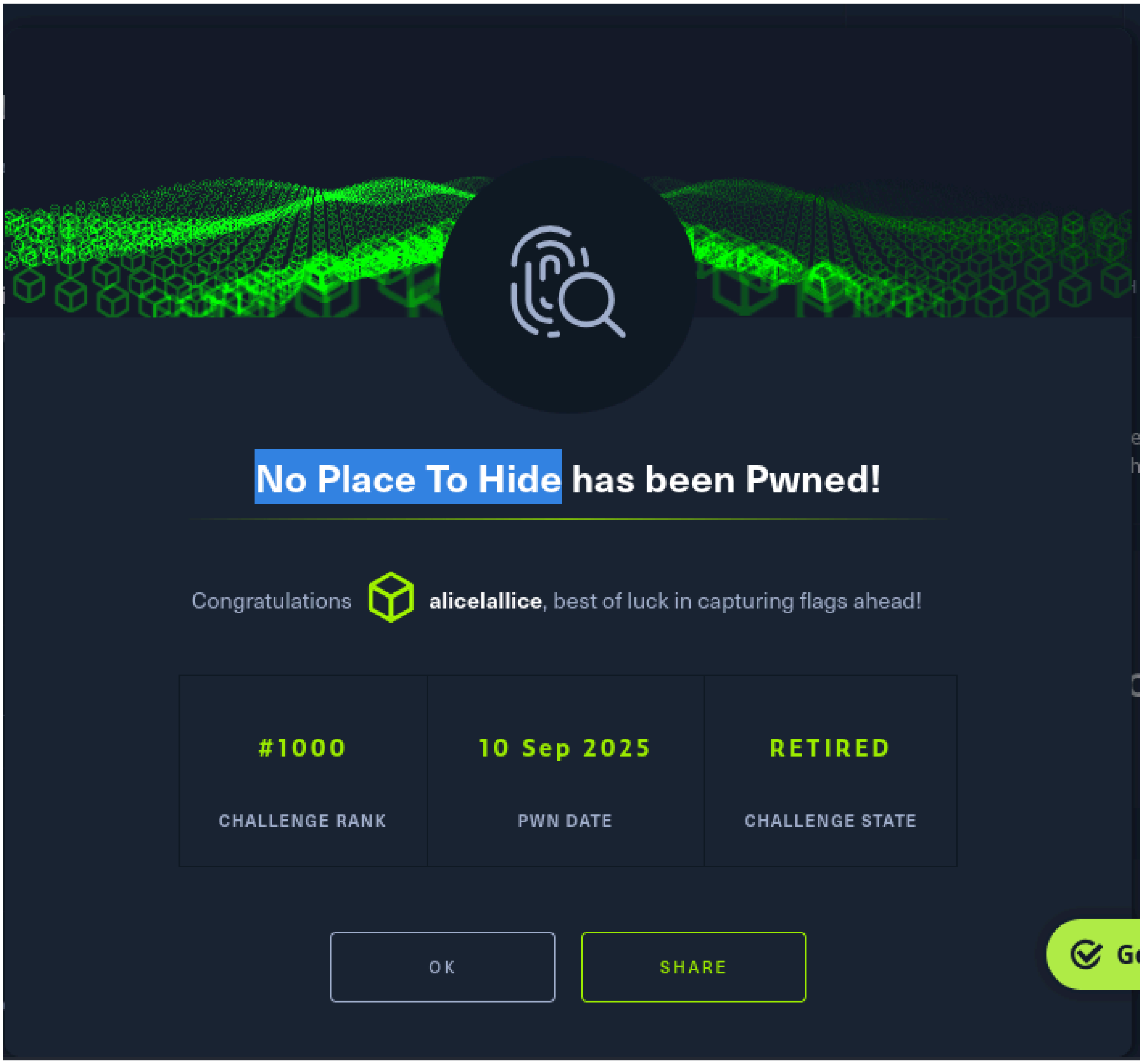
No Place To Hide

Types	forensic
CTF	HTB



No Place To Hide has been Pwned!

Congratulations alicelallice, best of luck in capturing flags ahead!

#1000	10 Sep 2025	RETIRED
CHALLENGE RANK	PWN DATE	CHALLENGE STATE

The challenge provided a **BMC cache file (`Cache0000.bin`)** which contained captured **remote desktop session tiles**. These tiles needed to be extracted and reassembled into the original desktop view to reveal the flag.

🛠️ Tools Used

- **Python 3**
- **Pillow (PIL)** – for image handling
- **bmc-tools** – to parse `.bin` and extract tile images
- **Linux CLI utilities**

Install dependencies:

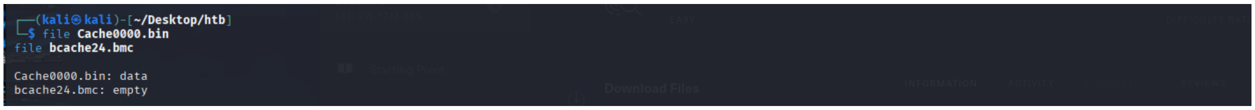
```
pip install pillow
git clone https://github.com/ANSSI-FR/bmc-tools
```

Step 1: Inspecting the Data

First, we checked the type of file:

```
file Cache0000.bin
```

It showed as raw **data**, meaning we couldn't directly open it as an image.

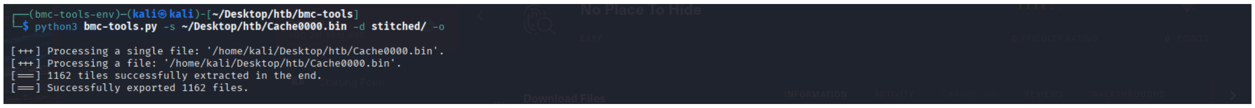


```
┌──(kali㉿kali)-[~/Desktop/htb]
└─$ file Cache0000.bin
file bcache24.bmc

Cache0000.bin: data
bcache24.bmc: empty
```

Step 2: Extracting Tiles with `bmc-tools`

We used **bmc-tools** to process the `.bin` and export the graphics tiles:

```
python3 bmc-tools.py -s ~/Desktop/htb/Cache0000.bin -d stitched/ -o
```



```
┌──(bmc-tools-env)─(kali㉿kali)-[~/Desktop/htb/bmc-tools]
└─$ python3 bmc-tools.py -s ~/Desktop/htb/Cache0000.bin -d stitched/ -o

[+++] Processing a single file: '/home/kali/Desktop/htb/Cache0000.bin'.
[+++] Processing a file: '/home/kali/Desktop/htb/Cache0000.bin'.
[===] 1162 tiles successfully extracted in the end.
[===] Successfully exported 1162 files.
```

This gave us **1162** `.bmp` **tile images** inside the `stitched/` directory.

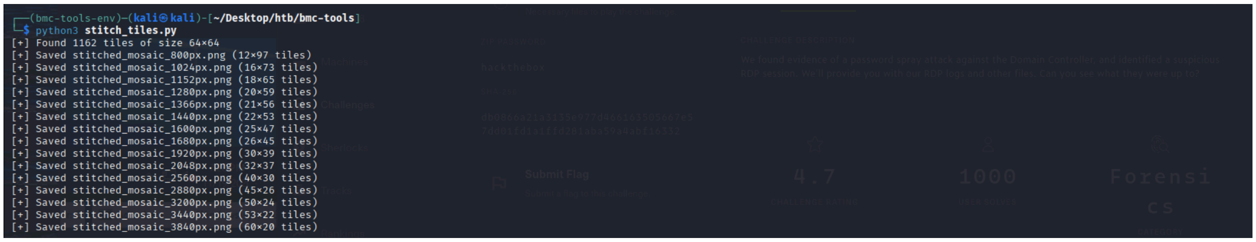
Step 3: Stitching the Tiles

Since the extracted tiles were small chunks of the screen, we needed to reassemble them into a larger image.

We wrote a Python script to test different **common screen widths** until the tiles aligned correctly:

```
from PIL import Image
import glob, math

tiles = sorted(glob.glob("stitched/Cache0000.bin_*.bmp"))
n = len(tiles)
w, h = Image.open(tiles[0]).size

print(f"[+] Found {n} tiles of size {w}x{h}")

common_widths = [800,1024,1280,1366,1440,1600,1680,1920,2048,2560,2880,
3200,3440,3840]

for width in common_widths:
    cols = width // w
    if cols == 0:
        continue
    rows = math.ceil(n / cols)
    mosaic = Image.new("RGB", (cols * w, rows * h))

    for idx, tile in enumerate(tiles):
        img = Image.open(tile)
        x = (idx % cols) * w
        y = (idx // cols) * h
        mosaic.paste(img, (x, y))
```

```
out_name = f"stitched_mosaic_{width}px.png"
mosaic.save(out_name)
print(f"[+] Saved {out_name}")
```

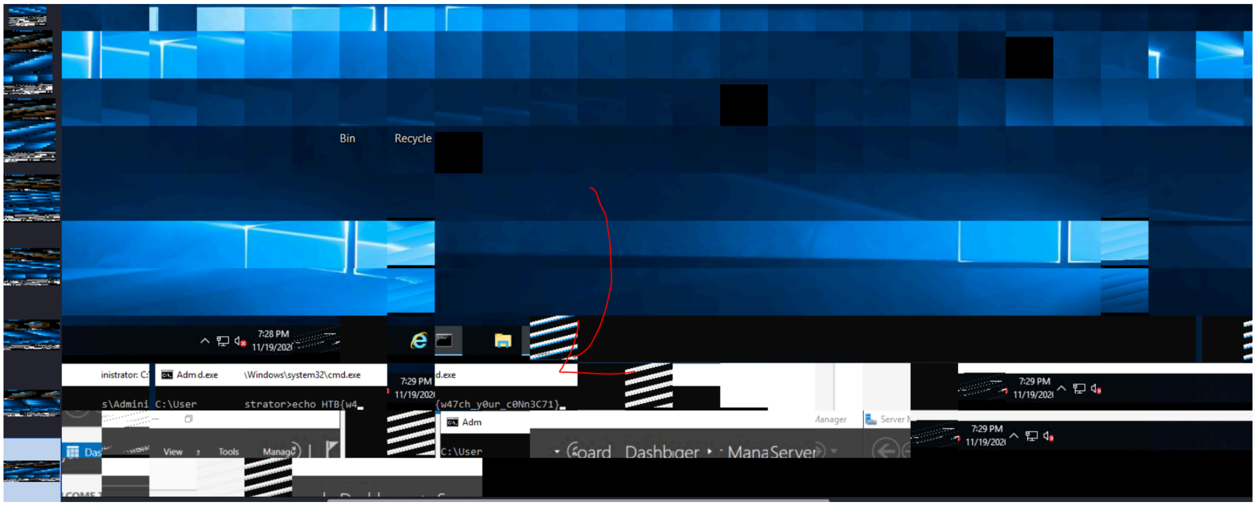


```
┌──(bmc-tools-env)─(kali㉿kali)-[~/Desktop/htb/bmc-tools]
└─$ python3 stitch_tiles.py
[+] Found 1162 tiles of size 64×64
[+] Saved stitched_mosaic_800px.png (12×97 tiles)
[+] Saved stitched_mosaic_1024px.png (16×73 tiles)
[+] Saved stitched_mosaic_1152px.png (18×65 tiles)
[+] Saved stitched_mosaic_1280px.png (20×59 tiles)
[+] Saved stitched_mosaic_1366px.png (21×56 tiles)
[+] Saved stitched_mosaic_1440px.png (22×53 tiles)
[+] Saved stitched_mosaic_1600px.png (25×47 tiles)
[+] Saved stitched_mosaic_1680px.png (26×45 tiles)
[+] Saved stitched_mosaic_1920px.png (30×39 tiles)
[+] Saved stitched_mosaic_2048px.png (32×37 tiles)
[+] Saved stitched_mosaic_2560px.png (40×30 tiles)
[+] Saved stitched_mosaic_2880px.png (45×26 tiles)
[+] Saved stitched_mosaic_3200px.png (50×24 tiles)
[+] Saved stitched_mosaic_3440px.png (53×22 tiles)
[+] Saved stitched_mosaic_3840px.png (60×20 tiles)
```

Step 4: Finding the Correct Alignment

After generating multiple stitched mosaics, the **1920px width** version aligned correctly, reconstructing a Windows desktop.

When zooming in, we found the flag clearly displayed.



final Flag

```
HTB{w47ch_y0ur_c0Nn3C71}
```

💡 Takeaways

- `.bin` files from BMC can hold **screen capture tiles**.
- With `bmc-tools`, we can **extract tiles** from cache dumps.
- Stitching tiles at the **correct resolution** reconstructs the screen and may leak sensitive information.