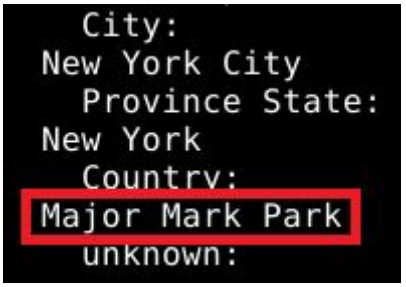
Andrew Chung

CTF Writeups

Lost Park

Given this jpg file, what is the name of this monument?

- A good start would be to check the image file's metadata fields. ***"identify -verbose statue1.jpg"***
- As seen in the image below, the metadata provides us with the name of this monument, Major Mark Park.



Too Many Puppers

Given a zip file full of pictures of puppies, find the hidden flag.

- Although we don't know how the flag is hidden, a good start is to search for the flag in plain-text.
- Use strings to search for all plain-text strings in the zip file and grep to search for the flag string. ***"strings puppy.zip | grep flag"***

```
PxNc/TvNkzcm/UdfV/BVoAp/yHkWSGaL/bBYy/lxANKI/hgugRa/fvgXjIbXg/OSvfOVZ/vSXzvo/FSs
Dewdd/NFFzMKA/flag{0h_y0u've_found_me}PK
```

Brain Game

Given this jpg file, find the flag.

- Checking the image file's metadata fields, ***"identify -verbose brain.jpg"***, we find a base64 encoded string, `Image Description: TjB0X0YxYUdfQnVUX200WV9uMzNk`
- Decoding the string, ***"echo TjB0X0YxYUdfQnVUX200WV9uMzNk | base64 -d"***, we get the plain text, `N0t_F1aG_BuT_m4Y_n33d`
- From here a good approach would be to analyse the image itself to check for any files within the image. ***"binwalk brain.jpg"***
- We find that there is indeed a zipfile within the image. Unzipping this file, ***"unzip***

brain.jpg", requires a password, in which the plain text we obtained before,

`N0t_F1aG_BuT_m4Y_n33d` comes in handy.

```
Archive:  brain.jpg
warning [brain.jpg]:  28650 extra bytes at beginning or within zipfile
  (attempting to process anyway)
[brain.jpg] flag.txt password:
  inflating: flag.txt
```

- Opening flag.txt shows:

```
Condor,

I found what you are looking for, but you know me better than that. You have to
look deeper in order to find the goods. F*CK the system! Hack the planet.

-Kevin M.

P.S. Lay3rz_0f_Obfusc4t10n
```

  - We get a hint that we need to look deeper. Insert ***steghide***. Extracting this file, ***"steghide extract -sf brain.jpg"***, requires a password, in which the plain text we obtained before, `Lay3rz_0f_Obfusc4t10n` comes in handy.
  - Opening realflag.txt shows:

```
Condor,

I know about your love for esoteric programming languages so I put the info
 you're looking for below.

-[------->+<]>-.+++++.----------.>-[--->+<]>-.[----->+<]>++.>--[-->+++++<]>
.+[-->+<]>++++.---[----->+<]>-.[-->+<]>-----.---•-[----->+<]>--.-----------
----•-[----->+<]>+.+[++>-----<]>.[-->+++<]>-.[->++++<]>-.+[--->++<]>++.[->+
++++<]>---•-[->++++++<]>.+++++[->++<]>.>-[----->+<]>.---[->++<]>-•-[----->+
<]>.[++>-------<]>.-------.[--->+<]>+++.


Good luck with everything,
-Kevin M.
```

  - Looks like we're provided with some "info". Googling esoteric programming languages, we find examples of different esoteric programming languages with one that stands out in particular, ***Brainfuck***.
  - Decoding the code, we obtain the flag. ***HMCTF{Br41n_G4M3z_4r3_Fun}***

A Friend

Given a zip file containing a word document, find the flag.

- This was the only exercise that wasn't solved by anyone during the seminar and I spent a fair share of my own time trying to analyse the word document for the flag to no avail. The word document was a red herring...
- I naively extracted the word document without even checking for embedded files in the zip archive.
- Using ***binwalk***, we can see there is a PNG image embedded inside the zip archive.

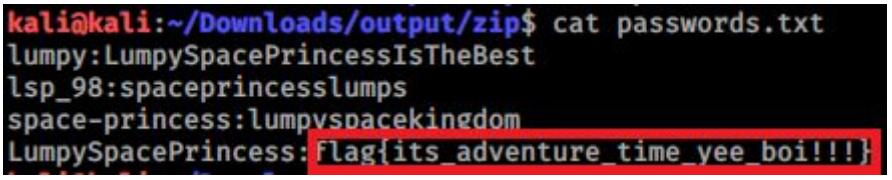
```
kali@kali:~/Downloads$ binwalk a_friend.zip

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Zip archive data, at least v2.0 to extract, compressed size: 12250, uncompressed size: 24626, name: a friend.docx
12404         0x3074          End of Zip archive, footer length: 22
12426         0x308A          PNG image, 5312 x 2988, 8-bit/color RGB, non-interlaced
15889         0x3E11          Zlib compressed data, default compression
```

- Extracting all embedded files from the zip archive, ***"binwalk --dd='.*' a_friend.zip"*** we obtain the flag from the PNG image.

lost flash drive

Given a zip file containing the boot sector for a flash drive, find the flag.

- No problems are encountered unzipping the archive, ***"unzip lost_flash_drive.zip"***, in which we receive the DOS/MBR boot sector for this flash drive. ***"file lost_flash_drive"***
- We can use ***Foremost***, a forensics program to recover lost files to output contents of the drive. ***"foremost lost_flash_drive"***
- Within the dumped contents there is another zipfile which contains ***passwords.txt***.
- Reading ***passwords.txt*** we obtain the flag.



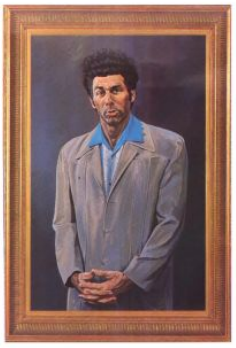
pdfcrypt

Given an encrypted pdf, find the flag.

- We can use ***John The Ripper***, a fast password cracker.
- Generate the hash file of the PDF using the ***pdf2john.pl*** tool. ***"./pdf2john.pl /home/kali/Downloads/encrypted.pdf > /home/kali/pdf.hash"***
- Now that we have the hash, we can pass it in as an argument to ***John The Ripper*** to guess the password of the PDF.

```
kali@kali:~/JohnTheRipper-bleeding-jumbo/run$ ./john /home/kali/pdf.hash
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
Cost 1 (revision) is 4 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:./password.lst
Proceeding with incremental:ASCII
hacked           (/home/kali/Downloads/encrypted.pdf)
1g 0:00:09:10 DONE 3/3 (2020-06-18 03:45) 0.001815g/s 47298p/s 47298c/s 472
98C/s habm9d..hackms
Use the "--show --format=PDF" options to display all of the cracked passwor
ds reliably
Session completed.
```

- Opening the PDF with the password found above, ***"hacked"*** we see a portrait of Kramer with the flag.



flag{kramer_the_best_hacker_ever}

unk

Given an unknown file, find the flag.

- A good start would be to check what the file actually is. ***"file unk"***

```
unk: Zip archive data, made by v4.5, extract using at least v2.0, last modi
fied Mon Jan 26 00:44:48 1970, uncompressed size 1364, method=deflate
```

- Unzipping this file, ***"unzip unk"***, we can see the contents extracted and we now know from the extracted contents of XML files that ***unk*** is a DOCX file.

```
kali@kali:~/Downloads$ unzip unk
Archive:  unk
file #1:  bad zipfile offset (local header sig):  0
  inflating: _rels/.rels
  inflating: word/_rels/document.xml.rels
  inflating: word/document.xml
  inflating: word/theme/theme1.xml
 extracting: docProps/thumbnail.jpeg
  inflating: word/settings.xml
  inflating: word/fontTable.xml
  inflating: word/webSettings.xml
  inflating: docProps/core.xml
  inflating: word/styles.xml
  inflating: docProps/app.xml
```

- From here, a good approach would be to check the thumbnail as it would show the flag if it was not hidden.