```
1 from
  testausserveri
  import
  hack, try_harder
2
3 while True:
4  try:
5   hack()
6  except:
7   try_harder()
```

CTF Challenge Creation

Used to be a presentation at Disobey 2025

A good CTF challenge is …

- Problem solving at its best
- A lesson, not a chore ~~*(or maybe a little)*~~
- Satisfying to solve
- Unique/Personal
- Simulating a real-world scenario

What not to do

- Make things difficult for the sake of “being difficult”
- Fake Flags
- Guess the tool

How to create a good challenge

1. Come up with an idea
   - What do you want to share with others?
   - What do you think is cool?

2. Plan the “flow”
   - What are the steps a competitor should accomplish to get the flag
   - Steps can be complicated, but try to keep the number of steps small
   - Minimum of 3 core steps

Example solve from the European CTF championship

(Misc) The good old days

“ Behold our new invention, the phone! “

You get
1. A phone
2. Phone Server configs
3. You can call other teams. Nice!

Then what?

1. Looking around; “Redacted”?

Found in the Central Phone Server Config (“PSTN”, using Asterisk)

```
[default]
exten => _00xx,1,Answer()
 same => n,Playback("/sounds/other/call_enter")
 same => n,MixMonitor(/recordings/${CALLERID(num)}-team${EXTEN:2}-${UNIQUEID}.wav,b)
 same => n,Set(CDR(recordingpath)=/recordings/${CALLERID(num)}-
team${EXTEN:2}-${UNIQUEID}.wav)
 same => n,Dial(PJSIP/0000@team${EXTEN:2},30)
 same => n,Playback("/sounds/other/call_busy")
 same => n,Hangup()

exten => _9999,1,Answer()
 same => n,Playback("/sounds/other/call_enter")
 same => n,MixMonitor(/recordings/${CALLERID(num)}-9999-${UNIQUEID}.wav,b)
 same => n,Set(CDR(recordingpath)=/recordings/${CALLERID(num)}-9999-${UNIQUEID}.wav)
 same => n,REDACTED

```

Dial 00xx -> Other Teams

Dial 9999 -> Redacted?

Hmm 🤔

2. Not so easy

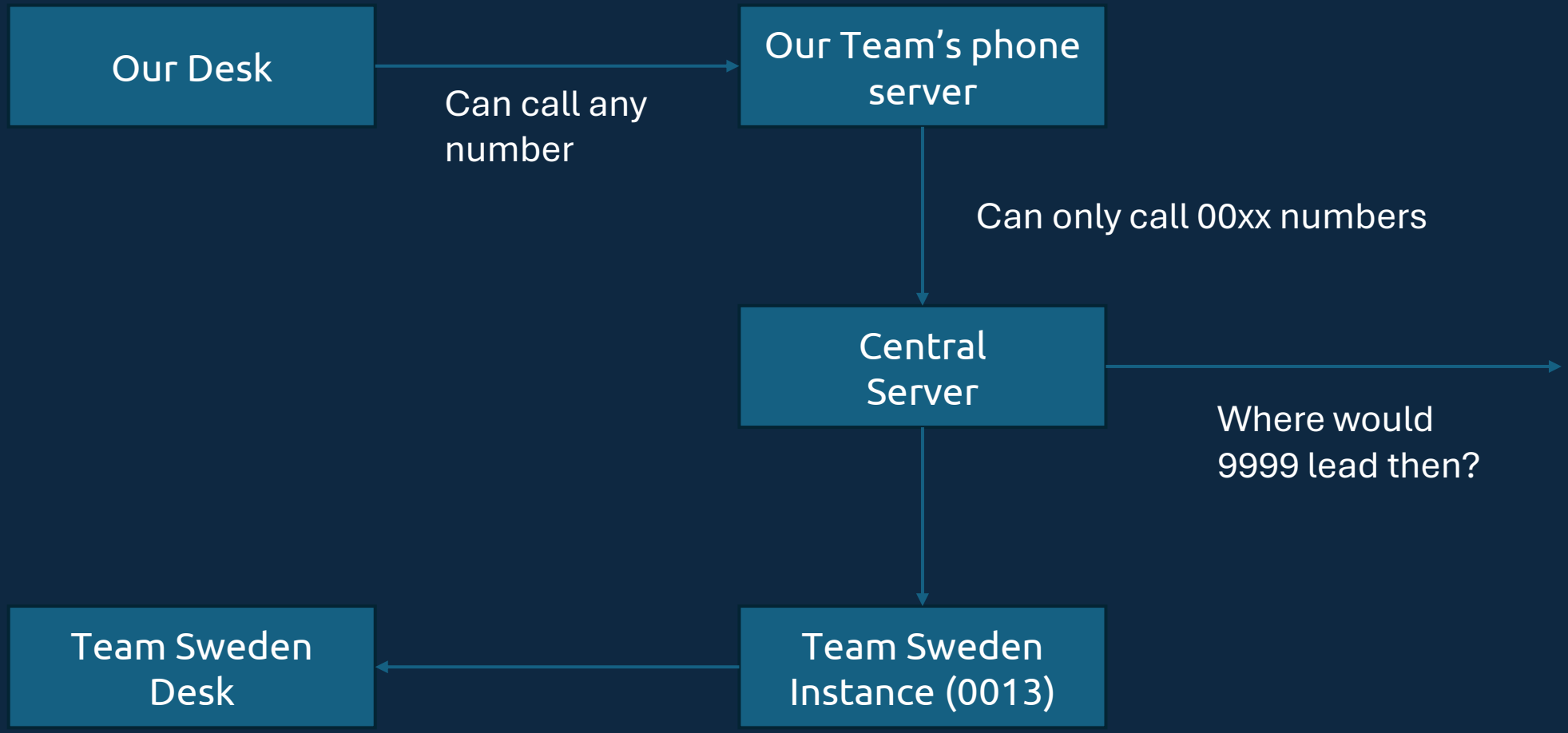
Our Phone Server Config; Only supports calling 00xx numbers

```
1 [default]
2 exten => _00.,1,Set(GROUP()=${EPOCH})
3  same => n,GotoIf($[${GROUP_COUNT(${EPOCH})}>${calls_per_sec}]?ratelimit,${EXTEN},1)
4  same => n,Dial(PJSIP/${EXTEN}@pstn,120)
5  same => n,Hangup()
6 exten => _0099,1,Goto(system-configuration,s,1)
7
```

That's it?

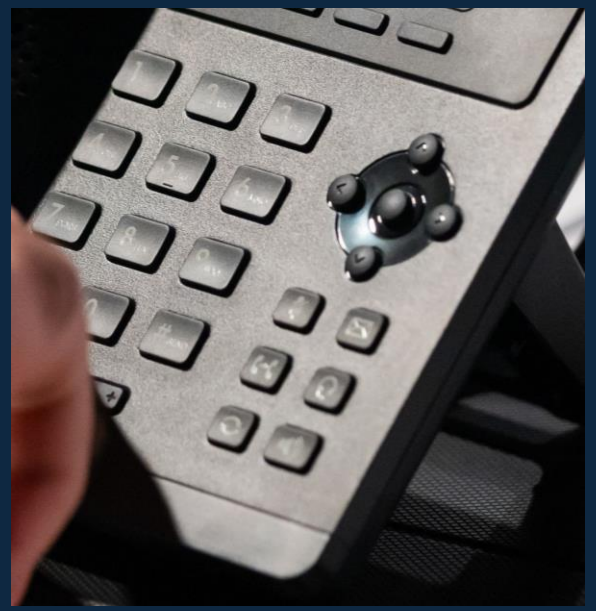
Ugh :/

3. Using the brain; Building a network diagram



4. The exploit

> We want to call 9999
> We can only call numbers that begin with 00



<- But we can input letters too!

```
1 [default]
2 exten => _00.,1,Set(GRO
3  same => n,GotoIf($[${G
4  same => n,Dial(PJSIP/$
5  same => n,Hangup()
6 exten => _0099,1,Goto(s
7
```

What if inject our own dial commands?

5. The exploit

& splits the command like in a shell

Our server now calls:

~~0000~~
9999

Call: 0000&PJSIP/9999

FLAG! A robot voice at 9999 will speak:
ECSC{JUS7_D0NT_G0_4ROUND_M3SS1NG_W1TH_P30PLE5_PHON3S_N0W!_C071C556}

Server-side ~~request~~ dial forgery!

Summary

- **Idea**
  Hacking physical phones is cool & understanding how the phone servers are configured

- **Steps**
  1. Understanding the provided files and how it affects the phone’s abilities
  2. Understanding where the flag is and why we can’t get it
  3. Discovering the command injection in the files
  4. Using the vulnerability

Challenge Prompts

WEB: Easy XSS

- **Idea**
  Have you heard of XSS? Performing exfiltration through the network.

- **Challenge**
  - A simple text-editor and document storage website
  - Users can login and create html rich documents with <script> tags
  - A bot will navigate to a user provided document to “scan it” for publishing
  - The flag is in the global scope of the bot’s runtime as “flag” (code provided)
  - The user can exfiltrate the flag through the network

- **Steps to solve**
  1. Understand creating documents and how the bot scans it
  2. Find the flag in the bot’s code
  3. Discover XSS and exploit it

MISC: Images have metadata

- **Idea**
  Have you heard of EXIF? A huge privacy risk, right?

- **Challenge**
  - Provide the user with bunch of images from around the world
  - The images have EXIF metadata in them that discloses the location the images were taken it
  - The coordinates in the images form a QR-code you can scan to get the flag

- **Steps to solve**
  1. Browse through the images, lots of places!
  2. Discover EXIF data
  3. Map the coordinates and scan the QR-code (requires coding)

REVERSE: Binary decompile

- **Idea**
  Ghidra has a very good decompiler!

- **Challenge**
  - Create a binary that checks user input in some way
  - Check user input character by character against an obfuscated flag stored as integers or in some non-plaintext form.

- **Steps to solve**
  1. Understand what the executable does
  2. Find a decompiler and use it
  3. Understand the decompiled code and reverse engineer the flag from it