How to insert javascript code in the image

GIF and BMP files

Here's what a regular gif and bmp file looks like in a high level detail in a text editor...
HEADER
IMAGE_DATA

On a bmp the header is BM
On a gif the header is GIF89a

To insert javascript code here, you have to do this
HEADER/*
IMAGE_DATA*/=<some arbitrary value>;
<Your javascript here>

If you treat it as a script with the <script> tag in html, javascript will see this:
HEADER=”string”;
<Your javascript here>;

1. Download any static gif file and any bmp file from the internet. The recommended BMP file to use in this tutorial is (tinyurl.com/y8zx2k52)
    P.S. Make sure it's appropriate for instructors to view
2. Create a html file to run your hidden code, the template for the html file be as follows

```
<html>
    <body>
        <img src="put the path of your bmp image here">
        <script src="put the path of your bmp image here"></script>

        <img src="put the path of your jpg image here">
        <script src="put the path of your jpg image here"></script>
    </body>
</html>
```

3. Save this file as index.html somewhere in the $UTORID/www folder so you can view your webpage via cslinux.utm.utoronto.ca/~$UTORID
4. When you view your webpage through a modern data browser, the console will log an error message and won’t run your script, this is ok
5. Download the current version of the Tiny Core Linux (http://tinyurl.com/y9zurs9u) or from the school computers by:
    $UTORID@dh2020pc01.utm.utoronto.ca:/virtual/csc427scapy/TinyCore-8.2.1.iso

6. Create a new virtual machine with vmware with the iso BUT make sure the set the memory (RAM) size to >512 mb otherwise running a webbrowser will be terribly slow
7. In Tiny Core, go to app installer and install "opera9.tcz".
8. Open up opera and go to cslinux.utm.utoronto.ca/~$UTORID and view your html file
9. If your javascript doesn't execute, try using a different image, not all images work!

JPG and PNG files

We can make jpeg/png polyglots that are viewed as html files at the same time refer to themselves as image files when then decode and execute javascript.

Step 1: Extract the stegosploit toolkit from https://stegosploit.info
HINT: Check section 7 of the POC || GTFO pdf for more info on how to extract the toolkit
Step 2: cd into the stegosploit toolkit folder and run the command:
python2 -m SimpleHTTPServer
Step 3: To access the toolkit, go to localhost:8000 on a web browser
Step 4: Go to stego/iterative_encoding.html and encode javascript into any jpeg/png you want
NOTE: Please take into attention the bit layer and the grid size is important to remember when decoding the javascript from the imajs file
Step 5: Now that we have encoded our javascript into our image file, should implement a decoder of some sort to execute our imajs
Step A: Open up the file exploits/decoder_cve_2014_0282.html and this is our html decoder for imajs files, take heed of the variables var bL=2,eC=3,gr=3
bL: is bit layer and we want the value to the be the same as the bit layer we encoded our image file
eC: is the color encoding channel we are decoding from the image:
0 = red
1 = green
2 = blue
3 = all channels
gr: Is the same pixel grid we specify in when encoding our imajs
Knowing all of this, you want the decoder to have the same bL, eC, and gr values as when we encoded our imajs earlier
Step 6: Now we have set up our decoder variables, depending on our image file, we want to run one of the 3 html_in_* perl scripts located in the imajs folder.
All three of the files take in the following arguments:
html_in_*.pl <our decoder html file> <our imajs file> <new imajs file with decoder>
Step 7: Assuming that everything worked, renaming the file extension of our newly created file to .html and loading it on a web browser will result in our exploit being executed.