A Scuffed CTF Walkthrough

https://github.com/utahsec/damctf23/tree/master

https://tinyurl.com/suy9727x

Problem 1: misc/de-compressed

- We are given one zipped folder: message.zip
- Objective is to get the flag from the zipped folder

Solving

- What files do we see when we unzip?

Solving

- What files do we see when we unzip?
- Is there anything unique about README.txt?

Solving

- What files do we see when we unzip?
- Is there anything unique about README.txt?
- From a cursory glance, README.txt looks like a normal text file
- Are there any hidden files inside the zip?

Solving

- What are some tools we can use to find hidden files?
- Are there any that can help us with hidden files inside a zip?

Binwalk

- https://github.com/ReFirmLabs/binwalk/blob/master/INSTALL.md
- Binwalk is a tool for searching a given binary image for embedded files and executable code. Specifically, it is designed for identifying files and code embedded inside of firmware images. Binwalk uses the libmagic library, so it is compatible with magic signatures created for the Unix file utility.
- On debian-based distros: `sudo apt install binwalk`

Use binwalk on message.zip

- binwalk message.zip
- Output?

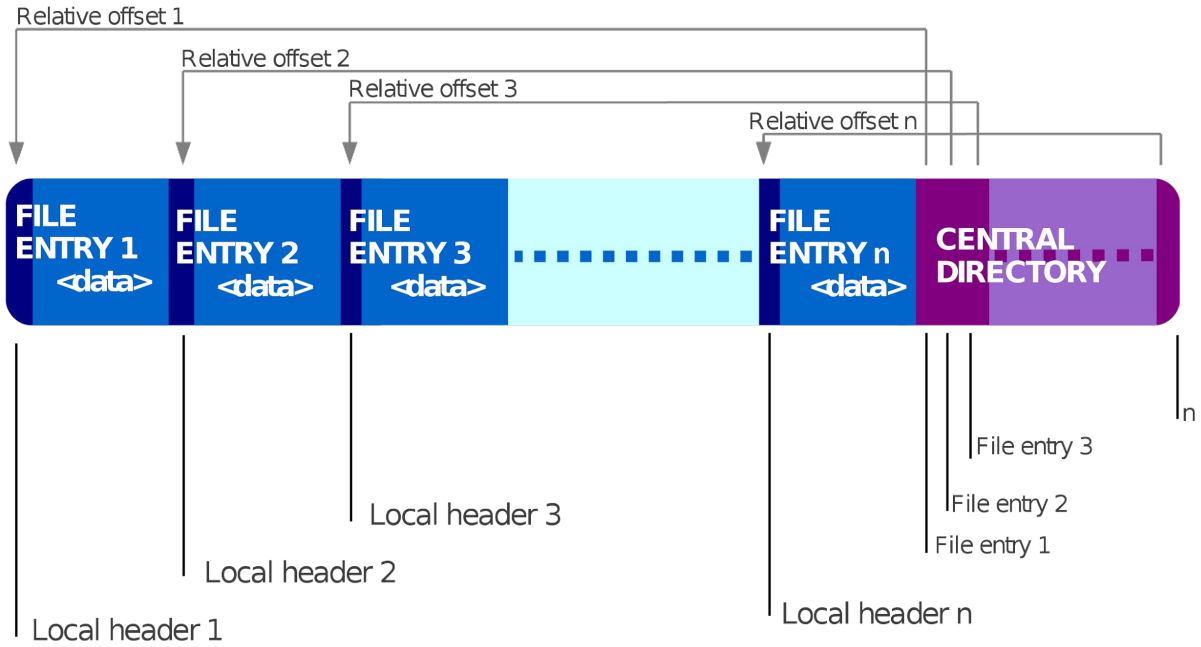
```
→ de-compressed git:(master) binwalk message.zip

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Zip archive data, at least v2.0 to extract, compressed size: 311, uncompressed size: 533, name: README.txt
351           0x15F           Zip archive data, at least v2.0 to extract, compressed size: 407, uncompressed size: 2104, name: secret.txt
854           0x356           End of Zip archive, footer length: 22
```

secret.txt

- How do we get to secret.txt?
- Why does it not unzip?

How Zip Files Work



secret.txt

- secret.txt is missing from the central directory
- is there any other way we can extract the file?

secret.txt

- https://stackoverflow.com/questions/1423346/how-do-i-extract-a-single-chunk-of-bytes-from-within-a-file (https://tinyurl.com/mrx4tcwb)
- dd if=input.binary of=output.binary skip=$offset count=$bytes iflag=skip_bytes,count_bytes

secret.txt

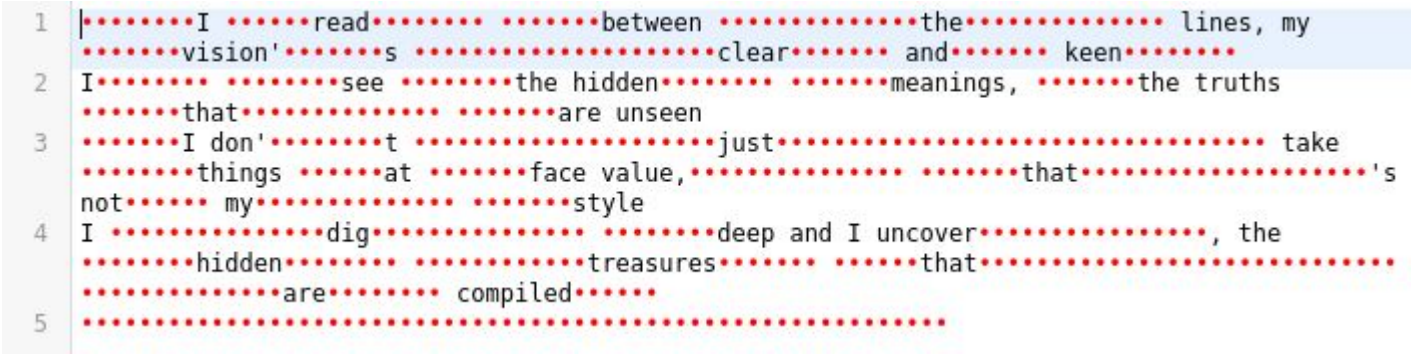
- https://stackoverflow.com/questions/1423346/how-do-i-extract-a-single-chunk-of-bytes-from-within-a-file (https://tinyurl.com/mrx4tcwb)
- dd if=input.binary of=output.binary skip=$offset count=$bytes iflag=skip_bytes,count_bytes
- **dd if=message.zip of=secret.zip skip=351 count=503 flag=skip_bytes,count_bytes**

secret.txt

- A simple text file
-
```
1 I read between the lines, my vision's clear and keen
2 I see the hidden meanings, the truths that are unseen
3 I don't just take things at face value, that's not my style
4 I dig deep and I uncover, the hidden treasures that are compiled
```

secret.txt

- Open it in notepad++



Unicode Steganography

- Paste the text into https://330k.github.io/misc_tools/unicode_steganography.html (https://tinyurl.com/4cnxb23c)
-

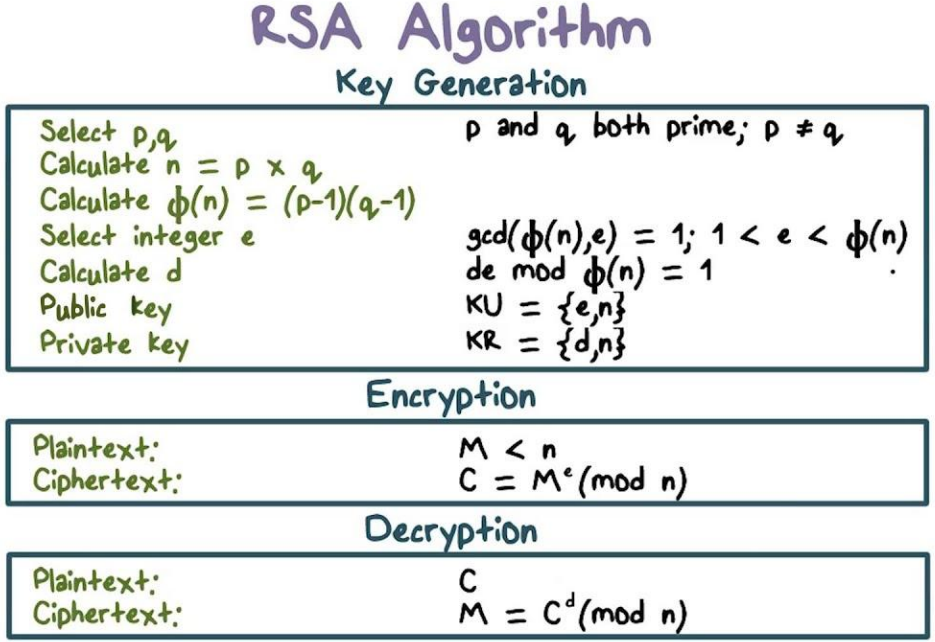
```
Disregard the README, I am still on the team.
dam{t1m3_t0_kick_b4ck_4nd_r3l4x}
```

Problem 2: crypto/crack-the-key

- Given: flag.enc, requirements.txt, super_secure_rsa.py, pub.pem
- We are given an encoded flag (encryption : RSA) + the code used to encrypt + the public key.
- Hint provided: use factordb

RSA: Public Key Crypto

- Based on the impossibility of computing factors of large primes
-



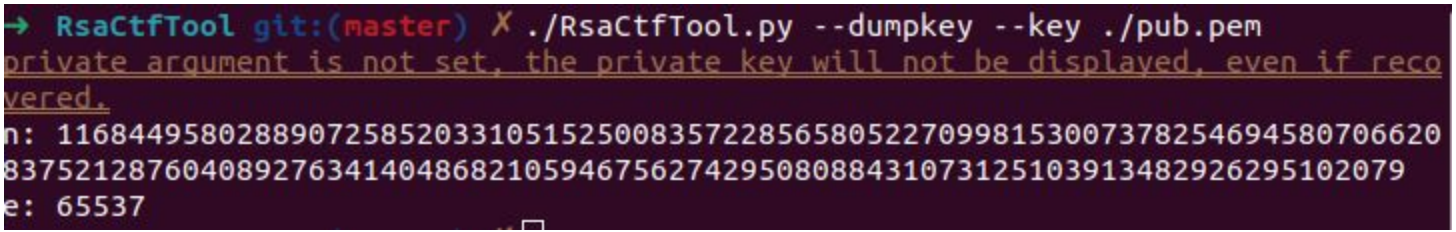
Primes

- If we can factorize n, we can get the private key.
- How do we get n from the public key?
- Using https://github.com/RsaCtfTool/RsaCtfTool
- “The tool will cycle through each selected attack for a given public key. RSA security relies on the complexity of the integer factorization problem. This project is a glue between various integer factorization algorithms.”

Dump N

`./RsaCtfTool.py --dumpkey --key ./key.pub`

- You get N and E.



FactorDB

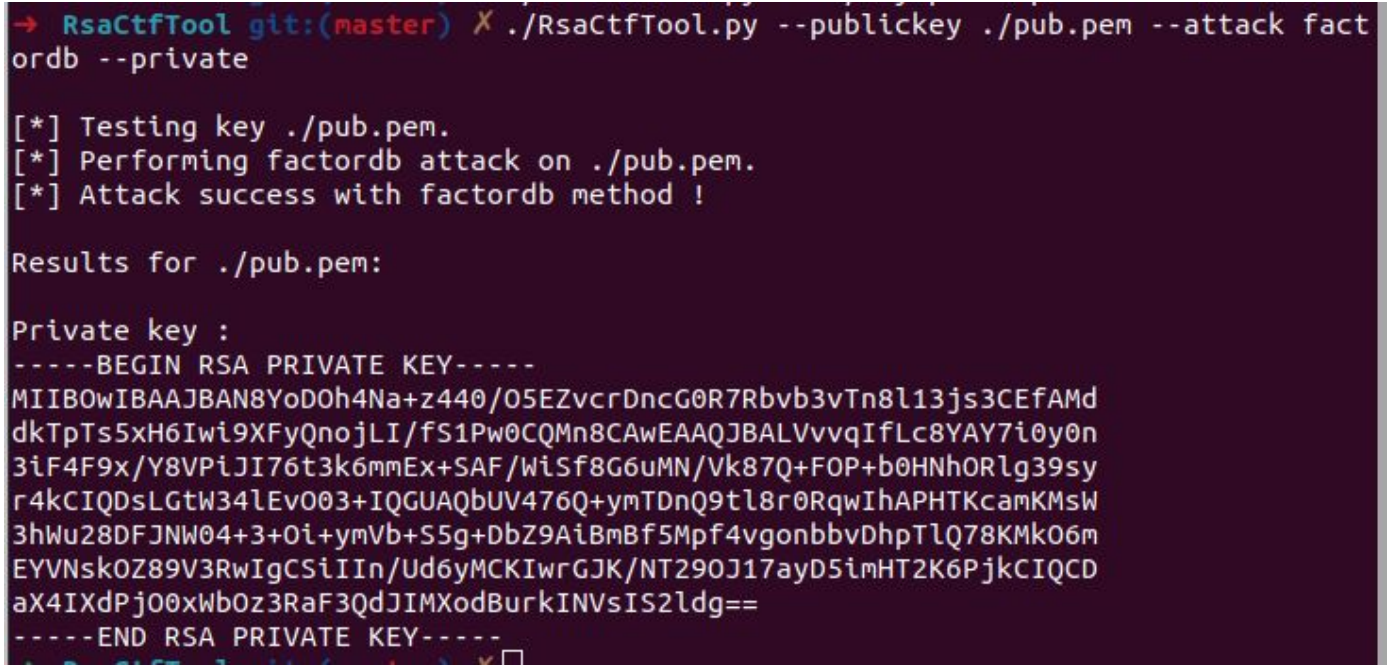
- Paste this N into FactorDB to check if factors are available.

FactorDB

- Paste this N into FactorDB to check if factors are available.
- They are - means we just have create a private key from these factors.
- You could go to https://www.dcode.fr/rsa-cipher and recalculate values, but RsaCtfTool has a built-in factordb attack that does that for you.

Run Attack

- ./RsaCtfTool.py --publickey ./pub.pem --attack factordb --private



Source Code: super_secure_rsa.py

- We are given only the encrypt() function
- Need to write decrypt()

Source Code: super_secure_rsa.py

- Code for decryption is missing
- Write code and get flag.
  - Write function to: load_private_key()
  - Add code in main to decrypt string from flag.enc using private key

Source Code: super_secure_rsa.py

- Code for decryption is missing
- Write code and get flag.
  - Write function to: load_private_key()
  - Add code in main to decrypt string from flag.enc using private key
- Full Code: https://tinyurl.com/55m93axj

Problem 3: misc/mesothelioma

- Find the location of this sign.
- Flag format: dam{dd.mm.ss,dd.mm.ss}

Solution

- Whats that triangle logo? BLM
- What BLM property has Asbestos? Clear Creek
- Reverse image search / article search
- Find article with another image of sign
- https://www.sfgate.com/green/article/SAN-BENITO-COUNTY-Curbing-off-road-recreation-2749702.php#photo-2188639
- That image has more context on where the sign is
- Another BLM sign in the background
  - a road split
  - two trees
  - based on that, it's in front of / just under the first tree
- You may also find this: https://cal4wheel.com/phocadownload/ierf_ccma_final_3_8_11.pdf
-

Solution

- Rounded to the nearest second:
- https://www.google.com/maps/place/36%C2%B022'00.0%22N+120%C2%B045'09.0%22W/@36.3668569,-120.7525615,149m/data=!3m1!1e3!4m4!3m3!8m2!3d36.3666667!4d-120.7525