

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



message.zip

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?

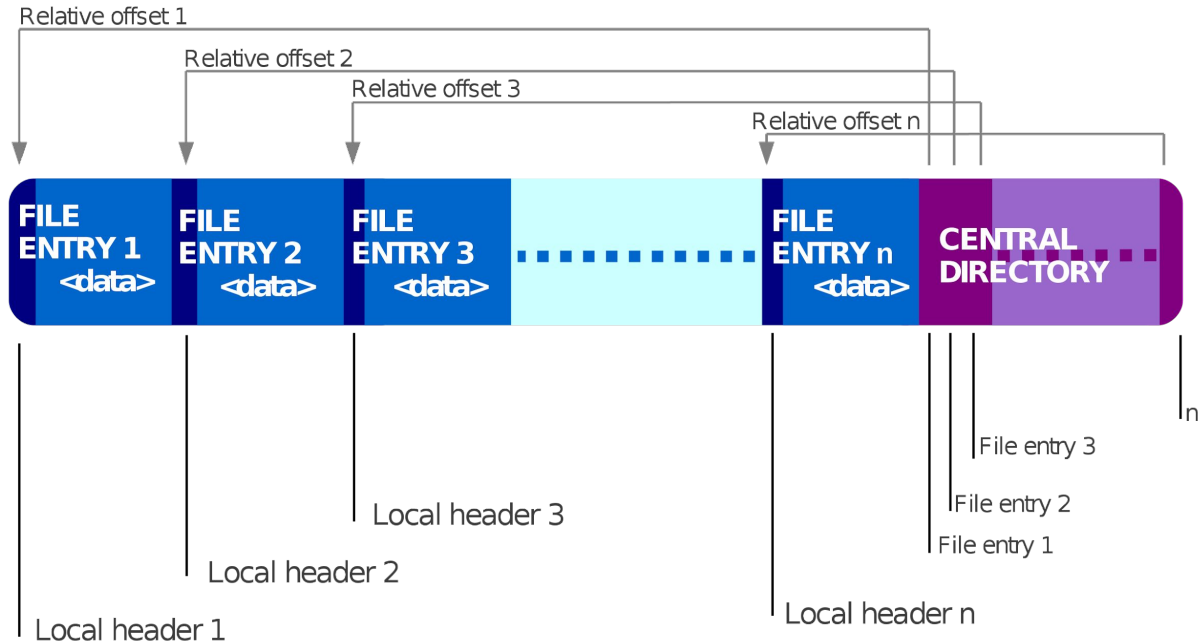
```
→ de-compressed git:(master) binwalk message.zip
```

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++

```
1 |Iread..... betweenthe..... lines, my
 |vision'.....sclear..... and..... keen.....
2 | I seethe hidden..... meanings,the truths
 |that.....are unseen
3 |I don'.....tjust..... take
 |thingsatface value,.....that.....'s
 | not..... my.....style
4 | Idig.....deep and I uncover....., the
 |hidden.....treasures.....that.....
 |are.....compiled.....
5 |
```



# Unicode Steganography

- Paste the text into  
[https://330k.github.io/misc\\_tools/unicode\\_steganography.html](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

## RSA Algorithm

### Key Generation

|                                  |                                         |
|----------------------------------|-----------------------------------------|
| Select $p, q$                    | $p$ and $q$ , both prime; $p \neq q$    |
| Calculate $n = p \times q$       |                                         |
| Calculate $\phi(n) = (p-1)(q-1)$ |                                         |
| Select integer $e$               | $\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$ |
| Calculate $d$                    | $de \bmod \phi(n) = 1$                  |
| Public key                       | $KU = \{e, n\}$                         |
| Private key                      | $KR = \{d, n\}$                         |

### Encryption

|             |                   |
|-------------|-------------------|
| Plaintext:  | $M < n$           |
| Ciphertext: | $C = M^e \bmod n$ |

### Decryption

|             |                   |
|-------------|-------------------|
| Plaintext:  | $C$               |
| Ciphertext: | $M = C^d \bmod n$ |

# 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.

```
→ RsaCtfTool git:(master) X ./RsaCtfTool.py --dumpkey --key ./pub.pem
private argument is not set, the private key will not be displayed, even if recovered.
n: 11684495802889072585203310515250083572285658052270998153007378254694580706620
837521287604089276341404868210594675627429508088431073125103913482926295102079
e: 65537
```

# 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`

```
→ RsaCtfTool git:(master) X ./RsaCtfTool.py --publickey ./pub.pem --attack factordb --private

[*] Testing key ./pub.pem.
[*] Performing factordb attack on ./pub.pem.
[*] Attack success with factordb method !

Results for ./pub.pem:

Private key :
-----BEGIN RSA PRIVATE KEY-----
MIIBOwIBAAJBAN8YoD0h4Na+z440/05EZvcrDncG0R7Rbvb3vTn8l13js3CEfAMd
dkTpTs5xH6Iwi9XFyQnojLI/fs1Pw0CQMn8CAwEAAQJBALVvvqIfLc8YAY7i0y0n
3iF4F9x/Y8VPiJI76t3k6mEx+SAF/WiSf8G6uMN/Vk87Q+FOP+b0HNhORlg39sy
r4kCIQDsLGtW34lEv003+IQGUAQbUV476Q+ymTDnQ9tl8r0RqwIhAPHTKcamKMSW
3hWu28DFJNW04+3+Oi+ymVb+S5g+DbZ9AiBmBf5Mpf4vgonbbvDhpTlQ78KMk06m
EYVNskOZ89V3RwIgCSiIIn/Ud6yMCKIwrGJK/NT290J17ayD5imHT2K6PjkCIQCD
aX4IXdPj00xWb0z3RaF3QdJIMXodBurkINVsiS2ldg==
-----END RSA PRIVATE KEY-----
```



## 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](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>