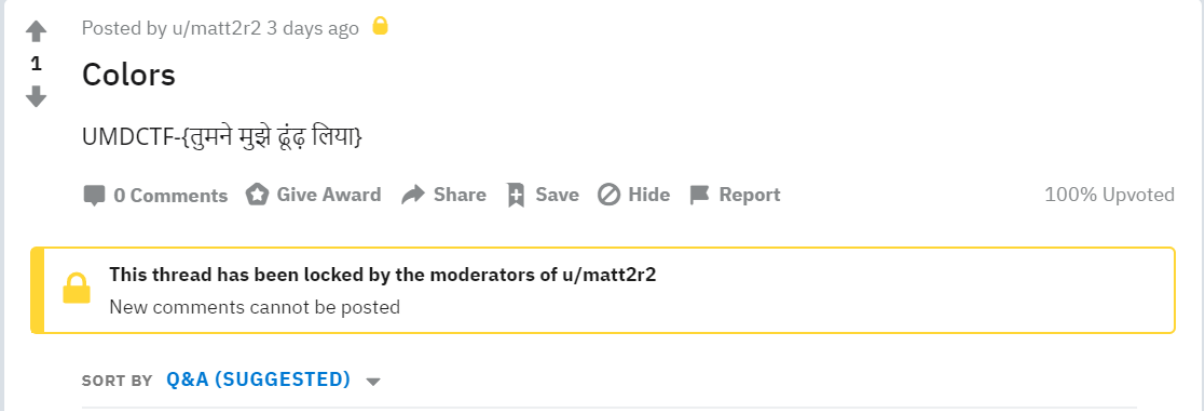
#chal Oh hi nyan 50 (Misc)

“Lumpus's friend matt2r2 really loves the Internet! You can talk to people from all over the world...”

So we’re given a hint that matt2r2 loves the internet, so we’ll start off with a simple google search: for “matt2r2”

The first link that we get is: https://www.reddit.com/user/matt2r2/comments/g23kk6/colors/



Voila here’s the flag: UMDCTF-{तुमने मुझे ढूंढ़ लिया}

#chal MemeCTF 200 (Misc)

I wish I was better at OSINT :(

`9012389aad4eb9be53d225c4bbe72098ebdb37b97a52893171ff1bce0d40f383`

Here’s what we are given^. So first things first we analyse the hash through https://www.tunnelsup.com/hash-analyzer/ so we know that it’s a SHA 256 hash, on decrypting it gives us “OSINT” as the plain text. So that’s a dead end.

On googling the hash, we come across this link: https://github.com/UMD-CSEC/hmm, in this repository we go through the commits history to find something interesting: there’s a commit named “Added Flag File”. On opening the commit: https://github.com/UMD-CSEC/hmm/commit/717289fb4574a7785e133afe59c34bbce5af4d1f?short_path=d1d85d1#diff-d1d85d151a4b9c6508208897fe6932c1

We see an image with the flag: UMDCTF-{meme_ch4llenges_ftw}

#chal Al's Advanced ATMs 100 (web)

Al says his advanced ATMs are impenetrable. I'm not so sure...

http://159.89.228.183:8087

So we're told the ATM's are impenetrable and we're given a link, let's visit the link: On going to the link we can see a simple react app with a login form! On trying basic combinations of credentials like user-password, username-password, admin-admin we can't get access.

Let's explore the source code, in a particular .js file:
http://159.89.228.183:8087/static/js/main.3d1e78a2.chunk.js

On looking carefully, we can see a username password pair. (umd-csec, VU1EQ1RGLXtvcDNuX2RAdEBfYjY0X2JAZH0=)

The password is base64 encoded, let's decode it. Voila! Here's our flag: UMDCTF-{op3n_d@t@_b64_b@d}

#chal CSEC Invasion – 1,2,3 (100,100,200 points) (web)

#1 Oh no! It looks like UMD CSEC's website has been invaded by Aliens! Can you help us fend them off?

Seems pretty easy, let's go to UMD CSEC's page: csec.umd.edu. On inspecting the source we get the flag: UMDCTF-{@l13ns_@r3_b3tt3r_th@n_hum@ns}. 1 down 2 to go! XD

#2 Good job fending off the aliens on UMD CSEC's website, but it looks like some robots have started to invade as well.

This looks like a reference to the robots.txt file, let's visit that page: https://csec.umd.edu/robots.txt
Doma Arigato Mr. Roboto ! We got out flag: UMDCTF-{d0m0_@r1g@t0_mr_r0b0t0}

#3 Those robots were tough, but are you any match for the monsters that are looking to mash their way through UMD CSEC's website.

This one's kinda tougher, hence more points too. Let's just explore all the links, web-ctf 101. On going through all the links on the source page we find an interesting one, the manifest.json file!

Towards the end of the file, we get an attribute named "calc" which has the flag: "UMDCTF-{w3_d1d_th3_m@th}"!

#chal Chungus Vault 2.0 (200) (web)

We found out the username and password of an administrator, but can't log in to the site as an administrator. Can you figure out what is going on?

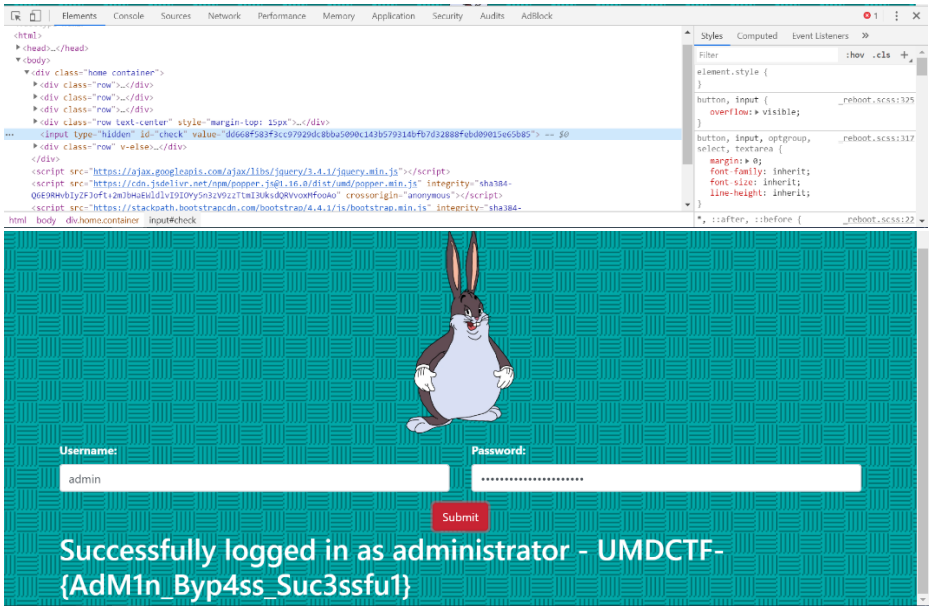
http://159.89.228.183:8088

So, we already have the admin's username password. What else could be missing to login as an admin, maybe a hidden field in the form or the http request? Let's explore the source code:

<input type="hidden" id="check" value="e9b7a334826ff2ff28b066a92ba8734cb1134720fc5879592247f47b8d2a17d7">.

As expected, we get a hashed value in the form of a hidden field named "check". On decrypting the hash SHA-256 hash we get user-login. Now all we need to do is generate a hash for "admin-login" & change the value for the check id through inspect element!

On replacing the SHA-256 hash for "admin-login" in the check input field through inspect element & logging in we get the flag!



Flag: UMDCTF-{AdM1n_Byp4ss_Suc3ssfu1}

#chal Easy Money (50) (Reverse Engineering)

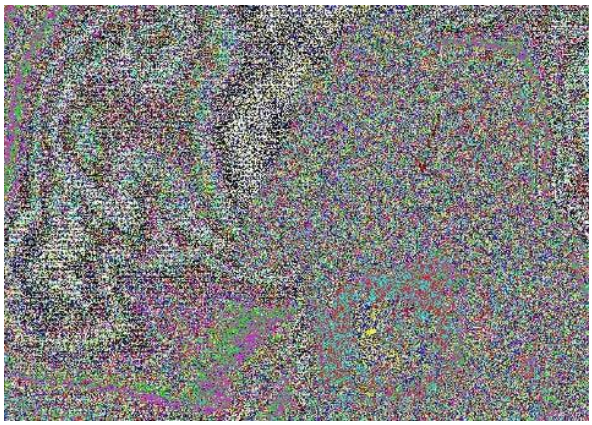
"Word on the street is this challenge is quite straightforward". In the challenge description itself we're told the challenge is quite straightforward! Let's just open notepad++ and explore the file contents, it seems to be a ELF File, towards the end of the file we find a base 64 encoded text: "VU1EQ1RGLXtlNHN5X3cxbnNfZTRzeV9tMG4zeX0=".

This give us the flag: "UMDCTF-{e4sy_w1ns_e4sy_m0n3y}"

#chal Colors (100) (Stego)

There is art in steganography. Other times, it's not what you think!

So we're given the following image:



Looking at the image it looks like a white-noise picture/rgb distortion. So, using stegonline I did a quick bit pane preview, extracted some strings and tried out the only red, blue, green, Inverse effects. Couldn't find anything. Next stop steghide.

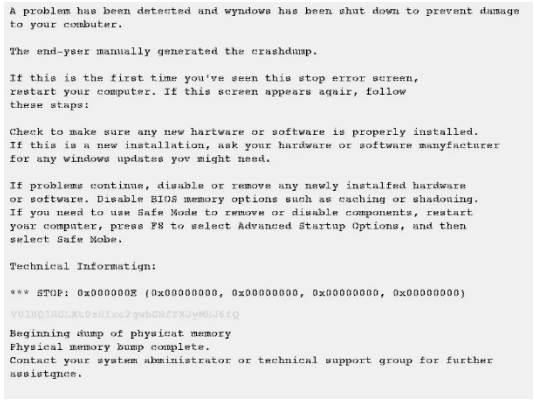
On running the steghide "steghide --extract -sf colors.jpg", it asks for a passphrase after trying all possible combinations, I finally tried: "colors" & voila! It worked 😊

Flag: UMDCTF-{Did you have some trouble? Colors are difficult :)}

#chal Crash Confusion (200) (Stego)

There's something weird about this crash screen...

So we're given a crash screen image, looking at the image we can see there is an error message written. So, using stegonline. I tried out the effects, in the LSB half filter we can see an interesting text appear out of nowhere in grey text, below the "STOP" line:



On using an online OCR tool to get the text & base64 decoding the text we get the flag! 😊

Flag: UMDCTF-{thr3sh0ld_err0rz}