CTF Challenge

- Thursday, 10 July, 2025

🕵️ Challenge – Objective Brief

You've discovered a directory named `CTF-Challenge-3` in your home directory.

🔍 **Your mission (should you choose to accept it):**

1. **Identify** all sensitive or suspicious files hidden across the structure — *they could be in plain sight or disguised.*
2. **Find at least 3 flags.** They might appear in the format `FLAG{}` or be buried in strange formats or places.
3. **Trace one trap or fake clue**. Document how it misled or attempted to mislead.
4. **Enumerate**:
   - Files with unusual or unsafe permissions
   - Files owned by users other than you
   - Hidden files or directories
   - World-writable or SUID-set files
5. **Bonus Objective** (Optional but glorious):

   Reconstruct what the **"attacker" or malicious user** was trying to do inside this structure.

   Use filenames, ownerships, timestamps, and content clues to hypothesize.

CTF Report

Analyst: `Jinay Shah [Jynx]`

Date: `July 10, 2025`

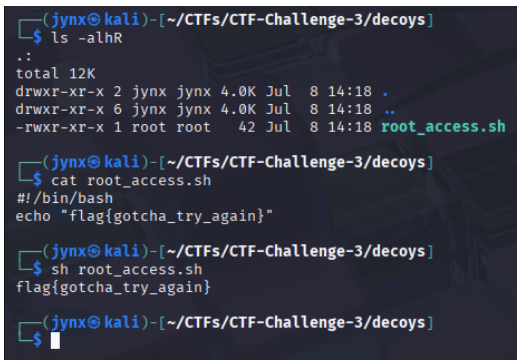
Challenge Directory: `~/CTF-Challenge-3`

Objective: Locate hidden files, capture flags, detect traps, analyze anomalies.

Flag No.	File Path	Flag Content	Extraction Technique	Notes
1	`~/CTFs/CTF-Challenge-3/decoys/root_access.sh`	`flag{gotcha_try_again}`	`sudo sh root_access.sh`	Hidden in a misleading folder in a relatively legitimate seeming file namely

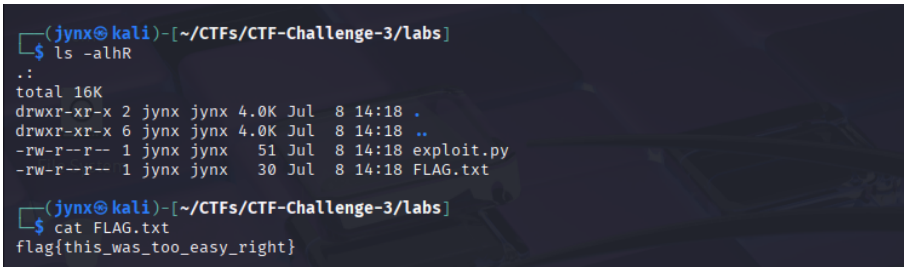
Flag No.	File Path	Flag Content	Extraction Technique	Notes
				root_access.sh

Output:



2	~/CTFs/CTF-Challenge-3/labs/FLAG.txt	flag{this_was_too_easy_right}	cat FLAG.txt	Relatively easy and simple to read precisely why its also a booby trap.

Output:

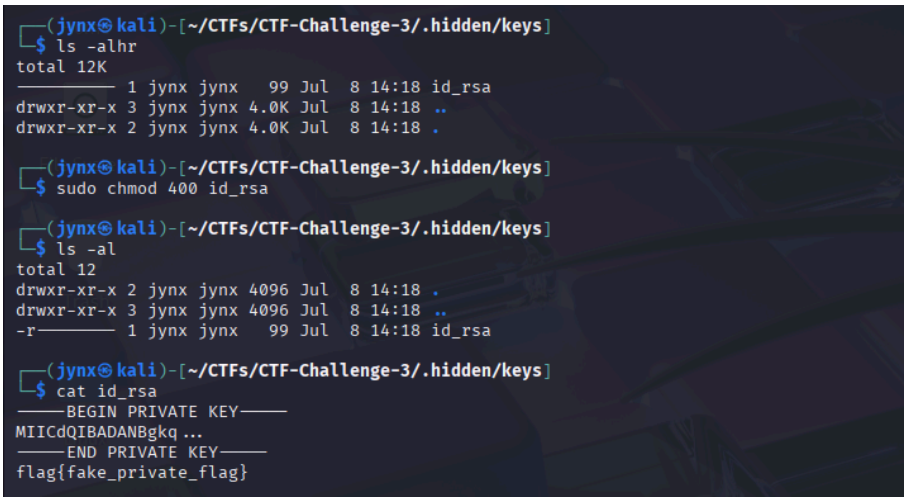


3	~/CTFs/CTF-Challenge-3/debug.log ~/CTFs/CTF-Challenge-3/.secret_ex	flag{you_set_the_shell_only_flag}	echo "U29tZXRpbWVzIHlvdSBqdXN0IG5lZWQgdG8gYmUgcm9vdCB0byBiZSB3b3J0aHkK | base64 -d

Output:

4	`~/CTFs/CTF-Challenge-3/.hidden/keys/id_rsa`	`flag{fake_private_flag}`	`cat id_rsa`	`-----BEGIN PRIVATE KEY-----MIICdQIBADANBgkq...-----END PRIVATE KEY-----` Seems like some RSA key, but still looks like a dead end.

Output:



🔍 1. Flag Captures

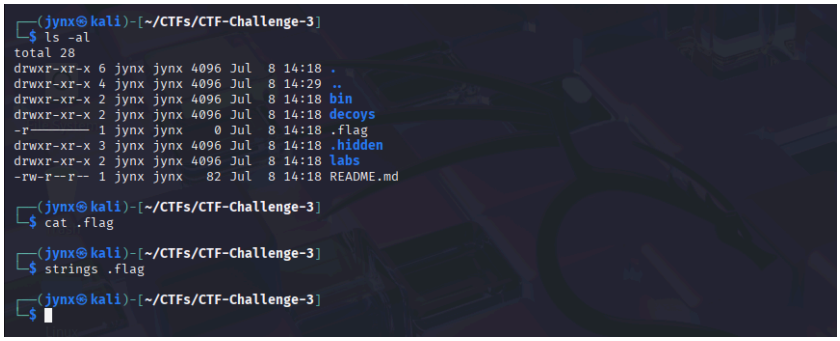
🗺️ 2. File Enumeration + Reasoning

File Path	Suspicious Why?	Permission/Ownership	Tool/Command Used
`~CTFs/CTF-Challenge-3/bin/.secret_exec`	SUID set, owned by root	`-rwsr-xr-x`	`find -perm -4000`
`~/CTFs/CTF-Challenge-3/decoys/root_access.sh`	`.sh` executable file by all the user types, which is suspicious to naked ye at glance, lead to nothing eventually though- just a red herring flag. Ironically in a `./decoys` named folder.	`-rwxr-xr-x`	`cat root_access.sh` [for safer side], but even if arbitrarily executed using `sh` it would not be exploited because its not malware or dangerous but which is not ideal nor cautious.

⚠️ 3. Traps & Distractions

Trap File	Why It's a Trap	How You Discovered It
`~/CTFs/CTF-Challenge-3/.flag`	Booby Trap [Nothing Inside]	File empty.

Output:

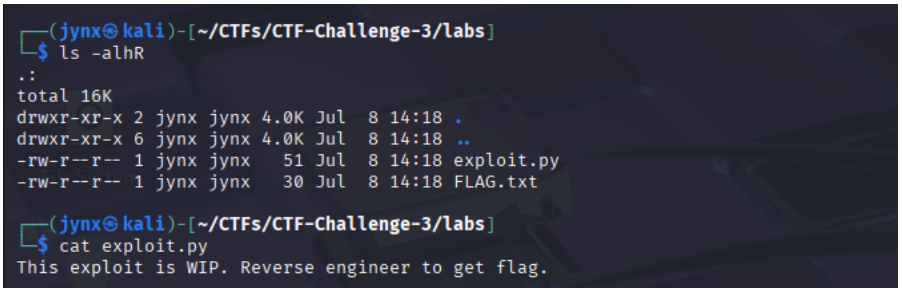


`.secret_exec`	SUID binary prints exact same flag as `debug.log` — redundancy or privilege requirement	Duplicate to debug.log(hashed), booby trap.

Output:

`exploit.py`	No embedded flag. Contains dummy message. Likely trap or future placeholder.	No Flag, dead end.

Output:



🧠 4. Techniques + Commands Used

- `ls -alhR` – full recursive listing in human readable size + long listing format.
- `sudo chmod 400` *filename* - Used to change the ownership of file with no permissions.
- `find` – advanced permission and ownership scans
- `grep` – to search within file contents

- `stat` – permission/owner/inode verification
- `cat` / `less` – content reading
- `base64 -d` - used to decode and retrieve the flag from hashed text
- Others: `file`, `strings`, `head`, `awk`, `chmod` for manipulation if needed

---

🧵 5. Narrative / Final Hypothesis

1. What do you think happened here?

   → There were extreme booby traps that seemed hiding everything but was a deep dead end and lead nowhere. Spent and wasted a lot of mental energy here and there around them. Some flags were in plain sight just needed to uncover with basecode64.

2. Was there a breach?

   → No I don't think so. There weren't any breaches because I isolated almost all the files, and only then accessed the flags from a rather passive manner. `exploit.sh` AND `.secret_exec` were the file that were suspicious ones, but lead nowhere in the sense of security vulnerability or data breaches. `.secret_exec` more so because of SUID bits but essentially nothing more than just a decoy.

3. Was a malicious user trying to exfiltrate data or escalate privileges?
   → Potentially with
   `.secret_exec` but as I mentioned forth was simply a decoy and a duplicate flag to debug.log, nothing of any value as such at the end of the day.

🕒 6. Timestamp Forensics Analysis

Due to the nature of how this CTF challenge was generated — via a scripted or single-session directory setup — file creation, modification, and change timestamps are clustered tightly around a short time frame. Additionally, file access times reflect the investigation activity rather than adversarial behavior. Therefore, while timestamps were reviewed, they did not offer independent forensic insights in this case.
