Steganography Tutorial

URL: https://tinyurl.com/uwgonan

If tutorial file is not on the lab machines, you can download from here as well(only valid till 10pm Tuesday):
https://send.firefox.com/download/4c7698d2aa5ab1fc/#zMgpeMdtHhOR6z7l-gJ9iA
Password: csc427

There are two main objectives of the the tutorial:

1. Modify a BMP
2. Extract the Stegosploit toolkit from the PDF
3. Encode an exploit to a png image and add the decoder

The tutorial should be straightforward and fun. The assignment will be more theoretical.

Files to submit:

- image.bmp
- bmp.html
- stegosploit-tools.7z
- payload.html
- polyglot.png

SETUP: You could try not using a VM - We know it works on Kali
Please open **vmware** with **Kali Linux**
If you don't have a Kali image, you can follow the following instructions
mkdir /virtual/$USER
cp -r /virtual/csc427/kali/ /virtual/$USER/kali

Part 1: Create a malicious BMP
Recall from lecture:

```
FILE FORMAT:
42 4D XX XX XX XX 00 00 00 00 ........
```

```
B   M Filesize    Empty Empty DIB data
```

1. Choose a bmp image of your choice or choose the bmp image provided (not all bmp images work)
2. Create an HTML file named **bmp.html** with the following (change the image name):

```
<html>
<body>
<img src = "img.bmp">
<script src = "img.bmp"></script>
</body>
</html>
```

3. Let's choose the payload to be an alert.
   Recall in Javascript, an alert is alert("Text");

4. Modify the header to comment out the image
   Recall that we want to modify the first two values in Filesize with "/*"

   Use your editor of choice (i.e. vim) to write in ASCII or you could use **bless** to modify the image with your payload in Hex

5. Write the payload (refer to slide 10 of the lecture) at the end of the bmp image

6. Run on Firefox provided in the lab (we are using Firefox 10 for this exploit):

```
chmod +x firefox
./firefox
```

7. Open the **bmp.html** on firefox

Part 2: Retrieve stegosploit tool from the pdf
The pdf is an example of steganography, hiding files in a normal looking pdf. Imagine what files could be hiding in the internet

```
unzip pocorgtfo08.pdf stegosploit_tool.png
```

Your goal is to extract the toolkit
Hint: Look at the image extracted from the pdf.

NOTE: You will need this for the assignment!!!!

Part 3: Encode an exploit to an image and add the decoder

1. Install pip2:

```
sudo apt-get update
sudo apt-get install python-pip
```

2. Skip this if already installed: Install Pillow

```
pip2 install Pillow
```

3. Create **payload.html** with some Javascript code to encode (i.e. alert)
4. Choose a PNG image (Choose a small png image)
5. Run the program that will encode the Javascript code to the image and also add the decoder to the image

```
python2 html_png_polyglot.py -i image.png -p payload.html
-o polyglot.png
```

6. Run the webserver:

```
python2 server.py
```

7. Visit the webpage: localhost:8000/polyglot.png
8. Look at the hex code and output it to png_hex.txt

End of Lab: Please look at above for a list of files to submit

The solutions to the lab will be posted after the submission

Below are some possible assignment questions for any who are curious.

Potential Assignment Questions

Not an exhaustive list of questions. More to come.

TO ARNOLD: I'll email you the final draft of questions with answers. These are just suggestions I thought of while preparing the tutorial and lecture. I'll have to try the questions myself to judge the feasibility and whether or not the questions are covered in lecture or not.

Files To Submit:

- answers.txt
- layer7.jpg
- layer3.jpg
- encode.png
- polyglot.png

Office Hours:
Kim: I am at school 7 days a week for about 10-12 hours a day in one of the labs or in DH Silent room, come find me and ask questions if you are stuck. Alternatively, you can email me: juhong.kim@mail.utoronto.ca, I should respond within a day.

Part 1: Conceptual Questions
Here are some warm up questions to start before working on the toolkit

1. What are some factors you need to consider when choosing which bit layer you wish to encode?
2. Iterative Encoding: How many passes does PNG require and why
3. Iterative Encoding: On slide 24, we say delta = M - M'
   What does that actually mean? Please explain in words. In addition how do we know when delta = 0 on the toolkit when we run the iterative encoding. (Hint: Try running iterative encoding on a jpg or look at slide 25)
4. What is a good indication that the iterative encoding will converge (hint look at slide 25)
5. Iterative Encoding: Why choosing the lowest layer may not be the best for JPG encoding
6. Is JPG encoding cross browser support? And why?
7. Why would a large code exploit not be able to be encoded into a small image?

From tutorial in **Part 2**, you needed to extract the toolkit, you will be using the toolkit for the next following questions

Part 2: Encoding

To start off, you will need to start the server to run any of the tools
Make sure you are in the root folder of **stegosploit-tools**

```
python -m SimpleHTTPServer
```

Step 1: Encode in different layers (JPG)
On your web browser, visit **localhost:8000/stego/iterative_encoding.html**

Note: You will need to write the path of the image. The path is relative to the webserver
Note: Make sure the image is a **jpg** and **not jpeg** (they are essentially the same file format but the program will not be able to recognize the image)

1. Encode a jpg file in a channel of your choice but in a grid size of 3 or 4 but with a layer 7
2. Choose the exploit code to be IE C-Input Exploit
3. Click on the **Process** button
4. Choose the option for slow and click on the iterate button
5. Rename the image as **layer7.jpg**
6. Write down the number of passes it went through to encode
7. Refresh the page
8. Repeat steps 1-5 but on a layer 3 instead and call the file **layer3.jpg**

Step 2: Encode a PNG File

1. On the same page as part 1, encode a png file on a layer that doesn't have any noticeable distortion. **Ensure to refresh the page before starting.**
2. Save the image as **encode.png**
3. Write down the number of iteration

Part 3: Add Decoder to the encoded image

1. Modify **exploits/decoder_cve_2014_0282.html** and ensure the following variables are set correctly as those you set when you did encoding:
   bL: the bit layer where the code was encoded to the image
   eC: is the channel we are decoding from
   0: red
   1: green
   2. Blue
   3: all channels (use this for grey scale images)
   gr: the grid size
2. Run **html_in_png.pl** under **imajs** directory to add the decoder to the

image

```
perl ./imajs/html_in_png.pl exploits/decoder_cve_2014_0282.html
./encode.png polyglot.png
```

3. View the file and observe that you can see the decoder in the image but not the malicious code
4. **OPTIONAL:** Copy the file and place it on the provided Windows VM (if provided) with Internet Explorer 9 and see what happens :D

Answers will be posted after the submission deadline for the assignment.