98> <80>P^X <85>^T^PA^CÿÙint main() { printf("Hello World\n"); return 0; }
```

If we can insert text, can't we add code as well.
Can we get code to execute?

Polyglot -

Definition: knowing or using several languages

Computing: Two or more scripting or programming language that coexist without invalidating syntax

```
#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
#*/
```

Recall:

is a comment in Bash + PHP
in C is for preprocessor
/* Is a comment in PHP + C */
2> redirect undesired output to stderr
printf exists in both C + Bash (bash does not need brackets)

```
zaku@zaku-laptop:~/Pictures/stegasploit$ gcc -E test.c | tail
# 17 "test.c"
int main(void)
{
printf( "Hello, world!\n") ;
return 0;
}
```

```
zaku@zaku-laptop:~/Pictures/stegasploit$ ./a.out
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
#*/
```

```
zaku@zaku-laptop:~/Pictures/stegasploit$ bash test.bash
\010Hello, world!\n
test.bash: line 5: a=5: command not found
Hello, world!
```

Stegosploit = Steganography + Polyglots

- A way to deliver web browser exploits via hiding exploit in an image
- Exploit code is hidden in the image (steganography)
- Exploit code does not violate image format syntax and is executed in the browser

IMAJS BMP: Combining JS with BMP

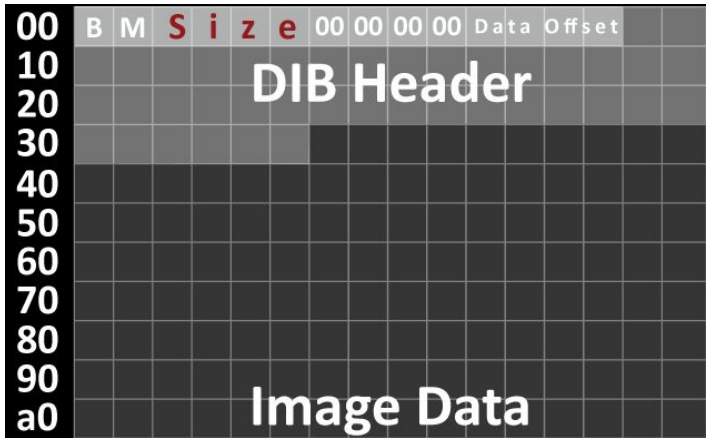
IMAJS: A polyglot of an Image and JavaScript

File Format:

- First 2 Bytes: BM

42 4D XX XX XX XX 00 00 00 00 .....

B M Filesize Empty Empty DIB data



FILE FORMAT:

42 4D XX XX XX XX 00 00 00 00 .....

B M Filesize Empty Empty DIB data

- Set the file size as “2F 2A XX XX”
  - 0x2A2F (hex) is “/*” which is a comment in Javascript
  - Filesize does not matter since the browser will not read whatever is encapsulated in the comment (Replace first two characters of filesize with “/*”)
  - Most browsers will render the image properly even if the header of BMP is messed up
- Append to the BMP image the exploit code

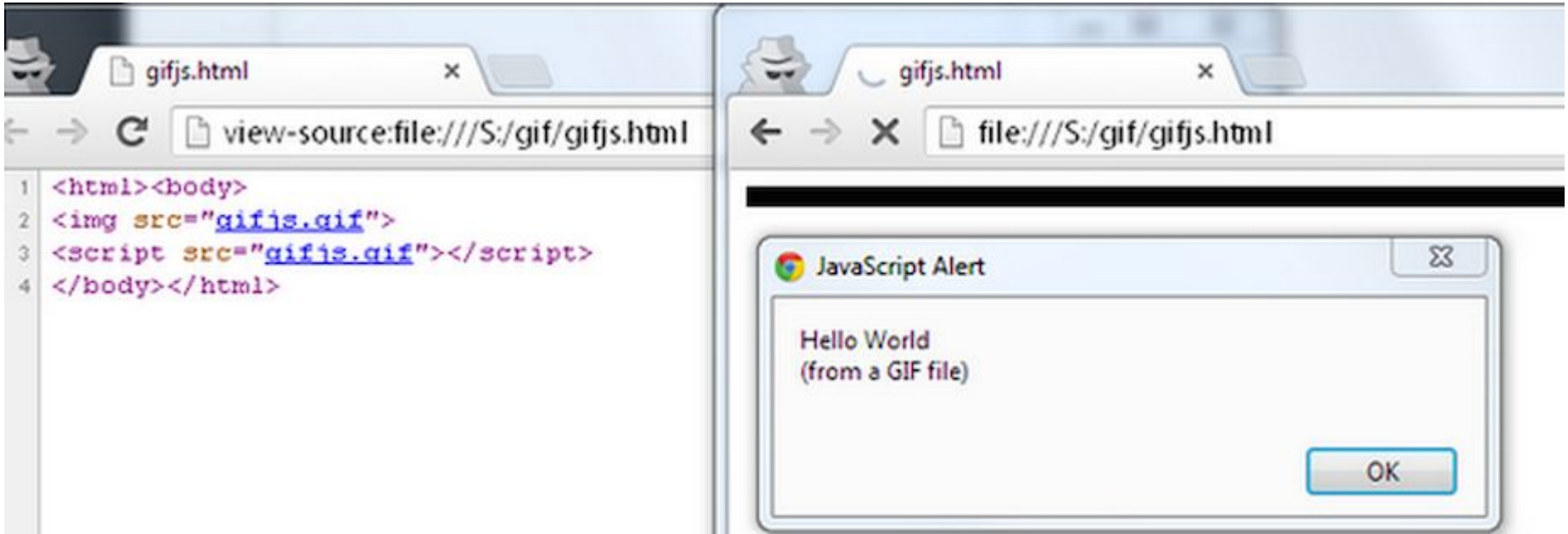
  Ex. BM/*...(BMP image data) …*/=”pwned”;alert(“427: Hacked”);

Polyglot

<img src = “evil.bmp”> Treats the bmp image as an image

But

<script src = “evil.bmp”> Treats the image as code



Problem

- Web browsers or plugins can sanitize image by detecting if there's a Javascript comment in the image

More Sophisticated Technique

Encode Exploit Code into the image

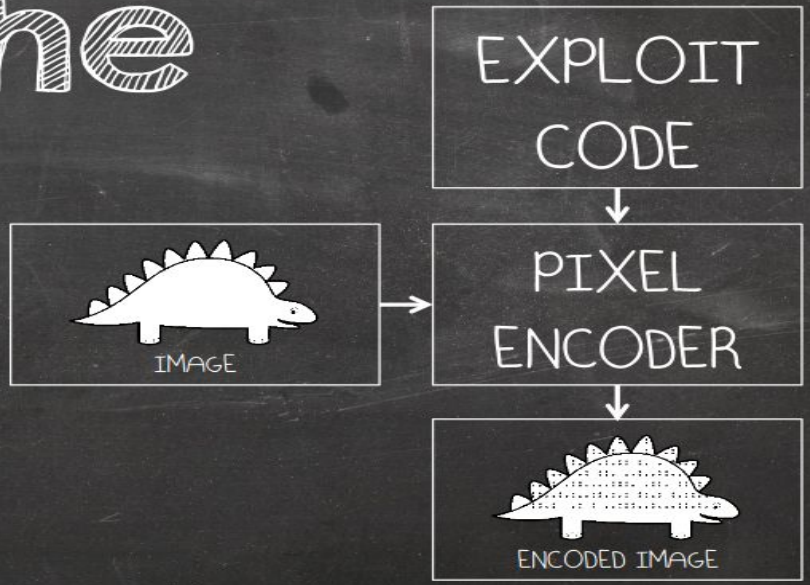
High-Level Explanation

- Encode characters of the code among the image's least-significant bits
  - Exploit code is hidden in the picture

- Use JS to reconstruct code from image and execute
  - Decoder is embedded in the image but not in the pixels

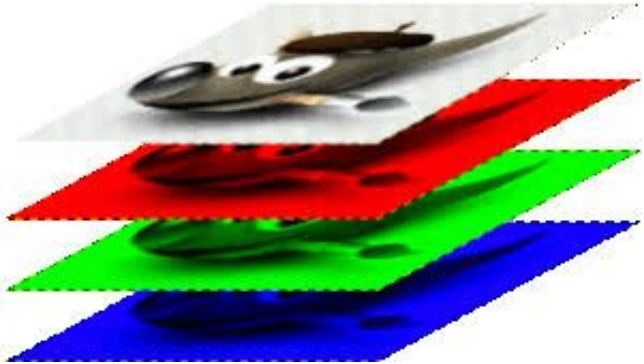
Step I.

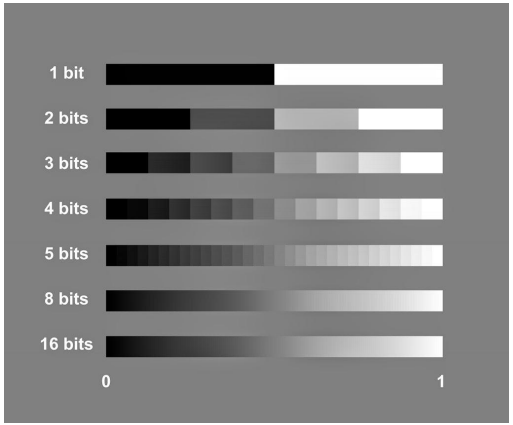
Hiding the Exploit Code in the Image



Images

- Images have several channels: RGB (sometimes you have Alpha - transparency value)
- Each pixel in a layer has 256 possible values (0-255)
  - 8 bit value ($2^8 = 256$)





Least Significant Bits Visualization

Color pixels consist of 3 bytes - RGB.

Orange: R - 11111111

G - 01111111

B - 0000000

Color

	Binary	Octal	**Decimal**	**Hexadecimal**
Red	11111111	377	255	ff
Green	01111111	177	127	7f
Blue	00000000	0	0	0

LSB (last 4 bits) don't affect color appearance.

We can replace these bits with the *first* four bits of another color (blue) to effectively mask it

Blue: R - 10011001

G - 11001100

B - 11001100

Color

	Binary	Octal	**Decimal**	**Hexadecimal**
Red	10011001	231	153	99
Green	11001100	314	204	cc
Blue	11001100	314	204	cc

Least Significant Bits Visualization

R: 1111*1001*
G: 0111*1100*
B: 0000*1100*

The combined result is a second color hidden in the LSB of the first.

We can use this method to hide code behind images

Color

	Binary	Octal	**Decimal**	**Hexadecimal**
Red	11111001	371	249	f9
Green	01111100	174	124	7c
Blue	00001100	14	12	c

B&W Image - Decomposed to 8 Layers

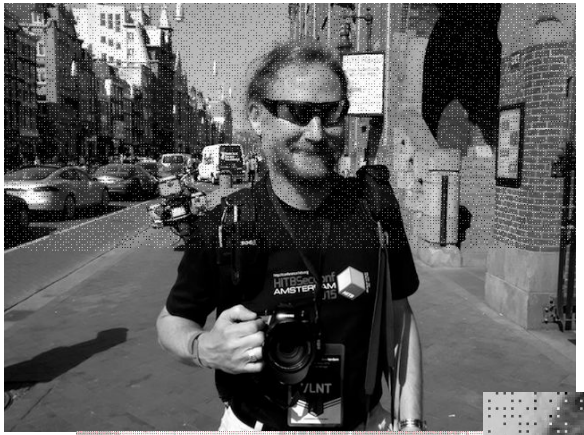
Bit Layers

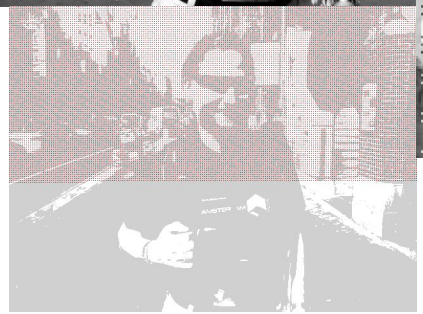
- Recall that in Greyscale, Black is 0 and white is 255
- Layer 0 is the Least significant bit of the pixel
- Layer 1 is the 2nd least significant bit
- …
- Layer 7 is formed by values of the most significant bit of all the pixels

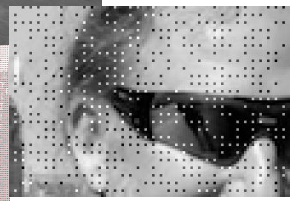
What do you observe?

Why is Bit Layer Important?

- The lower the bit layer, the less details we get
  - We can write exploit code to the lowest bits of the image (lowest layers) to minimize distortion







What do you observe?

Layer 7

Layer 0

Image Format

- Exploit code is converted to byte stream
- Can encode exploit code in either jpg, or png
  - png easy to encode: lossless compression
  - jpeg is harder to work due to compression being lossy (i.e. quality goes down)
    - A pixel may be approximated to its nearest neighbor - bad for steganography
    - Compression may lead to change of exploit code

Low compression (high quality) JPEG

High compression (low quality) JPEG

Jpeg Compression Solution

- Iterative Encoding to achieve error-free encoding
  - Encode and compress the image till all the exploit code is encoded without errors (what causes error?)
  - Higher level of encoding => the lower the compression
    - Max quality of encoding still has some loss
- **NOTE:** When encoding, you do not want to encode byte stream exploit code in consecutive pixels
  - To minimize pixel approximation, encode in a grid (i.e. encode on every nth pixel in row and column)
    - I.e. 3x3 or 4x4 grid pixel
    - Bigger grid size does not lead to lower errors

Watch the following video if you are interested in Discrete Cosine Transformation
https://www.youtube.com/watch?v=Q2aEzeMDHMA

Iterative Encoding - Diagram

- I: source image
- M: message to encode
- ENCODE: Steganographic encoder
- DECODER
- b: number of bit layer (0-7)
- J: jpeg encoder

while delta != 0:

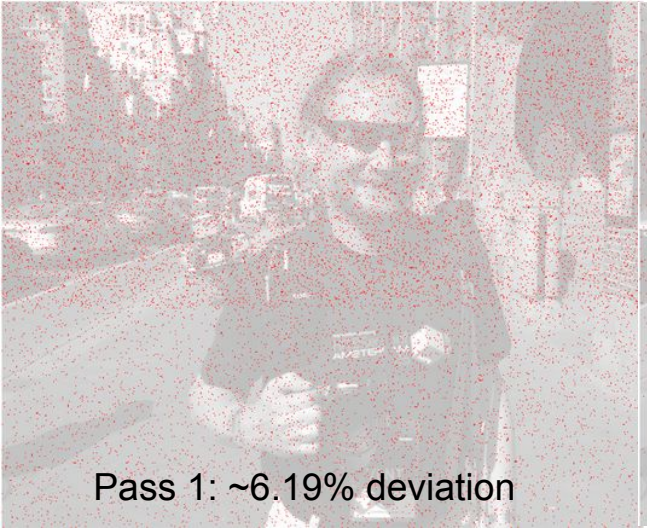
I’ = JPG(ENCODE(I, M, b)) #compress the modified image

M’ = DECODE(I’, b) #Decode the code from the compressed image

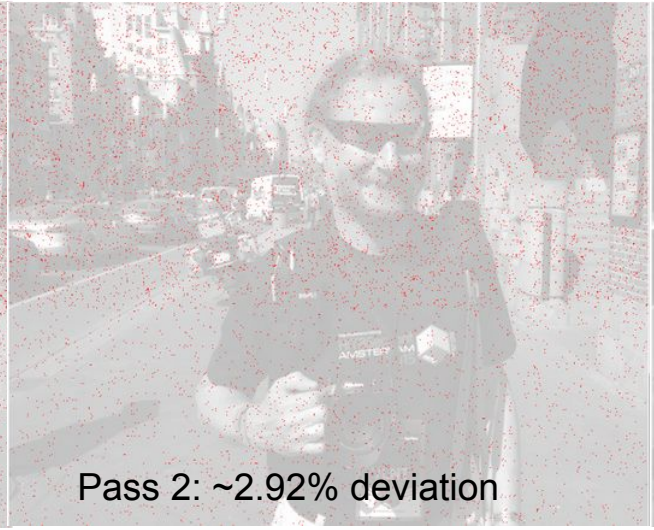
delta = M - M’ #find the difference between the original and decoded code

I = I’

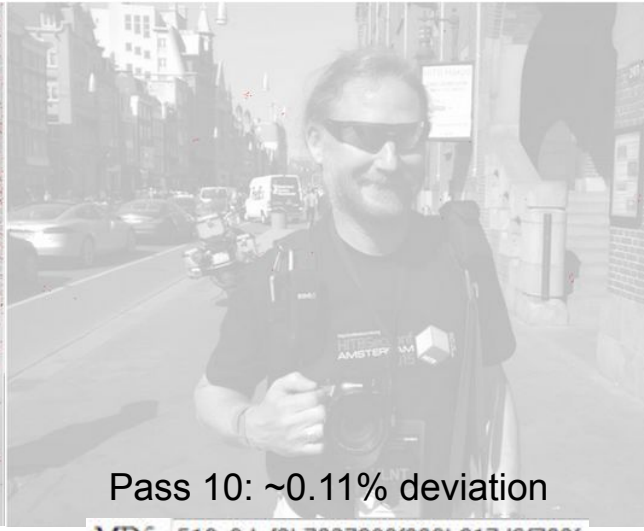
- If delta’ < delta, then we can assume convergence will occur after N iterations
  - Possible for delta to never converge to 0
  - What does it mean for delta to be different? MD5 Hash of the messages don’t match



Pass 1: ~6.19% deviation



Pass 2: ~2.92% deviation



Pass 10: ~0.11% deviation

After 10 passes, exploit code was successfully encoded to the image

Original hash: MD5: 516c9def8b7207299f939b617d3f788f

MD5: 516c9def8b7207299f939b617d3f788f

Why do we not care about the deviation after 10 passes?

STEP 2.

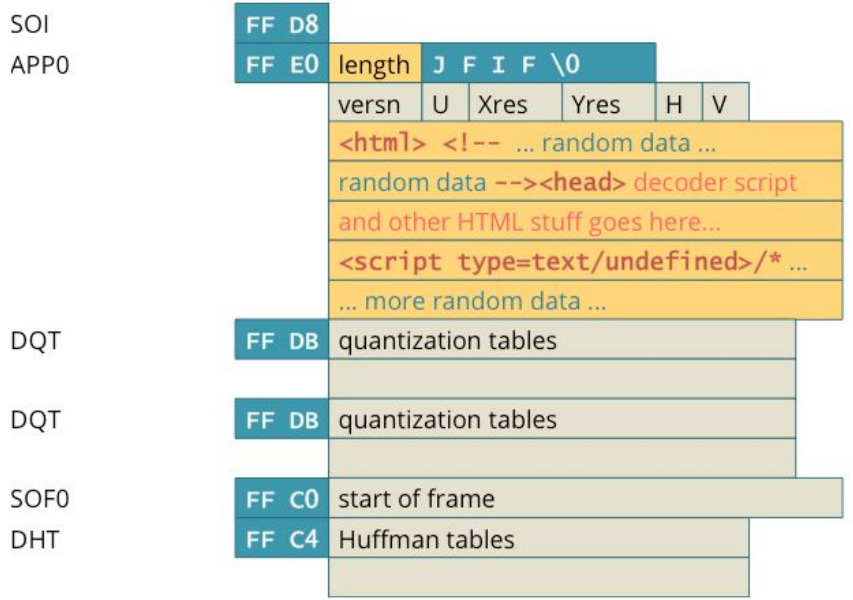
Insert Decoder into Image

How to place decoder to the image

- Need decoder to extract encoded code from the image and execute the code
- Exploit code is encoded into the image pixels
- Decoder is not encoded into the image pixels but in the “header” of the image before the actual image data (pixels)
- Making the image a polyglot when we add the decoder

JPG HEADER

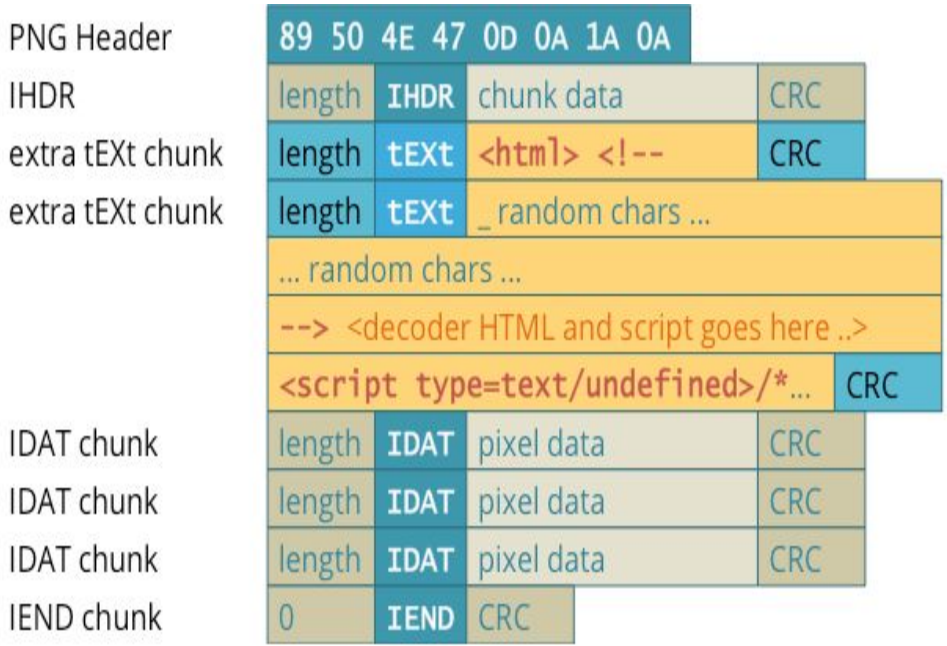
- Change the length
- Insert decoder in **APP0** segment



Marker	Code	Name
FF D8	SOI	Start Of Image
FF E0	APP0	JFIF File
FF DB	DQT	Define Quantization Table
FF C0	SOF	Start Of Frame
FF C4	DHT	Define Huffman Table
FF DA	SOS	Start Of Scan
FF D9	EOI	End Of Image

PNG HEADER

- Easier to add HTML Decoder
- Take advantage of **tEXt chunks** (used to contain metadata)
- Insert Decoder after IHDR

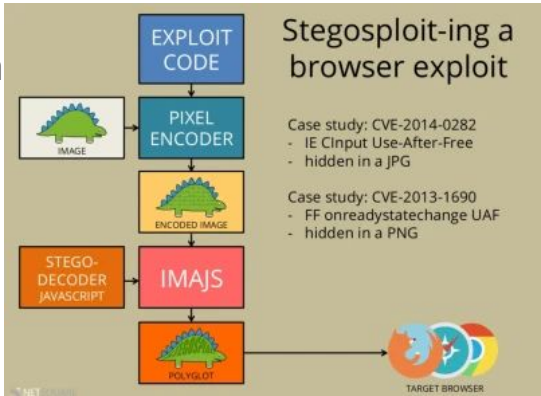


DEMO

- PNG popup
- Run xxd -g1 and show the decoder

How to Decode Exploit Code from Image

- Encoding and decoding through HTML5 Canvas
- Canvas allows drawing graphics via Javascript and HTML
- STEPS:
    1. Load HTML containing the decoder Javascript in the Browser
    2. HTML Decoder loads image carrying steganographically encoded exploit code
    3. Javascript Decoder creates a new canvas element
    4. Pixel data from image is loaded into the canvas (destroy parent image from DOM - Document Object Model)
    5. Decoder reconstructs exploit code bitstream from the pixel values
    6. Exploit code is reassembled into JS Code from decoded bitstream
    7. Exploit code is then executed as JS (via eval or some other form)



Stegosploit - Delivering to Target Browser

1. Host a webserver with malicious image

2. Upload image on 3rd party websites

Difficulties of Stegosploit

- Not scalable: different browsers will render/process images differently
  - Exploit code encoded using Firefox may not work on Internet Explorer
- Need to somehow get the images uploaded to a web server
  - Perhaps via social media or breaking into the server
- Need to ensure web servers don’t corrupt the encoded exploit - many big social media services encode images
- Depending on the exploit, you may need to change the HTTP request to serve the image as http or require some modifications to the webpage

Mitigation

- Stegosploit is a “drive-by” browser exploit, meaning victims just have to load whatever image contains malicious code to be affected.
- Client side: NoScript - Limiting scripts to a whitelist
- Server side: Transcoding - Taking JPGs, converting them to a lower quality PNG, and then back to JPG. Compression damages the encoding. Or resize the image
- If original image hash is known, can compare the hash of the two images
- Have browsers be strict on standards and reject invalid code
    - Have browsers parse data (i.e. code and images) like a compiler

Using Existing Exploits with Stegosploit

- Firefox 3.5 Font Tags Buffer Overflow (2009-2478)
- Exploit:
  - Heap Spray Vulnerability that affects Firefox 3.5 on Windows
    - **Heap Spray:** “Spray” the heap with a large string that also contains some shell code
      - Sounds Like Buffer Overflow: It sounds similar except buffer overflow requires attacker to write more than the buffer can hold while heap spraying targets a certain section of memory and starts “spraying” the string around the heap (increases chance of exploit working)
  - Remote Code Execution to get a shell
- Internet Explorer 8/9/10 CInput Use-after-free POC
  - Use After Free Crash Vulnerability: attempt to access memory after it has been freed
    - Can potentially allow arbitrary code execution
  - Can launch Windows shellcode

Overview

1. Encode the exploit into the lower bit layers
2. Add the decoder onto the image in the headers (makes image a polyglot)
3. Send image to the network

Stegosploit is...

not a 0-day attack with a cute logo
not exploit code hidden in EXIF
not a PHP/ASP webshell
not a new XSS vector

Stegosploit is ...

"Browser Exploits Delivered as Pictures."

Malvertisements

Bringing Social Engineering*, Malware, and Steganography together

Malvertising

Malicious Advertisements - uses legitimate online advertisement network to distribute malware with little to no user interaction.

Utilizes steganography and browser exploits.

Two methods of infection: active and “drive-by”

Interactive: requires user to click a link or interact with the ad

Drive-by: loading the web page hosting the ad launches malicious script and redirects you to an exploit landing page

How are malvertisements delivered?

Ads are fetched in bundles (image + tracking code). Code to execute the hidden instructions in the image can be paired with the tracking code and passed through publishers undetected.

Many sites rely on third parties to supply ads, where issues with security arise.

Once the ads are displayed on the hosted site, the script decodes the malicious code in the image pixels and attempts to infect connecting devices

Attack Vectors

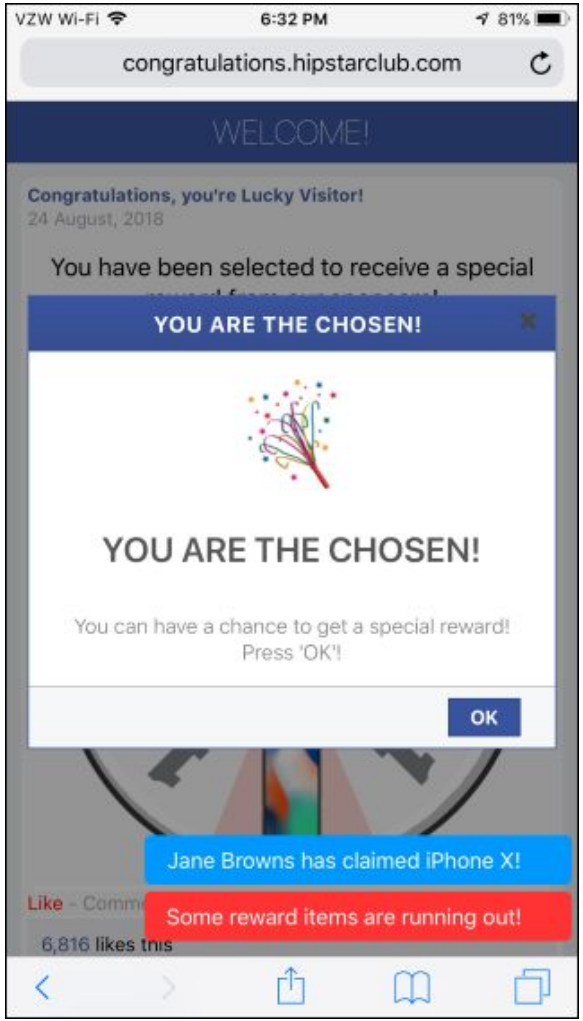
Drive-by method: Exploit API calls for plugins
- DownloadAndInstall API of the Sina ActiveX
Buffer Overflow
- write shellcode into memory.

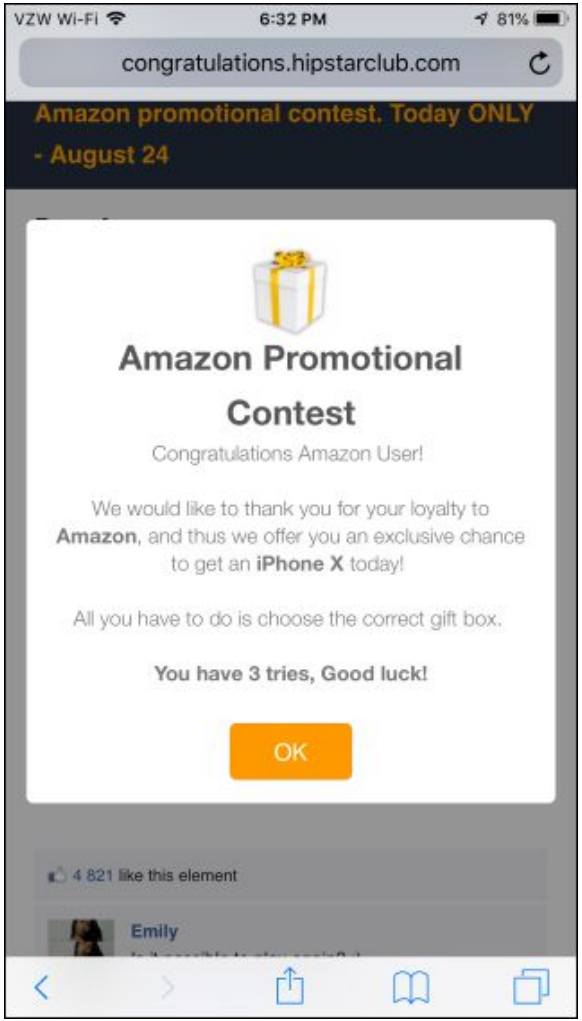
Purpose: Gain download/upload capabilities so they can upload malware or steal information without the user's knowledge or participation.

Attack Vectors

Click Method: Most common method. Attackers host the malicious ad on a legitimate website which redirects the browser to load code from a second site, the *exploit server*.

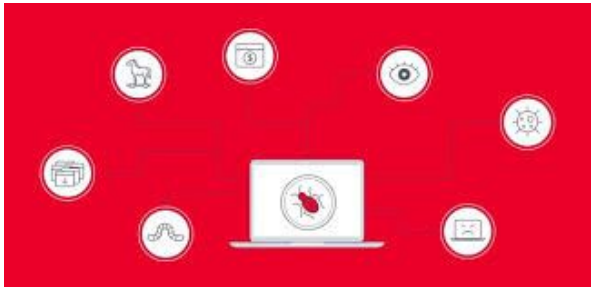
Redirects: Use HTTP headers to determine browser, version, plugins, etc and which exploit to use. Chain of infection. Also used in MDNs (malware distribution networks)


VZW Wi-Fi
6:32 PM
81%
congratulations.hipstarclub.com
WELCOME!
Congratulations, you're Lucky Visitor!
24 August, 2018
You have been selected to receive a special
YOU ARE THE CHOSEN!
YOU ARE THE CHOSEN!
You can have a chance to get a special reward!
Press 'OK'!
OK
Jane Browns has claimed iPhone X!
Some reward items are running out!
Like
6,816 likes this


VZW Wi-Fi
6:32 PM
81%
congratulations.hipstarclub.com
Amazon promotional contest. Today ONLY
- August 24
Amazon Promotional
Contest
Congratulations Amazon User!
We would like to thank you for your loyalty to
Amazon, and thus we offer you an exclusive chance
to get an iPhone X today!
All you have to do is choose the correct gift box.
You have 3 tries, Good luck!
OK
4 821 like this element
Emily

Various Malware Payloads

- Ransomware - Makes your system or files inaccessible
- Spyware - Observes user activities without permission
- Adware - Spams you with advertisements (can be malicious advertisements)
- Virus
- Malicious Cryptomining - uses your system’s resources to mine coins



Current prediction of damage done by Malware in 2021 is $6 trillion USD (about 3.5 times the Canadian GDP)

Examples of Malvertisements

- **Malware within video** — video players do not protect against malware. For example, a standard video format called VAST contains pixels from third parties, which could contain malicious code. Videos can infect users by displaying a malicious URL at the end of the video.

- **Malware on a landing page** — even on legitimate landing pages served by reputable websites, there may be clickable elements that execute malicious code. This type of malware is particularly dangerous because users click an ad, land on a real, legitimate landing page, but are infected by a malicious on-page element.

Video viruses: embedding hyperlinks (https://www.opswat.com/blog/can-video-file-contain-virus)

Note: landing page: **“**the section of a website accessed by clicking a hyperlink on another web page, typically the website's home page”

Impact

Spotify

NY Times

NFL

Yahoo

Attacks:

- Ransomware
- Financial Information Theft

Mitigation

Action required by both end-users and publishers

End-Users: Antivirus and Anti Malware software
Ad-Blockers
Updating browsers and plugins
+Frequently remove browser cache
+Don’t save passwords in the browser

What we learned

How images are structured (headers, bit layers)

How to hide code behind images

Encoding:

- Why encoding JPG is harder than PNG
- Where to encode

Where to add the decoder

Practical applications via malvertising, including methods and attacks.

Mitigation

References

https://stegosploit.info/
https://www.alchemistowl.org/pocorgtfo/pocorgtfo06.pdf
https://www.blackhat.com/docs/eu-15/materials/eu-15-Shah-Stegosploit-Exploit-Delivery-With-Steganography-And-Polyglots.pdf
https://tunnelshade.in/blog/2015/06/stegosploit-fun/
https://books.google.ca/books?id=-GP5TKE_dmgC&pg=PA54&lpg=PA54&dq=DownloadAndInstall+API+of+the+Sina+ActiveX&source=bl&ots=F8-e-HBXn7&sig=ACfU3U1wBLJjkwUIGtBk98wdSxf6Ay6_ag&hl=en&sa=X&ved=2ahUKEwiw573JhZDoAhUJCM0KHe-HBCsQ6AEwAHoECAoQAQ#v=onepage&q=DownloadAndInstall%20API%20of%20the%20Sina%20ActiveX&f=false