Lab10 - The Downfall of SHA1

Intro

We discussed in class the SHA-1 collision that was discovered in February 2017. The team at Google had planned to wait 90 days after their paper and announcement before they released their code for others to try it. However, the Internet beat them to it. Two days after the announcement a collider program was available online. You are going to use one of these programs to create a SHA-1 collision of two files. The collider program you will be using was written by Robert Xiao of Carnegie Mellon University. It is available on Github. You will have to take a **screenshot of the SHA-1 collisions you generate to include in your lab report. Steps H and M** are where you should take your screenshots. And, you will include **an answer to step N** in your lab report**.** You will also **upload the four out-pdf files you create as described in steps G and L.**

Part 1

a. Log into ISERink (iselab01.ece.iastate.edu) and onto your Ubuntu desktop.
```
sudo apt-get install libjpeg-progs
sudo apt-get install git
cd ~/Downloads
git clone https://github.com/nneonneo/sha1collider.git
sudo apt-get -y install python3-pip
sudo python3 -m pip install Pillow argparse
```
b. Open LibreOffice Writer and write two different documents. One would be considered a "good" letter. For example something like a recommendation letter for someone. The other one would be considered an "evil" file. The example would be taking that recommendation letter and slamming that person ensuring he/she would not get the job (he\she is slothful, doesn't bathe, plays on his/her cell phone while at work, etc.) The only requirement on the two files is that they need to be the same number of pages.
c. Save the LibreOffice files as pdf files.
d. Run the pdfs through the SHA-1 Collider program. **Upload the two out-filename.pdf files to Canvas.**
   i. `python3 collide.py --progressive <filename1.pdf> <filename2.pdf>`
e. Open up the two out-filename.pdf files, and ensure that you can still read the message. If the files are corrupted, try again with different messages.
f. Hash the two files using sha1sum and sha512sum. **Take a screenshot to include in your lab report.**
   i. `sha1sum <out-filename1.pdf> <out-filename2.pdf>`
   ii. `sha512sum <out-filename1.pdf> <out-filename2.pdf>`
g. Now you will try it with images. Using Google to find two jpeg images that are the same size. You can use Google Advanced Image Search or you can use a

request for a specific resolution adding imagesize:200x200 to your Google search string. Do not select high resolution images. The collision program will error if the files are too large.

h. `sudo apt-get -y install imagemagick`
i. Edit `/etc/ImageMagick-6/policy.xml and remove the line <policy domain="coder" rights="none" pattern="PDF" />`
j. `convert img.jpg img.pdf`
k. Run the pdfs through the SHA-1 Collider program to create out-filename1.pdf and out-filename2.pdf. **Upload the two out-filename.pdf files to Canvas.**
   i. `python3 collide.py --progressive <filename1.pdf> <filename2.pdf>`
l. Open up the two out-filename.pdf files, and ensure that you can still see the images. If the files are corrupted, try again with different images.
m. Hash the two files using sha1sum and sha512sum. **Take a screenshot to include in your lab report.**
   i. `sha1sum <out-filename1.pdf> <out-filename2.pdf>`
   ii. `sha512sum <out-filename1.pdf> <out-filename2.pdf>`
n. So, now that you have created hash collisions for these two sets of files how likely is this to be used to carry out an attack in the wild? Please give me a clear example of when it could be used in the wild or when it would not work in the wild to support your argument. **Include your answer to these questions in your lab report.**

Part 2

Iowa State has decided to make a lottery (not actually) to raise funds for scholarships. The lottery works in the following way: On odd numbered days, a lottery generator is run that produces a winning lottery ticket for that day and the next day. (i.e, On October 29th a lottery is number is generated for October 29th and October 30th). You have been given the code that is used to generate these “random” winning tickets. Use the information provided in the template to answer the following: **Assuming that the winning lottery ticket on November 11th is 5-4-3-1-2-9-1-4-2-5, what is the winning lottery ticket on November 12th, 2022?**

For example, assuming the winning ticket on October 1st is 7-8-9-9-3-6-8-3-5-8, the winning ticket on October 2nd is 6-3-9-3-9-3-5-0-2-8.

Download the template code from repo.331.com

```
cpre331@desktop:~$ scp repo@repo.331.com:/home/repo/lab10/template.py .
repo@repo.331.com's password:
template.py                                   100% 1275     3.0MB/s   00:00
cpre331@desktop:~$
```

Lab 10 Template

Part 1

1) **“Good” text pdf (Include in canvas submission)**
   (10 points)

2) **“Bad” text pdf (Include in canvas submission)**
   (10 points)

3) **Screenshot of the hashes of the text PDFs colliding for SHA1 and differing with SHA512**
   (10 points)

4) **Image out-1.pdf (Include in canvas submission)**
   (10 points)

5) **Image out-2.pdf (Include in canvas submission)**
   (10 points)

6) **Screenshot of the hashes of the image PDFs colliding for SHA1 and differing with SHA512**
   (10 points)

7) **Answer questions in part p**
   a. **How likely is this to be used to carry out an attack in the wild?**
   (10 points)

Part 2

1) **Upload your code to canvas as a <u>seperate .py file</u>**
   (10 points)

2) **What is the winning lottery number for November 12th, 2022?**
   (10 points)

3) **In your own words, what is the seed for this random number generator? Why is this a good or bad thing? Justify your answer.**
   (10 points)