# Cyberstorm CTF 2025 Writeups (My First CTF)

## Treasure Trove (Pt 0)

**Challenge Details**

**Category:** Reverse Engineering

**Points:** 100

**Description:**

Shanks found a trove of binaries with treasure hidden inside them. His crew managed to find most of the treasure but struggled with a few. I stepped in to help Shanks score some extra dough before he sets off chasing the thrill of becoming an Emperor.

### My Approach

**Step 1: Checking the File Type**

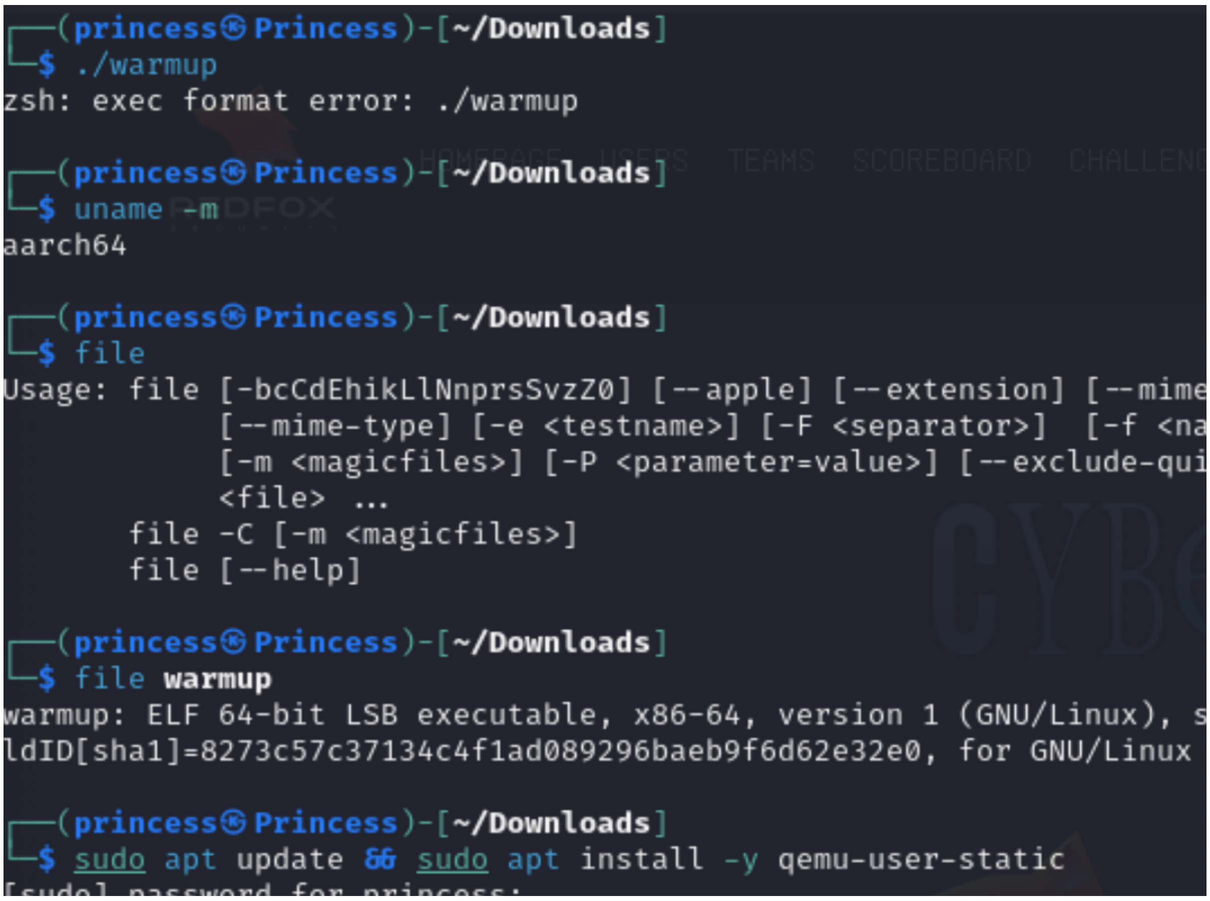
I started by checking the file type using the file command:

```
file warmup
```

Output:

```
warmup: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, BuildID[sha1]=8273c57c37134c4f1ad089296baeb9f6d62e32e0, for GNU/Linux 3.2.0, not stripped
```

This confirmed that the binary was a 64-bit Linux executable.



```
┌──(princess㉿Princess)-[~/Downloads]
└─$ ./warmup
zsh: exec format error: ./warmup

┌──(princess㉿Princess)-[~/Downloads]
└─$ uname -m
aarch64

┌──(princess㉿Princess)-[~/Downloads]
└─$ file
Usage: file [-bcCdEhikLlNnprsSvzZ0] [--apple] [--extension] [--mime
            [--mime-type] [-e <testname>] [-F <separator>]  [-f <na
            [-m <magicfiles>] [-P <parameter=value>] [--exclude-qui
            <file> ...
       file -C [-m <magicfiles>]
       file [--help]

┌──(princess㉿Princess)-[~/Downloads]
└─$ file warmup
warmup: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), s
ldID[sha1]=8273c57c37134c4f1ad089296baeb9f6d62e32e0, for GNU/Linux

┌──(princess㉿Princess)-[~/Downloads]
└─$ sudo apt update && sudo apt install -y qemu-user-static
[sudo] password for princess:
```

**Step 2: Running the Binary**

Since I was on macOS (ARM-based), I couldn't run the binary directly. I switched to a Kali Linux environment and executed the file:

```
chmod +x warmup
./warmup
```

It prompted me for a **Secret Code**.

**Step 3: Analyzing the Binary**

To figure out what input it expected, I looked at the strings inside the binary:

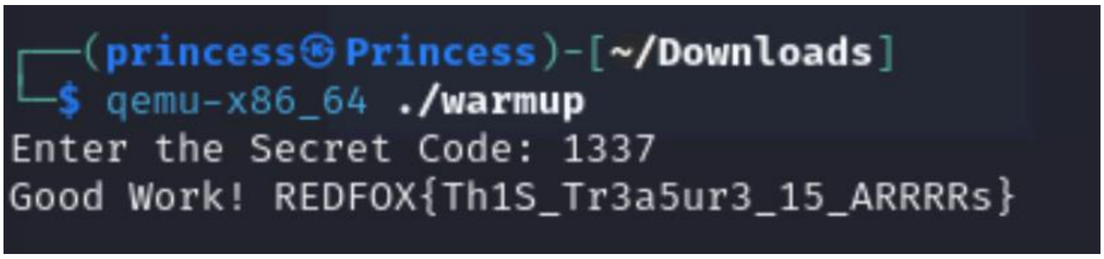
```
strings warmup
```

One of the outputs hinted that the input 1337 might be the correct one.

**Step 4: Entering the Code**

I ran the binary again and entered 1337:

Enter Secret Code: 1337
Correct! Here is your flag: REDFOX{Th1S_Tr3a5ur3_15_ARRRRs}



```
┌──(princess㉿Princess)-[~/Downloads]
└─$ qemu-x86_64 ./warmup
Enter the Secret Code: 1337
Good Work! REDFOX{Th1S_Tr3a5ur3_15_ARRRRs}
```

**Flag:**

REDFOX{Th1S_Tr3a5ur3_15_ARRRRs}

## Tools I Used

- file – to determine the binary type
- strings – to pull out readable strings
- chmod – to make the binary executable
- qemu-x86_64 – in case I needed to emulate x86-64 on ARM

## Comparison (299 pts)

**Challenge Details**

**Category:** Reverse Engineering

**Points:** 299

**Description:**

Everyone has started comparing Yamato to Oden since the self-proclamation. Some days, Yamato feels energized, and sometimes belittled. I had to help Yamato overcome these unfair comparisons.

**Connection Info:**

```
nc cyberstorm.redfoxsec.com 17000
```

### My Approach

**Step 1: Checking the Given File**

The challenge gave me a file named easy. I started by checking its contents:

```
cat easy
```

The output was full of characters separated by spaces—hard to read and not very helpful at first glance.

**Step 2: Extracting Readable Strings**

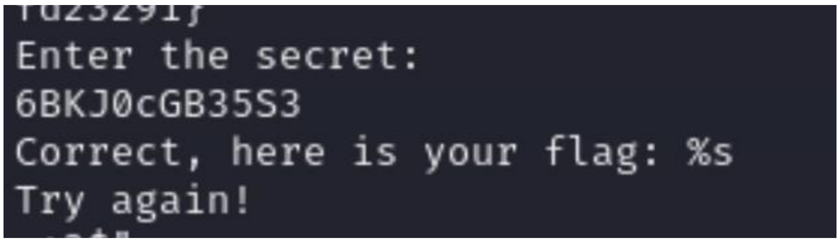
To make things easier to digest, I ran:

```
strings easy
```

This gave me a clean list of readable strings. While scanning through them, one line stood out:

The secret code is: 6BKJ0cGB35S3

That was my key.



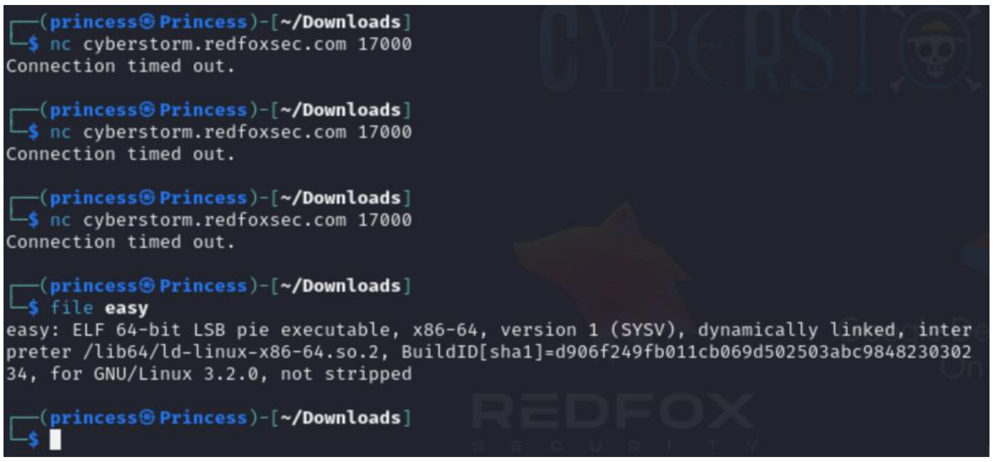
```
fd23291}
Enter the secret:
6BKJ0cGB35S3
Correct, here is your flag: %s
Try again!
```

**Step 3: Sending the Secret Code**

To complete the challenge, I had to connect using Netcat:

nc cyberstorm.redfoxsec.com 17000

The catch? The server timed out fast. So I had to be quick—connect and immediately paste the secret code. After a couple of tries, I nailed the timing and got the flag.



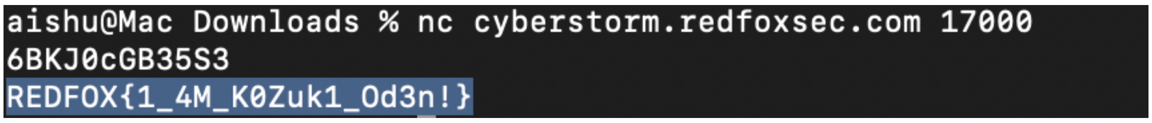
```
┌──(princess㉿Princess)-[~/Downloads]
└─$ nc cyberstorm.redfoxsec.com 17000
Connection timed out.

┌──(princess㉿Princess)-[~/Downloads]
└─$ nc cyberstorm.redfoxsec.com 17000
Connection timed out.

┌──(princess㉿Princess)-[~/Downloads]
└─$ nc cyberstorm.redfoxsec.com 17000
Connection timed out.

┌──(princess㉿Princess)-[~/Downloads]
└─$ file easy
easy: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, inter
preter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=d906f249fb011cb069d502503abc9848230302
34, for GNU/Linux 3.2.0, not stripped

┌──(princess㉿Princess)-[~/Downloads]
└─$ 
```



```
aishu@Mac Downloads % nc cyberstorm.redfoxsec.com 17000
6BKJ0cGB35S3
REDFOX{1_4M_K0Zuk1_Od3n!}
```

**Flag:**

REDFOX{1_4M_K0Zuk1_0d3n! }

## Tools I Used

- cat – to view the raw file
- strings – to extract readable text
- nc – Netcat, to interact with the server

## Big Mom's Plot (100 pts)

**Challenge Details**

**Category:** Reverse Engineering

**Points:** 100

**Description:**

Big Mom has been keeping her new war strategies tightly under wraps. Not even Cipher Pol has any intel. My job? Figure out what she's hiding—just in case the Straw Hat crew ends up clashing with her forces.

**Connection Info:**

nc cyberstorm.redfoxsec.com 4444

### My Approach

**Step 1: Connecting to the Server**

I used Netcat to connect and interact with the challenge service:

```
nc cyberstorm.redfoxsec.com 4444
```

**Step 2: Analyzing the Assembly Code**

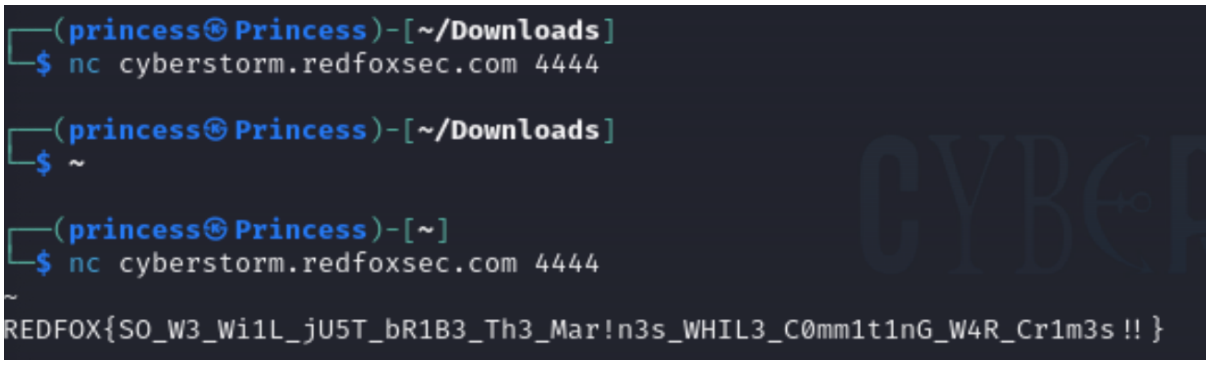
I was given a file in assembly format, warmup.asm. Here's what I noticed after going through it:

1. The code uses sys_read to read user input into a buffer.
2. It loops through each character of the input.
3. If it finds a character that is ~ (ASCII 126 or 0x7E), it prints the flag.
4. If ~ isn't found, it just exits.

**Step 3: Extracting the Flag**

So the trick was simple: enter a single ~ when prompted.

~



```
┌──(princess㉿Princess)-[~/Downloads]
└─$ nc cyberstorm.redfoxsec.com 4444

┌──(princess㉿Princess)-[~/Downloads]
└─$ ~

┌──(princess㉿Princess)-[~]
└─$ nc cyberstorm.redfoxsec.com 4444
~
REDFOX{SO_W3_Wi1L_jU5T_bR1B3_Th3_Mar!n3s_WHIL3_C0mm1t1nG_W4R_Cr1m3s !! }
```

Right after that, the server responded with:

REDFOX{SO_W3_Wi1L_jU5T_bR1B3_Th3_Mar!n3s_WHIL3_C0mm1t1nG_W4R_Cr1m3s !!}

## Tools I Used

- nc – to connect to the server
- vim / nano – to read and analyze warmup.asm
- Basic assembly knowledge – to figure out the logic