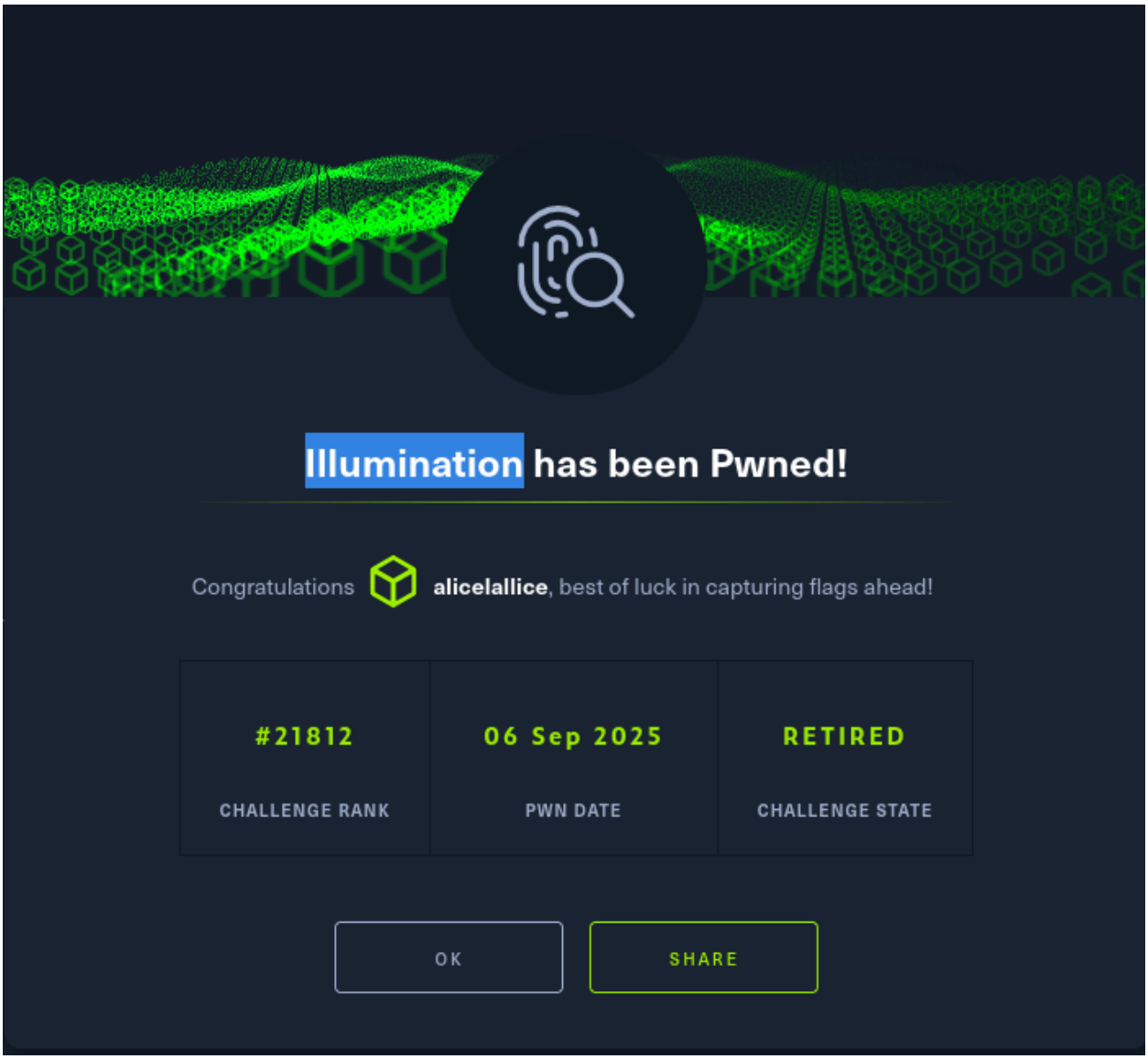
Illumination

Types	forensic
CTF	HTB



Challenge Description

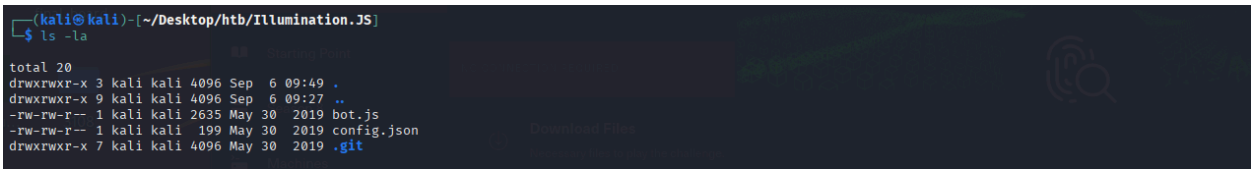
We are given a small Node.js project containing a Discord bot (`bot.js`) and a `config.json` . The challenge hints that sensitive information might be hidden in the

repository.

Initial Enumeration

Start by listing the files:

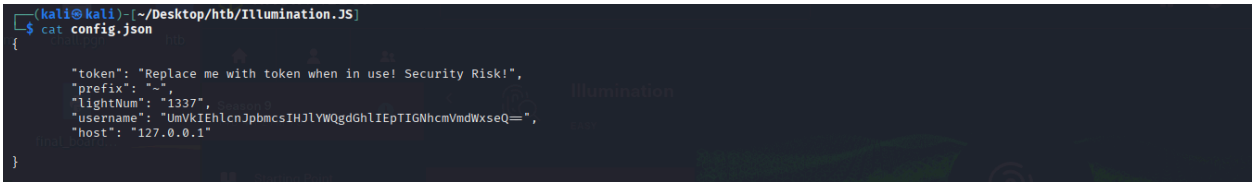
```
ls -la
```



The `.git/` folder immediately stands out. This means we can inspect the repository history for secrets.

Inspecting `config.json`

The current `config.json` contains:



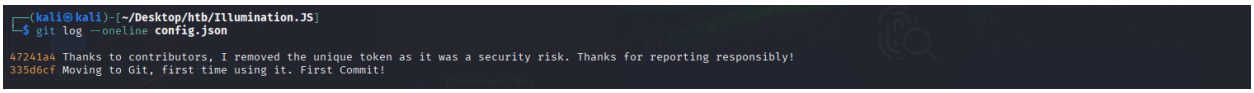
Decoding the base64 in `username`:

⚠️ This tells us the username field is a **decoy**. We need to dig deeper.

Exploring Git History

Check commits that touched `config.json`:

```
git log --oneline config.json
```

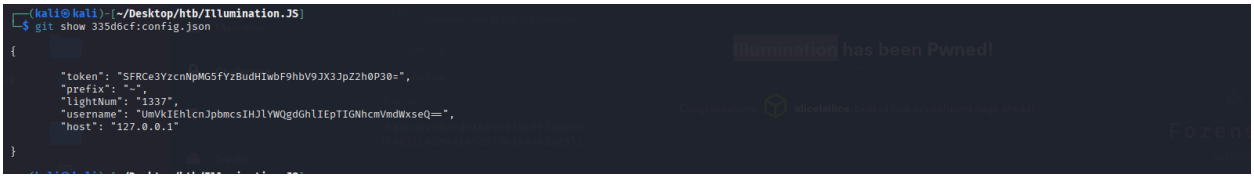


The first commit (`335d6cf`) is suspicious because it likely contains the original token.

Recovering the Old Token

Show `config.json` at commit `335d6cf`:

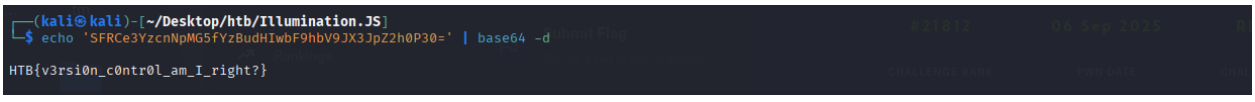
```
git show 335d6cf:config.json
```



Decoding the Token

Decode with base64:

```
echo 'SFRCe3YzcnNpMG5fYzBudHIwbF9hbV9JX3JpZ2h0P30=' | base64 -d
```



`HTB{v3rsi0n_c0ntr0l_am_I_right?}`

That's the flag.

Root Cause / Lessons Learned

- **Version control leaks**: Developers often commit secrets to Git before realizing the mistake and overwriting them. But git keeps history, so the secrets remain.
- **Base64 encoding**: The bot stored its token in base64, but that provides no security, only obfuscation.
- **Forensic tip**: Always check `.git/` or backup files in forensic challenges. Secrets love to hide in history.