

## The Code Composition

CTF #2 – Part 2

medium challenge:

**Goal:** get the flag somewhere inside /home/ubuntu

**IP:** 206.189.220.181

**Hint:**

- *be careful of honeypots*
- *flag is a string that MUST start with "ctf" as the first three letters*
- netcat, cryptography

So, we know at least two things. What we need to find, and the IP.

First thing to do with an IP: **Check opened port.**

```
D:\Desktop {git}
{lamb} nmap -p 1-65535 -T4 -A -v 206.189.220.181
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-23 23:16 Paris, Madrid
NSE: Loaded 151 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 23:16
Completed NSE at 23:16, 0.00s elapsed
Initiating NSE at 23:16
Completed NSE at 23:16, 0.00s elapsed
Initiating NSE at 23:16
Completed NSE at 23:16, 0.00s elapsed
Initiating Ping Scan at 23:16
Scanning 206.189.220.181 [4 ports]
Completed Ping Scan at 23:16, 0.53s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 23:16
Completed Parallel DNS resolution of 1 host. at 23:16, 0.13s elapsed
Initiating SYN Stealth Scan at 23:16
Scanning 206.189.220.181 [65535 ports]
Discovered open port 22/tcp on 206.189.220.181
Discovered open port 8080/tcp on 206.189.220.181
Discovered open port 80/tcp on 206.189.220.181
```

Figure 1 First nmap output

Will waiting for the end of it, let's give an eye on what nmap already found.

**Port 22:** This is common port for SSH, and right now, it's effectively used for SSH.

**Port 80:** Web server, we will give an eye on what happen on when we go on the ip with 80 as port.

**Port 8080:** Alternative HTTP port. Could just be a clone of port 80, we'll check it either.

Few seconds later, more info came to nmap:

```
Discovered open port 8080/tcp on 206.189.220.181
Discovered open port 80/tcp on 206.189.220.181
SYN Stealth Scan Timing: About 27.81% done; ETC: 23:18 (0:01:20 remaining)
SYN Stealth Scan Timing: About 57.69% done; ETC: 23:18 (0:00:45 remaining)
Discovered open port 1337/tcp on 206.189.220.181
Completed SYN Stealth Scan at 23:18, 99.14s elapsed (65535 total ports)
Initiating Service scan at 23:18
Scanning 4 services on 206.189.220.181
Completed Service scan at 23:18, 32.20s elapsed (4 services on 1 host)
Initiating OS detection (try #1) against 206.189.220.181
Retrying OS detection (try #2) against 206.189.220.181
Initiating Traceroute at 23:18
Completed Traceroute at 23:18, 3.02s elapsed
Initiating Parallel DNS resolution of 16 hosts. at 23:18
Completed Parallel DNS resolution of 16 hosts. at 23:18, 0.20s elapsed
NSE: Script scanning 206.189.220.181.
Initiating NSE at 23:18
Completed NSE at 23:18, 5.42s elapsed
Initiating NSE at 23:18
Completed NSE at 23:18, 1.06s elapsed
Initiating NSE at 23:18
Completed NSE at 23:18, 0.00s elapsed
Nmap scan report for 206.189.220.181
Host is up (0.16s latency).
Not shown: 65529 closed ports
PORT      STATE      SERVICE      VERSION
22/tcp    open      ssh          OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   2048 a2:27:36:b2:9c:3b:bb:ce:c5:bd:eb:ba:1d:dd:ce:4f (RSA)
|   256 b4:aa:49:86:50:79:5d:79:d4:9a:8b:e3:95:32:8c:76 (ECDSA)
|_  256 32:bf:6d:a4:58:6b:b2:fc:33:3f:c3:f5:6d:9a:6b:17 (ED25519)
25/tcp    filtered  smtp
80/tcp    open      http         nginx 1.14.0 (Ubuntu)
|_ http-methods:
|_   Supported Methods: GET HEAD
|_ http-server-header: nginx/1.14.0 (Ubuntu)
|_ http-title: Welcome to nginx!
1337/tcp  open      waste?
|_ fingerprint-strings:
|_   GenericLines, GetRequest, HTTPOptions, RTSPRequest:
|_   Command not found, please use 'help' for more info
8080/tcp  open      http         nginx 1.14.0 (Ubuntu)
|_ http-methods:
|_   Supported Methods: GET HEAD POST
|_ http-open-proxy: Proxy might be redirecting requests
```

Figure 2 Second part of nmap output

**Port 1337** is also used. Not a really common port used. Well, let's see what is on port 80 and 8080.

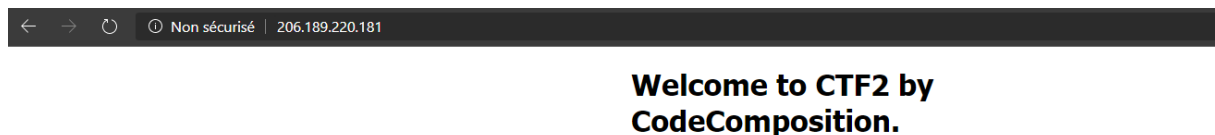


Figure 3 On port 80

This is what happen when we directly go to *206.189.220.181*. Don't seems to interesting right now. I gave an eye to the source code and ... Nothing.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Welcome to nginx!</title>
5 <style>
6     body {
7         width: 35em;
8         margin: 0 auto;
9         font-family: Tahoma, Verdana, Arial, sans-serif;
10    }
11 </style>
12 </head>
13 <body>
14 <h1>Welcome to CTF2 by CodeComposition.</h1>
15 </body>
16 </html>
17
```

Well... Let's check 8080 in that case.

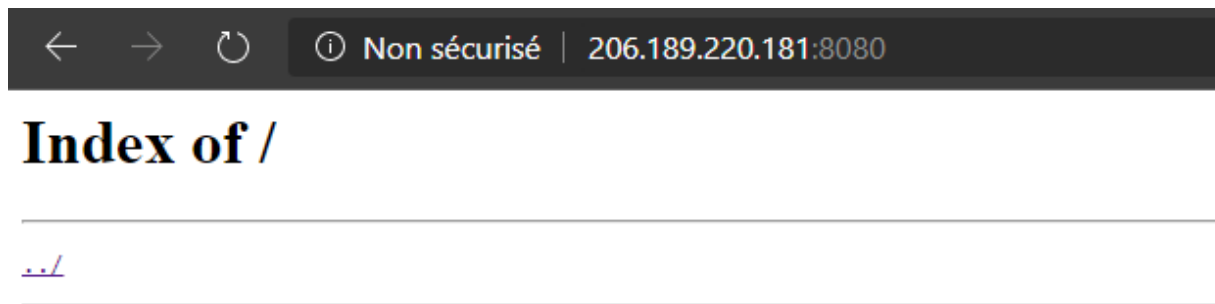


Figure 4 On port 8080

We'll this **could** be interesting. An index of means that any files dropped in the folder defined by .htaccess could be accessed from the http server. ... But there is nothing right now. Still not useful right now, and for sure source code couldn't help there.

Last way is to try port **1337**.

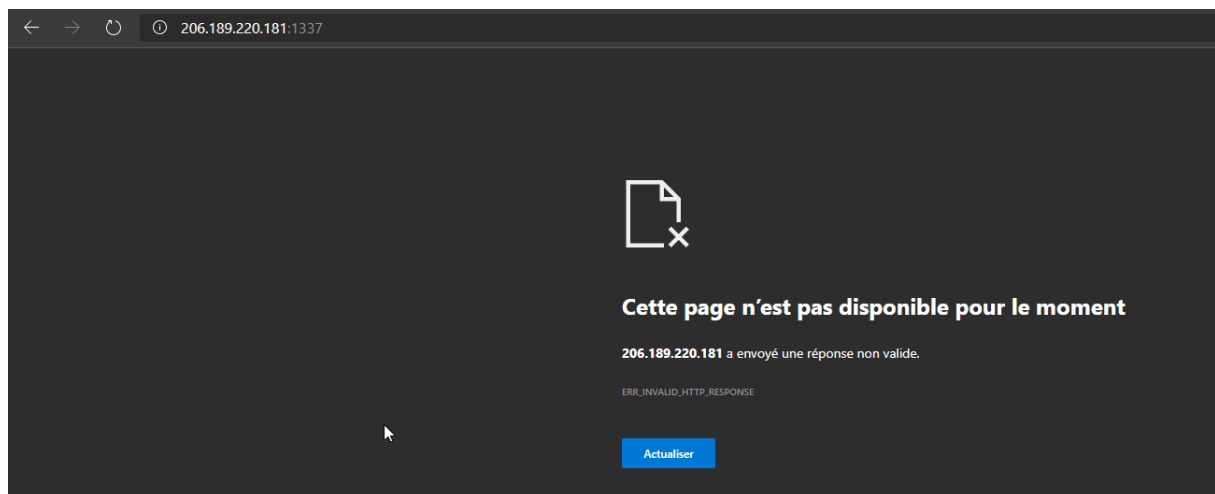


Figure 5 Error on port 1337...

Hmm... This is pretty cool in fact. It means it's not a web server answering, but something answers!

Let's check what happen with netcat.

```
D:\Desktop {git}
{lamb} nc 206.189.220.181 1337
ls
ss
qsdqsdqs
Command not found, please use 'help' for more info
```

Figure 6 netcat works!

Yeah! That means the software working on port 1337 wait on us, and even ask us to write *'help'* if we want more info... Of course, we want! Let's see what happen with the given command.

```
help
list - list all the files in this directory
get <filename> - upload the file to web server, can be downloaded from port 8080
```

*Figure 7 help output*

So, help give us the two-following command:

- **list**
- **get <filename>**

Well, let's see what the files we have here.

```
list
cmdLog.txt
fileTransfer.py
output.log
run.sh
secret1
secret10
secret11
secret12
secret13
secret14
secret15
secret16
secret17
secret18
secret19
secret2
secret20
secret21
secret22
secret23
secret24
secret25
secret26
secret27
secret28
secret29
secret3
secret30
secret31
secret32
secret33
secret34
secret35
secret36
secret37
secret38
secret39
secret4
secret40
secret41
secret42
secret43
secret44
secret45
secret46
secret47
secret48
secret49
secret5
secret50
secret6
secret7
secret8
secret9
start.sh
```

*Figure 8 list output*

Well... That's a looooot of files..... Anyway, let's try to do a **get run.sh**.

```
get run.sh
File transferred, please go to the web server to download it.
```

Figure 9 get run.sh output

OK, so it might have been sent to **206.189.220.181:8080**. Let's check it out.



Figure 10 8080 showing run.sh as expected

Oh, yeah indeed. Well, let's go get it all!

```
get cmdLog.txt
File transferred, please go to the web server to download it.
get fileTransfer.py
File transferred, please go to the web server to download it.
get output.log
File transferred, please go to the web server to download it.
get secret*
File transferred, please go to the web server to download it.
```

Figure 11 putting all files to port 8080



Non sécurisé | 206.189.220.181:8080

# Index of /

---

<a href="#">../</a>		
<a href="#">cmdLog.txt</a>	23-Dec-2019 22:38	3607
<a href="#">fileTransfer.py</a>	23-Dec-2019 22:38	3730
<a href="#">output.log</a>	23-Dec-2019 22:38	213K
<a href="#">run.sh</a>	23-Dec-2019 22:38	11
<a href="#">secret1</a>	23-Dec-2019 22:38	8304
<a href="#">secret10</a>	23-Dec-2019 22:38	8304
<a href="#">secret11</a>	23-Dec-2019 22:38	8304
<a href="#">secret12</a>	23-Dec-2019 22:38	8304
<a href="#">secret13</a>	23-Dec-2019 22:38	8304
<a href="#">secret14</a>	23-Dec-2019 22:38	8304
<a href="#">secret15</a>	23-Dec-2019 22:38	8304
<a href="#">secret16</a>	23-Dec-2019 22:38	8304
<a href="#">secret17</a>	23-Dec-2019 22:38	8304
<a href="#">secret18</a>	23-Dec-2019 22:38	8304
<a href="#">secret19</a>	23-Dec-2019 22:38	8304
<a href="#">secret2</a>	23-Dec-2019 22:38	8304
<a href="#">secret20</a>	23-Dec-2019 22:38	8304
<a href="#">secret21</a>	23-Dec-2019 22:38	8304
<a href="#">secret22</a>	23-Dec-2019 22:38	8304
<a href="#">secret23</a>	23-Dec-2019 22:38	8304
<a href="#">secret24</a>	23-Dec-2019 22:38	8304
<a href="#">secret25</a>	23-Dec-2019 22:38	8304
<a href="#">secret26</a>	23-Dec-2019 22:38	8304
<a href="#">secret27</a>	23-Dec-2019 22:38	8304
<a href="#">secret28</a>	23-Dec-2019 22:38	8304
<a href="#">secret29</a>	23-Dec-2019 22:38	8304
<a href="#">secret3</a>	23-Dec-2019 22:38	8304
<a href="#">secret30</a>	23-Dec-2019 22:38	8304
<a href="#">secret31</a>	23-Dec-2019 22:38	8304
<a href="#">secret32</a>	23-Dec-2019 22:38	8304
<a href="#">secret33</a>	23-Dec-2019 22:38	8304
<a href="#">secret34</a>	23-Dec-2019 22:38	8304
<a href="#">secret35</a>	23-Dec-2019 22:38	8440
<a href="#">secret36</a>	23-Dec-2019 22:38	8304
<a href="#">secret37</a>	23-Dec-2019 22:38	8304
<a href="#">secret38</a>	23-Dec-2019 22:38	8304
<a href="#">secret39</a>	23-Dec-2019 22:38	8304
<a href="#">secret4</a>	23-Dec-2019 22:38	8304
<a href="#">secret40</a>	23-Dec-2019 22:38	8304
<a href="#">secret41</a>	23-Dec-2019 22:38	8304
<a href="#">secret42</a>	23-Dec-2019 22:38	8304
<a href="#">secret43</a>	23-Dec-2019 22:38	8304
<a href="#">secret44</a>	23-Dec-2019 22:38	8304
<a href="#">secret45</a>	23-Dec-2019 22:38	8304
<a href="#">secret46</a>	23-Dec-2019 22:38	8304
<a href="#">secret47</a>	23-Dec-2019 22:38	8304
<a href="#">secret48</a>	23-Dec-2019 22:38	8304
<a href="#">secret49</a>	23-Dec-2019 22:38	8304
<a href="#">secret5</a>	23-Dec-2019 22:38	8304
<a href="#">secret50</a>	23-Dec-2019 22:38	8304
<a href="#">secret6</a>	23-Dec-2019 22:38	8304
<a href="#">secret7</a>	23-Dec-2019 22:38	8304
<a href="#">secret8</a>	23-Dec-2019 22:38	8304
<a href="#">secret9</a>	23-Dec-2019 22:38	8304
<a href="#">start.sh</a>	23-Dec-2019 22:38	99



Figure 12 8080 showing all the files

Awesome. Now, let's download everything from the website to be able to investigate it.

```
D:\Desktop {git}
{lamb} wget -r -l1 http://206.189.220.181:8080/
--2019-12-23 16:44:21-- http://206.189.220.181:8080/
Connecting to 206.189.220.181:8080... connected.
HTTP request sent, awaiting response... 200 OK
```

Figure 13 downloading everything from 8080

Nice. So, let's give an eye on what is secret1.

[illegible]

Figure 14 content of secret1

Well, that's not human-readable. At least, not everything. But we get a nice info here: it's an **ELF** file. Which stands for **Executable and Linkable Format**. A line executable file, in other words. Let's run it.

```
shiirosan@DESKTOP-8H0A55E:/mnt/d/Desktop/Crack/Cracked/TCC CTF2$ ./secret1
lolol, got trolled.
shiirosan@DESKTOP-8H0A55E:/mnt/d/Desktop/Crack/Cracked/TCC CTF2$
```

Figure 15 secret1 is just a troll file... damn

Oh no... That's just for troll... God damn... I hope they are not all like this ... Let's check it fast.

*[I converted them all to hex before, that's why they are called 1.hex 2.hex et caetera]*

```

shiirosan@DESKTOP-8H0A55E:/mnt/d/Desktop/Crack/Cracked/TCC CTF2$ md5sum *.hex
5b3923922031f26c8f801a3c8ff96d8f 10.hex
5b3923922031f26c8f801a3c8ff96d8f 11.hex
5b3923922031f26c8f801a3c8ff96d8f 12.hex
5b3923922031f26c8f801a3c8ff96d8f 13.hex
5b3923922031f26c8f801a3c8ff96d8f 14.hex
5b3923922031f26c8f801a3c8ff96d8f 15.hex
5b3923922031f26c8f801a3c8ff96d8f 16.hex
5b3923922031f26c8f801a3c8ff96d8f 24.hex
5b3923922031f26c8f801a3c8ff96d8f 25.hex
5b3923922031f26c8f801a3c8ff96d8f 26.hex
5b3923922031f26c8f801a3c8ff96d8f 27.hex
5b3923922031f26c8f801a3c8ff96d8f 28.hex
5b3923922031f26c8f801a3c8ff96d8f 29.hex
5b3923922031f26c8f801a3c8ff96d8f 2.hex
5b3923922031f26c8f801a3c8ff96d8f 30.hex
5b3923922031f26c8f801a3c8ff96d8f 31.hex
5b3923922031f26c8f801a3c8ff96d8f 32.hex
5b3923922031f26c8f801a3c8ff96d8f 33.hex
5b3923922031f26c8f801a3c8ff96d8f 34.hex
1f8691fffa2e35b9aac69e98d893dd47 35.hex
5b3923922031f26c8f801a3c8ff96d8f 36.hex
5b3923922031f26c8f801a3c8ff96d8f 37.hex
5b3923922031f26c8f801a3c8ff96d8f 38.hex
5b3923922031f26c8f801a3c8ff96d8f 39.hex
5b3923922031f26c8f801a3c8ff96d8f 3.hex
5b3923922031f26c8f801a3c8ff96d8f 40.hex
5b3923922031f26c8f801a3c8ff96d8f 41.hex
5b3923922031f26c8f801a3c8ff96d8f 42.hex
5b3923922031f26c8f801a3c8ff96d8f 43.hex
5b3923922031f26c8f801a3c8ff96d8f 44.hex
5b3923922031f26c8f801a3c8ff96d8f 45.hex
5b3923922031f26c8f801a3c8ff96d8f 46.hex
5b3923922031f26c8f801a3c8ff96d8f 47.hex
5b3923922031f26c8f801a3c8ff96d8f 48.hex
5b3923922031f26c8f801a3c8ff96d8f 49.hex
5b3923922031f26c8f801a3c8ff96d8f 4.hex
5b3923922031f26c8f801a3c8ff96d8f 50.hex
5b3923922031f26c8f801a3c8ff96d8f 5.hex
5b3923922031f26c8f801a3c8ff96d8f 6.hex
5b3923922031f26c8f801a3c8ff96d8f 7.hex
5b3923922031f26c8f801a3c8ff96d8f 8.hex
5b3923922031f26c8f801a3c8ff96d8f 9.hex

```

Figure 16 md5c checksum compare

They all have same md5 sign, **except** 35.hex! Let's see what contains 35.hex 😊

000005c0	0a 20 00 48 85 c0 74 02	ff d0 48 83 c4 08 c3 00	. .H..t...H....
000005d0	ff 35 d2 09 20 00 ff 25	d4 09 20 00 0f 1f 40 00	.5.. ..%.. ..@.
000005e0	ff 25 d2 09 20 00 68 00	00 00 00 e9 e0 ff ff ff	.%.. .h.....
000005f0	ff 25 ca 09 20 00 68 01	00 00 00 e9 d0 ff ff ff	.%.. .h.....
00000600	ff 25 c2 09 20 00 68 02	00 00 00 e9 c0 ff ff ff	.%.. .h.....
00000610	ff 25 ba 09 20 00 68 03	00 00 00 e9 b0 ff ff ff	.%.. .h.....
00000620	ff 25 d2 09 20 00 66 90	00 00 00 00 00 00 00 00	.%.. .f.....
00000630	31 ed 49 89 d1 5e 48 89	e2 48 83 e4 f0 50 54 4c	1.I..^H..H...PTL
00000640	8d 05 3a 02 00 00 48 8d	0d c3 01 00 00 48 8d 3d	...H.....H.=
00000650	e6 00 00 00 ff 15 86 09	20 00 f4 0f 1f 44 00 00	.....D...
00000660	48 8d 3d a9 09 20 00 55	48 8d 05 a1 09 20 00 48	H.=.. .UH.... .H
00000670	39 f8 48 89 e5 74 19 48	8b 05 5a 09 20 00 48 85	9.H..t.H..Z. .H.
00000680	c0 74 0d 5d ff e0 66 2e	0f 1f 84 00 00 00 00 00	.t.]..f.....
00000690	5d c3 0f 1f 40 00 66 2e	0f 1f 84 00 00 00 00 00	]...@.f.....
000006a0	48 8d 3d 69 09 20 00 48	8d 35 62 09 20 00 55 48	H.=i. .H.5b. .UH
000006b0	29 fe 48 89 e5 48 c1 fe	03 48 89 f0 48 c1 e8 3f	).H..H...H..H..?
000006c0	48 01 c6 48 d1 fe 74 18	48 8b 05 21 09 20 00 48	H..H..t.H...!. .H
000006d0	85 c0 74 0c 5d ff e0 66	0f 1f 84 00 00 00 00 00	..t.]..f.....
000006e0	5d c3 0f 1f 40 00 66 2e	0f 1f 84 00 00 00 00 00	]...@.f.....
000006f0	80 3d 19 09 20 00 00 75	2f 48 83 3d f7 08 20 00	.=.. ..u/H.=.. .
00000700	00 55 48 89 e5 74 0c 48	8b 3d fa 08 20 00 e8 0d	.UH..t.H.=.. ...
00000710	ff ff ff e8 48 ff ff ff	c6 05 f1 08 20 00 01 5d	....H..... ..
00000720	c3 0f 1f 80 00 00 00 00	f3 c3 66 0f 1f 44 00 00	.....f..D...
00000730	55 48 89 e5 5d e9 66 ff	ff ff 55 48 89 e5 48 83	UH..].f...UH..H.
00000740	ec 50 89 7d bc 48 89 75	b0 64 48 8b 04 25 28 00	.P.}.H.u.dH..%(.
00000750	00 00 48 89 45 f8 31 c0	48 b8 45 6c 65 33 46 33	..H.E.1.H.Ele3F3
00000760	36 42 48 89 45 c7 c6 45	cf 00 48 b8 64 76 69 65	6BH.E..E..H.dvie
00000770	73 69 70 6d 48 ba 77 64	65 71 70 76 63 65 48 89	sipmH.wdeqpvcH.
00000780	45 d0 48 89 55 d8 48 b8	63 67 7a 73 76 6e 62 71	E.H.U.H.cgzsvnbq
00000790	48 ba 73 69 6d 6e 78 77	6a 6c 48 89 45 e0 48 89	H.simnxwjlH.E.H.
000007a0	55 e8 66 c7 45 f0 62 74	c6 45 f2 00 48 8b 45 b0	U.f.E.bt.E..H.E.
000007b0	48 83 c0 08 48 8b 10 48	8d 45 c7 48 89 d6 48 89	H...H..H.E.H..H.
000007c0	c7 e8 4a fe ff ff 89 45	c0 83 7d c0 00 75 24 48	..J....E..}.u\$H
000007d0	8d 45 d0 48 89 c6 48 8d	3d b7 00 00 00 b8 00 00	.E.H..H.=.....
000007e0	00 00 e8 19 fe ff ff 48	8d 3d c0 00 00 00 e8 ed	.....H.=.....
000007f0	fd ff ff b8 00 00 00 00	48 8b 4d f8 64 48 33 0c	.....H.M.dH3..
00000800	25 28 00 00 00 74 05 e8	e4 fd ff ff c9 c3 66 90	%(...t.....f..
00000810	41 57 41 56 49 89 d7 41	55 41 54 4c 8d 25 7e 05	AWAVI..AUATL.%~.
00000820	20 00 55 48 8d 2d 7e 05	20 00 53 41 89 fd 49 89	.UH..~. .SA..I..
00000830	f6 4c 29 e5 48 83 ec 08	48 c1 fd 03 e8 77 fd ff	.L).H...H....w...
00000840	ff 48 85 ed 74 20 31 db	0f 1f 84 00 00 00 00 00	.H..t 1.....
00000850	4c 89 fa 4c 89 f6 44 89	ef 41 ff 14 dc 48 83 c3	L..L..D..A...H..
00000860	01 48 39 dd 75 ea 48 83	c4 08 5b 5d 41 5c 41 5d	.H9.u.H...[A\A]
00000870	41 5e 41 5f c3 90 66 2e	0f 1f 84 00 00 00 00 00	A^A_..f.....
00000880	f3 c3 00 00 48 83 ec 08	48 83 c4 08 c3 00 00 00	....H...H.....
00000890	01 00 02 00 41 77 65 73	6f 6d 65 2c 20 74 68 65	....Awesome, the
000008a0	20 66 6c 61 67 20 69 73	3a 20 25 73 0a 00 6f 72	flag is: %s..or
000008b0	20 69 73 20 69 74 3f 00	01 1b 03 3b 3c 00 00 00	is it?....;<...
000008c0	06 00 00 00 18 fd ff ff	88 00 00 00 68 fd ff ff	.....h...
000008d0	b0 00 00 00 78 fd ff ff	58 00 00 00 82 fe ff ff	....x...X.....
000008e0	c8 00 00 00 58 ff ff ff	e8 00 00 00 c8 ff ff ff	....X.....
000008f0	30 01 00 00 00 00 00 00	14 00 00 00 00 00 00 00	0.....
00000900	01 7a 52 00 01 78 10 01	1b 0c 07 08 90 01 07 10	.zR..x.....
00000910	14 00 00 00 1c 00 00 00	18 fd ff ff 2b 00 00 00	.....+...
00000920	00 00 00 00 00 00 00 00	14 00 00 00 00 00 00 00	.....
00000930	01 7a 52 00 01 78 10 01	1b 0c 07 08 90 01 00 00	.zR..x.....
00000940	24 00 00 00 1c 00 00 00	88 fc ff ff 50 00 00 00	\$.....P...
00000950	00 0e 10 46 0e 18 4a 0f	0b 77 08 80 00 3f 1a 3b	...F..J..w...?.;
00000960	2a 33 24 22 00 00 00 00	14 00 00 00 44 00 00 00	*3\$".....D...
00000970	b0 fc ff ff 08 00 00 00	00 00 00 00 00 00 00 00	.....
00000980	1c 00 00 00 5c 00 00 00	b2 fd ff ff d4 00 00 00	....\.....

Figure 17 hex display of secret35

Oh, no more 'troll', cool. Many things trigger me right now.....

Awesome, the flag is: %s

Ele3F36B

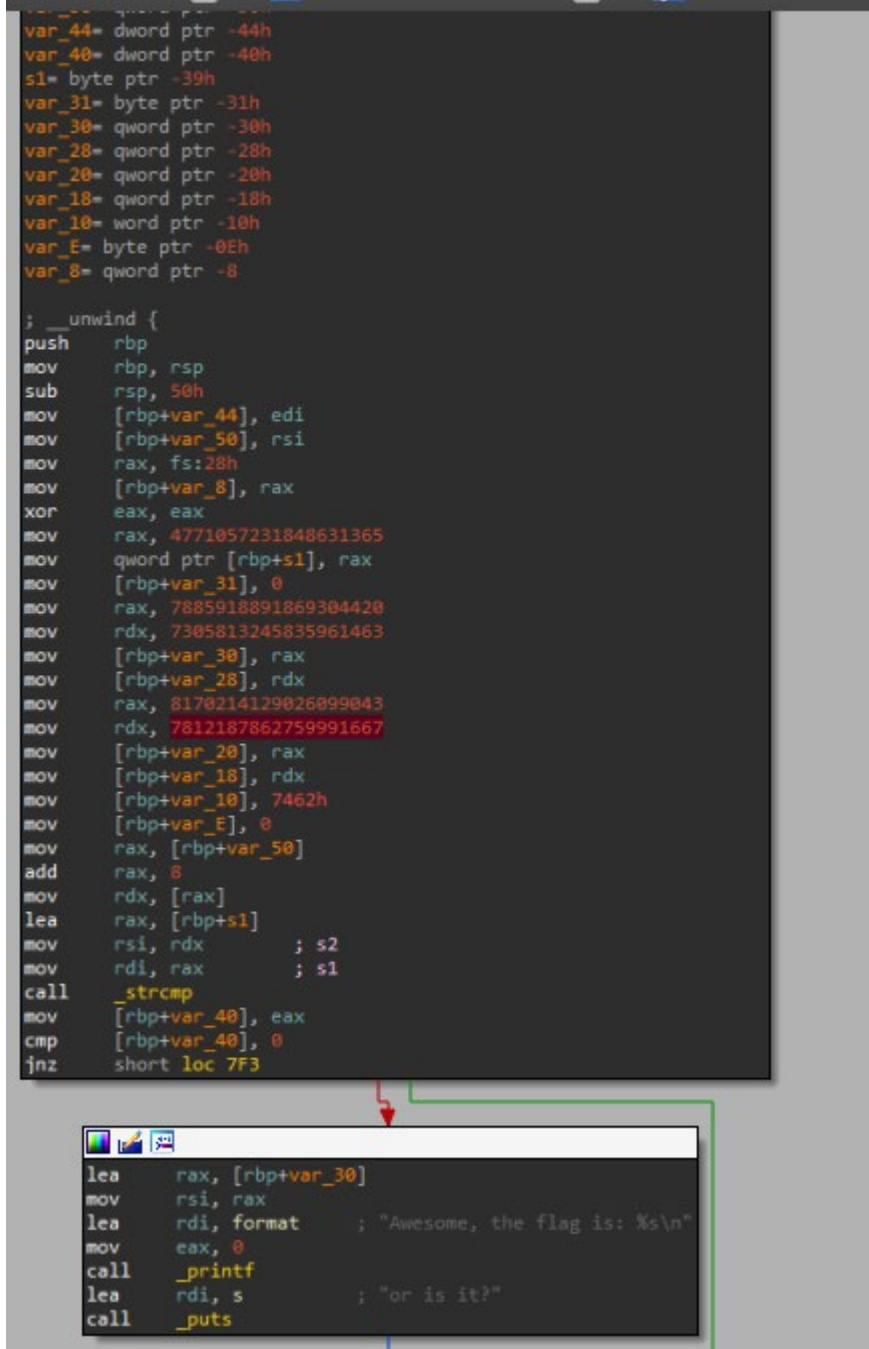
dviesipmwdeqpvcgzswnbqsimnxwjlbt

These string seems to normal for being generated by compiler... Let's run secret35 and see what happen.

```
shiirosan@DESKTOP-8H0A55E:/mnt/d/Desktop/Crack/Cracked/TCC CTF2$ ./secret35
Segmentation fault (core dumped)
```

Figure 18 strange error with secret35

Well, ok? I guess? Uh... I don't really know. Let's open it on IDA Pro and see what could happen in



```
var_44= dword ptr -44h
var_40= dword ptr -40h
s1= byte ptr -39h
var_31= byte ptr -31h
var_30= qword ptr -30h
var_28= qword ptr -28h
var_20= qword ptr -20h
var_18= qword ptr -18h
var_10= word ptr -10h
var_E= byte ptr -0Eh
var_8= qword ptr -8

; __unwind {
push    rbp
mov     rbp, rsp
sub     rsp, 50h
mov     [rbp+var_44], edi
mov     [rbp+var_50], rsi
mov     rax, fs:28h
mov     [rbp+var_8], rax
xor     eax, eax
mov     rax, 4771057231848631365
mov     qword ptr [rbp+s1], rax
mov     [rbp+var_31], 0
mov     rax, 7885918891869304420
mov     rdx, 7305813245835961463
mov     [rbp+var_30], rax
mov     [rbp+var_28], rdx
mov     rax, 8170214129026099043
mov     rdx, 7812187862759991667
mov     [rbp+var_20], rax
mov     [rbp+var_18], rdx
mov     [rbp+var_10], 7462h
mov     [rbp+var_E], 0
mov     rax, [rbp+var_50]
add     rax, 8
mov     rdx, [rax]
lea     rax, [rbp+s1]
mov     rsi, rdx          ; s2
mov     rdi, rax          ; s1
call    _strcmp
mov     [rbp+var_40], eax
cmp     [rbp+var_40], 0
jnz     short loc_7F3

lea     rax, [rbp+var_30]
mov     rsi, rax
lea     rdi, format      ; "Awesome, the flag is: %s\n"
mov     eax, 0
call    _printf
lea     rdi, s            ; "or is it?"
call    _puts
```

normal scenario.

Figure 19 loaded secret35 in IDA

Nice,

*Awesome, the flag is: %s* is back. After getting called by a **jnz**. Let's analyze it fast.

```

1  #include <stdio.h>
2  #include <string.h>
3
4  int main ()
5  {
6      char* s1 = "Ele3F36B";
7      char* var_30 = "dviesipmwdeqpvccegzsvnbqsimnxwjlbt";
8      char* s2 = argv[1];
9      puts ("Please enter the password:");
10     gets(&s2);
11     if(strcmp(s1, s2))
12         return 0;
13
14     printf("Awesome, the flag is: %s\n", &var_30);
15     puts("or is it?");
16     return 0;
17 }

```

Figure 20 pseudo c++ code post analysis

Well, that means if I send Ele3F36B as an argument of secret35, I'll get the flag (indeed, we already have it but... Let's see if the analysis is right or not)

```

shiirosan@DESKTOP-8H0A55E:/mnt/d/Desktop/Crack/Cracked/TCC CTF2$ ./secret35 Ele3F36B
Awesome, the flag is: dviesipmwdeqpvccegzsvnbqsimnxwjlbt
or is it?

```

Figure 21 getting the code

Yeah, it works just right. But... Remember the instruction?

*flag is a string that MUST start with "ctf" as the first three letters.*

That's clearly not the case here.... Let's compare what **we know** with what **we have**.

*d* should be a *c*, *v* should be a *t*, and *i* should be a *f*

Let's see the padding used there. For simplifying it, I'll use ASCII decimal value.

```

15  d -1 d
16  100 99
17  v -2 t
18  118 116
19  i -3 f

```

Figure 22 counting the padding between letters

Oh, ok. Seems kinda easy. We're just decreasing when progression in the chain. Which means *e* should be *a*, and keeps going. Let's make a script for it.

```

1  using System;
2  using System.Globalization;
3
4  namespace TCC_CTF2_Challenge2_Solver
5  {
6      class Program
7      {
8          static void Main(string[] args)
9          {
10             string encrypted = "dviesipmwdeqpvcecgzsvnbqsimnxwjlbt";
11             string decrypted = "";
12             int j;
13             for (int i = 0; i < encrypted.Length; i++)
14             {
15
16                 decrypted += (char)(encrypted[i] - (i + 1));
17             }
18             Console.WriteLine(decrypted);
19         }
20     }
21 }
22

```

Console de débogage Microsoft Visual Studio

ctfancienZZechTURUg\_aXKYZORR[YKLAR

Figure 23 First result isn't convincing...

Well... Nearly good. But nah, it's still give us info, it might be a sentence looking like this:

*Ctfancient??ech??*

And I guess everything else is kinda messed up. Let's see... I can easily guess ZZ should be tt, according to ASCII table, Z is 90, t is 116. And d is 100. As d is the 10<sup>th</sup> letter, that seems legit we get Z and not t according to the source code... We need to stay in lower letter, so let's improve a little bit our code.

```

1  using System;
2  using System.Globalization;
3
4  namespace TCC_CTF2_Challenge2_Solver
5  {
6      class Program
7      {
8          static void Main(string[] args)
9          {
10             string encrypted = "dviesipmwdeqpvcecgzsvnbqsimnxwjlbt";
11             string decrypted = "";
12             int j;
13             for (int i = 0; i < encrypted.Length; i++)
14             {
15                 if (((int)encrypted[i] - (i + 1)) < 97)
16                 {
17                     decrypted += (char)(122 - (i - (encrypted[i] - 97)));
18                 }
19                 else
20                 {
21                     decrypted += (char)(encrypted[i] - (i + 1));
22                 }
23             }
24             Console.WriteLine(decrypted);
25         }
26     }
27 }
28
29
30

```

Console de débogage Microsoft Visual Studio

ctfancienttechnologyarestillusef[[".

Figure 24 this is waaaay better!

Well, I guess we got it. *Ctfancienttechnologyarestillusef[[".*

Yeah. That's shouldn't be a fkin [ but u. Let's add a little things to patch it out reaaaaaally quickly, then we'll send the code and see if it worked or not (who knows?)

```
h.cs  C:\Users\shirosan\source\repos\TCC_CTF2_Challenge2_Solver\TCC_CTF2_Challenge2_Solver\Program.cs
TCC_CTF2_Challenge2_Solver  TCC_CTF2_Challenge2_Solver.Program

using System;
using System.Globalization;

namespace TCC_CTF2_Challenge2_Solver
{
    class Program
    {
        static void Main(string[] args)
        {
            string encrypted = "dviesipmwdeqpvcecgzsvnbqsimnxwjlbt";
            string decrypted = "";
            int j;
            for (int i = 0; i < encrypted.Length; i++)
            {
                if (((int)encrypted[i] - (i + 1)) < 97)
                {
                    if (i == 32) //quick patch to get it working nicely :ok_hand:
                        j = i - 26;
                    else
                        j = i;
                    decrypted += (char)(122 - (j - (encrypted[i] - 97)));
                }
                else
                {
                    decrypted += (char)(encrypted[i] - (i + 1));
                }
            }
            Console.WriteLine(decrypted);
        }
    }
}
```

Console de débogage Microsoft Visual Studio

ctfancienttechnologyarestilluseful

Figure 25 shhht, that's just a quick patch :D

This time, the output is correct. Let's submit it!

```
D:\Desktop {git}
{lamb} nc 2b2tmcpe.org 6517
Welcome to flag validator, please enter your flag and your discord ID in this format:
<flag> <discordID>
example: U2F0IE5vdiAzMCAyMToyODoyMiBQU1QgMjAxOQo 293420269832503306
ctfancienttechnologyarestilluseful 320082578457624589
Invalid flag, please try again.
ctfancienttechnologyarestilluseful 320082578457624589
Congratulations on completing the medium challenge, your discord ID have been logged.
^C
```

(Bad copy & paste at first god damn)

Yeah! That worked!

Let's try ... Part 1 and 3 now!