



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	<code>~/CTFs/CTF-Challenge-3/decoys/root_access.sh</code>	<code>flag{gotcha_try_again}</code>	<code>sudo sh root_access.sh</code>	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:

```
(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/decoys]
└─$ ls -alhR
.:
total 12K
drwxr-xr-x 2 jynx jynx 4.0K Jul  8 14:18 .
drwxr-xr-x 6 jynx jynx 4.0K Jul  8 14:18 ..
-rw-r--r-- 1 root root  42 Jul  8 14:18 root_access.sh

(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/decoys]
└─$ cat root_access.sh
#!/bin/bash
echo "flag{gotcha_try_again}"

(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/decoys]
└─$ sh root_access.sh
flag{gotcha_try_again}

(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/decoys]
└─$
```

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:

```
(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/labs]
└─$ ls -alhR
.:
total 16K
drwxr-xr-x 2 jynx jynx 4.0K Jul  8 14:18 .
drwxr-xr-x 6 jynx jynx 4.0K Jul  8 14:18 ..
-rw-r--r-- 1 jynx jynx   51 Jul  8 14:18 exploit.py
-rw-r--r-- 1 jynx jynx   30 Jul  8 14:18 FLAG.txt

(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/labs]
└─$ cat FLAG.txt
flag{this_was_too_easy_right}
```

3	~/CTFs/CTF-Challenge-3/debug.log	flag{you_set_the_shell_only_flag}	echo "U29tZXRpWWzIHLvdSBqdXN0IG5IZWQgdG8gYmUgcm9vdCB0byBiZSB3b3J0aHkK" base64 -d
	~/CTFs/CTF-Challenge-3/.secret_ex		

Output:

```
(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/bin]
└─$ ls -alhr
total 16K
drwxr-xr-x 2 jynx jynx 4.0K Jul 8 14:18 .
drwxr-xr-x 6 jynx jynx 4.0K Jul 8 14:18 ..
-rw-r--r-- 1 jynx jynx 109 Jul 8 14:18 debug.log
-rwsr-xr-x 1 root root 81 Jul 8 14:18 .secret_exec

(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/bin]
└─$ cat debug.log
U29tZXRpbwZlHlvdS8qdXN0IG5lZWQgdG8gYmUgc9vdCB0byB1ZSB3b3J0aHkKZmxhZt5b3Vfc2V0X3RoZV9zaGVsbF9vbmx5X2ZsYWd9

(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/bin]
└─$ echo "U29tZXRpbwZlHlvdS8qdXN0IG5lZWQgdG8gYmUgc9vdCB0byB1ZSB3b3J0aHkKZmxhZt5b3Vfc2V0X3RoZV9zaGVsbF9vbmx5X2ZsYWd9" | base64 -d
Sometimes you just need to be root to be worthy
flag{you_set_the_shell_only_flag}
(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/bin]
└─$ cat .secret_exec
Sometimes you just need to be root to be worthy
flag{you_set_the_shell_only_flag}
(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/bin]
└─$
```

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

Output:

```
(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/.hidden/keys]
└─$ ls -alhr
total 12K
      1 jynx jynx 99 Jul 8 14:18 id_rsa
drwxr-xr-x 3 jynx jynx 4.0K Jul 8 14:18 ..
drwxr-xr-x 2 jynx jynx 4.0K Jul 8 14:18 .

(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/.hidden/keys]
└─$ sudo chmod 400 id_rsa

(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/.hidden/keys]
└─$ ls -al
total 12
drwxr-xr-x 2 jynx jynx 4096 Jul 8 14:18 .
drwxr-xr-x 3 jynx jynx 4096 Jul 8 14:18 ..
-r----- 1 jynx jynx 99 Jul 8 14:18 id_rsa

(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/.hidden/keys]
└─$ cat id_rsa
-----BEGIN PRIVATE KEY-----
MIICdQIBADANBgkq ...
-----END PRIVATE KEY-----
flag{fake_private_flag}
```

1. Flag Captures

2. File Enumeration + Reasoning

File Path	Suspicious Why?	Permission/Ownership	Tool/Command Used
<code>~/CTFs/CTF-Challenge-3/bin/.secret_exec</code>	SUID set, owned by root	<code>-rwsr-xr-x</code>	<code>find -perm -4000</code>
<code>~/CTFs/CTF-Challenge-3/decoys/root_access.sh</code>	<code>.sh</code> 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 <code>./decoys</code> named folder.	<code>-rwxr-xr-x</code>	<code>cat root_access.sh</code> [for safer side], but even if arbitrarily executed using <code>sh</code> 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
<code>~/CTFs/CTF-Challenge-3/flag</code>	Booby Trap [Nothing Inside]	File empty.

Output:

```
(jynx㉿kali)-[~/CTFs/CTF-Challenge-3]
└─$ ls -al
total 28
drwxr-xr-x 6 jynx jynx 4096 Jul  8 14:18 .
drwxr-xr-x 4 jynx jynx 4096 Jul  8 14:29 ..
drwxr-xr-x 2 jynx jynx 4096 Jul  8 14:18 bin
drwxr-xr-x 2 jynx jynx 4096 Jul  8 14:18 decoys
-r----- 1 jynx jynx   0 Jul  8 14:18 .flag
drwxr-xr-x 3 jynx jynx 4096 Jul  8 14:18 .hidden
drwxr-xr-x 2 jynx jynx 4096 Jul  8 14:18 .labs
-rw-r--r-- 1 jynx jynx  82 Jul  8 14:18 README.md

(jynx㉿kali)-[~/CTFs/CTF-Challenge-3]
└─$ cat .flag

(jynx㉿kali)-[~/CTFs/CTF-Challenge-3]
└─$ strings .flag

(jynx㉿kali)-[~/CTFs/CTF-Challenge-3]
└─$
```

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

Output:

```
(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/bin]
└─$ ls -alhR
.:
total 16K
drwxr-xr-x 2 jynx jynx 4.0K Jul  8 14:18 .
drwxr-xr-x 6 jynx jynx 4.0K Jul  8 14:18 ..
-rw-r--r-- 1 jynx jynx 109 Jul  8 14:18 debug.log
-rwsr-xr-x 1 root root  81 Jul  8 14:18 .secret_exec

(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/bin]
└─$ cat debug.log
U29tZXRpBWzIHlvd8qdxN0IG51ZWQgdG8gYmUgcw9vdCB0byBzSB3b3J0aHKKZmxhZ3t5b3Vfc2V0X3RoZv9zaGVsbF9vbmx5X2zsYwd9

(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/bin]
└─$ echo "U29tZXRpBWzIHlvd8qdxN0IG51ZWQgdG8gYmUgcw9vdCB0byBzSB3b3J0aHKKZmxhZ3t5b3Vfc2V0X3RoZv9zaGVsbF9vbmx5X2zsYwd9" | base64 -d
Sometimes you just need to be root to be worthy
flag{you_set_the_shell_only_flag}
(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/bin]
└─$ cat .secret_exec
Sometimes you just need to be root to be worthy
flag{you_set_the_shell_only_flag}
(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/bin]
└─$
```

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

Output:

```
(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/labs]
└─$ ls -alhR
.:
total 16K
drwxr-xr-x 2 jynx jynx 4.0K Jul  8 14:18 .
drwxr-xr-x 6 jynx jynx 4.0K Jul  8 14:18 ..
-rw-r--r-- 1 jynx jynx  51 Jul  8 14:18 exploit.py
-rw-r--r-- 1 jynx jynx  30 Jul  8 14:18 FLAG.txt

(jynx㉿kali)-[~/CTFs/CTF-Challenge-3/labs]
└─$ cat exploit.py
This exploit is WIP. Reverse engineer to get flag.
```

🧠 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 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.
