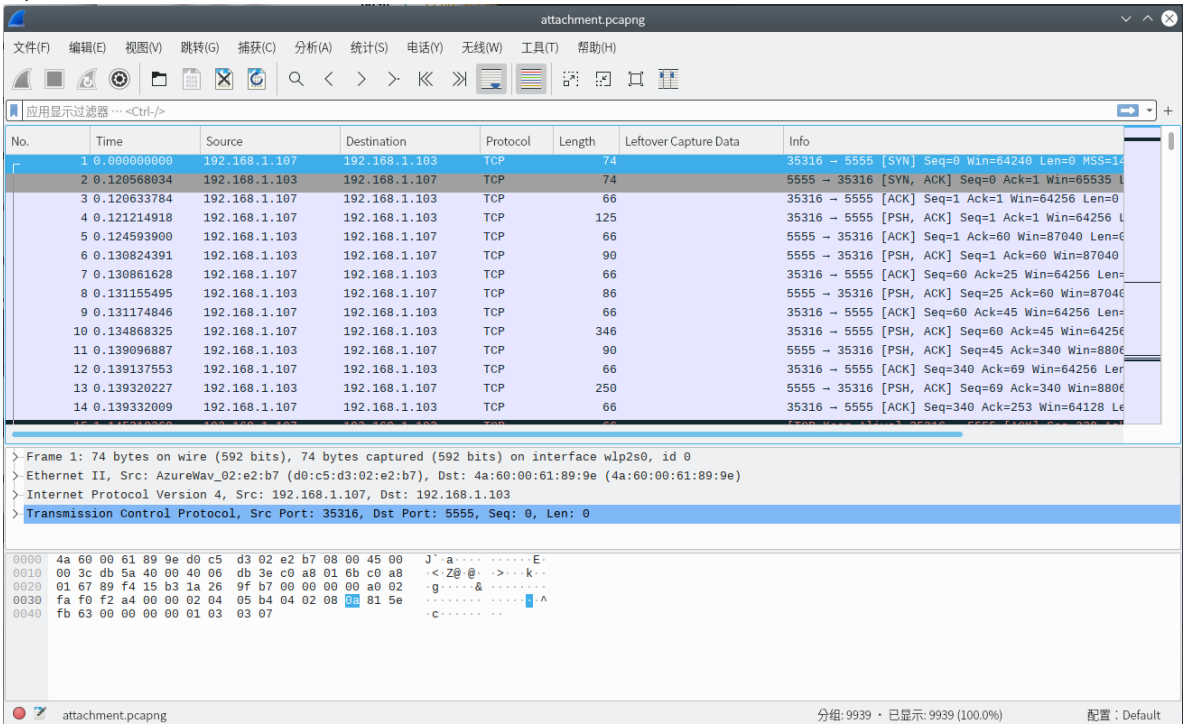
Open the attachment in Wireshark:

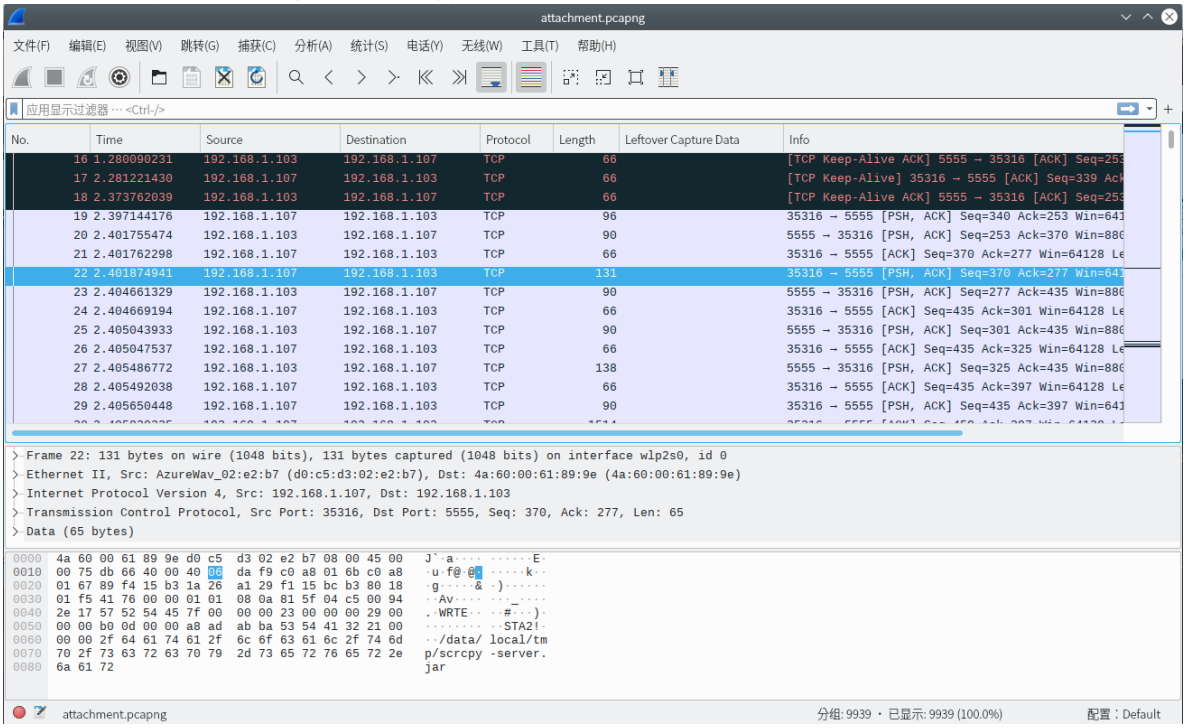


Port 5555, It's `Network ADB` , let's dig deeper.

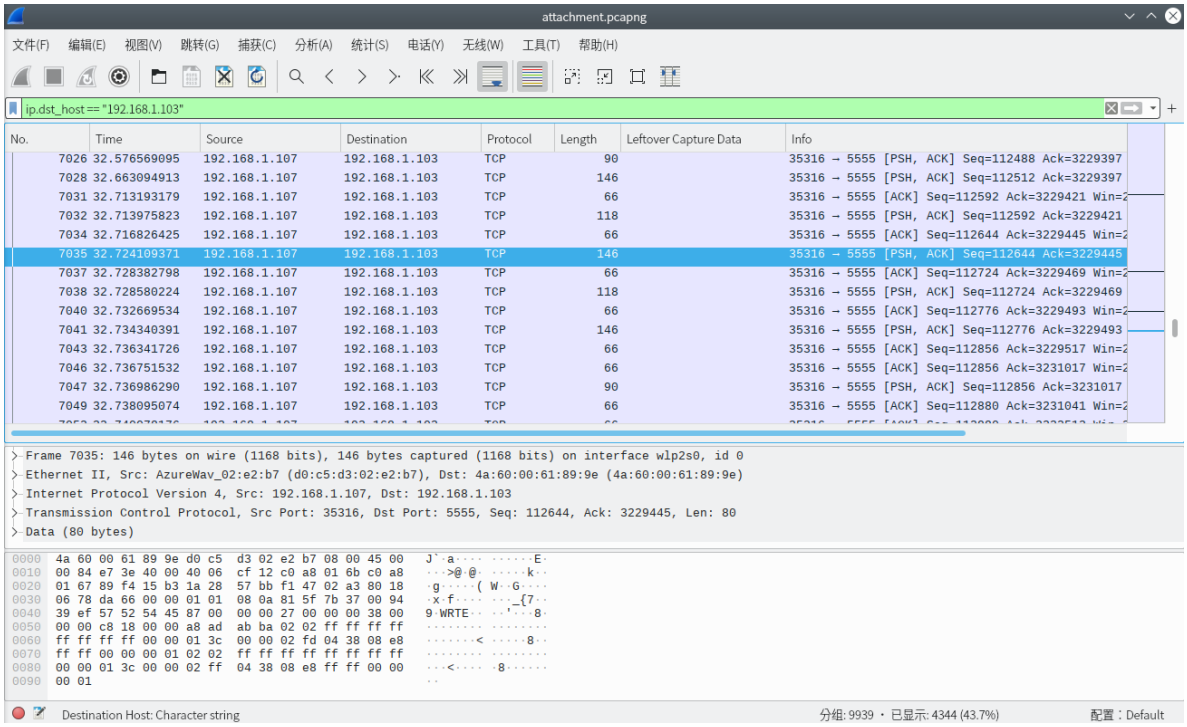
Generally we want to see the document of ADB protocol, which can be found here:
https://github.com/cstyan/adbDocumentation
You can know that `WRTE` means the packet is sent to the client.

> you can also skip this if you directly found this packet, but it may be harder afterwards:

So it's actually something about scrcpy.



Filter out packets sent from the compter:
what does these packets mean?



Search the web and you can find this Github issue:
https://github.com/Genymobile/scrcpy/issues/673
says there's no documentation, but you can refer to the source code:
https://github.com/Genymobile/scrcpy/blob/6b3d9e3eab1d9ba4250300eccd04528dbee9023a/app/tests/test_control_msg_serialize.c

```
static void test_serialize_inject_mouse_event(void) {
    struct control_msg msg = {
        .type = CONTROL_MSG_TYPE_INJECT_MOUSE_EVENT,
        .inject_mouse_event =
            {
                .action = AMOTION_EVENT_ACTION_DOWN,
                .buttons = AMOTION_EVENT_BUTTON_PRIMARY,
                .position =
                    {
                        .point =
                            {
                                .x = 260,
                                .y = 1026,
                            },
                        .screen_size =
                            {
                                .width = 1080,
                                .height = 1920,
                            },
                    },
            },
    };
    unsigned char buf[CONTROL_MSG_SERIALIZED_MAX_SIZE];
    int size = control_msg_serialize(&msg, buf);
    assert(size == 18);
    const unsigned char expected[] =
    { CONTROL_MSG_TYPE_INJECT_MOUSE_EVENT,
      0x00, // AKEY_EVENT_ACTION_DOWN
      0x00,
      0x00,
```

```
      0x00,
      0x01, // AMOTION_EVENT_BUTTON_PRIMARY
    } 0x00,
        0x00, 0x01, 0x04, 0x00, 0x00, 0x04, 0x02, // 260 1026
        0x04, 0x38, 0x07, 0x80,                   // 1080 1920
};
assert(!memcmp(buf, expected, sizeof(expected)));
```

Then export the capture file as json and match patterns like above, extracting the points(X,Y): (fish script, and `57:52:54:45` is the `adb` command `WRTE` you got from the documentation)

```
for i in (cat /home/leohearts/Desktop/tmp.json |jq .[]._source.layers.tcp | grep
57:52:54:45 | grep -o -E '00:00:..:..:00:00:..:..:04:38:08:e8:ff' | grep -o -E
'..:..:' | grep -v "00:00" | grep -v '04:38' | grep -v '08:e8' | sed 's/://g')
bash -c 'echo $((16#'$i'))' >> pixels
end
```

Then use Python to draw it back to an image:

```
from PIL import Image
def newImg():
    img = Image.new('RGB', (2000, 2000))
    while True:
        try:
            x = int(input())
            y = int(input())
            img.putpixel((x,y), (155,155,55))
        except:
            break
    img.save('sqr.png')
    return img

wallpaper = newImg()
wallpaper.show()
```

`cat pixels | python3 image.py`

Then thats the flag.