Raspberry Pi Honeypot + Cloudflare + Cowrie
Full Runbook, Issues, and Fixes
(Compiled from multi■day setup + debugging)

==========
0. HIGH-LEVEL ARCHITECTURE
==========

- Device: Raspberry Pi running Debian / Raspberry Pi OS (64■bit)
- Role: Exposed “honeypot” for students to SSH into via a web browser
- Exposure: Cloudflare Tunnel + Cloudflare Access (Browser SSH)
- Honeypot engine: Cowrie SSH honeypot
- Real SSH:
  - Pi’s real SSH daemon listens on the normal SSH port (for you only).
  - Cowrie listens on TCP port 2222 on localhost.
- Cloudflare Tunnel:
  - Public hostname: honeypot.void-nano.uk
  - Service: ssh://localhost:2222
  - Cloudflared runs as a systemd service and connects your Pi to Cloudflare’s edge.
- User Flow:
  1. Student browses to honeypot.void-nano.uk
  2. Cloudflare Access prompts for email / OTP, enforces policy.
  3. After auth, Access opens a Browser SSH session to the service defined on the tunnel.
  4. Cloudflare connects to your tunnel, your tunnel forwards to localhost:2222.
  5. Cowrie accepts the SSH connection and presents the fake filesystem + challenge.

==========
1. CLOUDFLARE TUNNEL CONFIGURATION
==========

1.1 Tunnel config on the Pi

/etc/cloudflared/config.yml:

```
tunnel: a6e5db48-8c89-4bf3-b2b2-0c707af29d2f
credentials-file: /etc/cloudflared/a6e5db48-8c89-4bf3-b2b2-0c707af29d2f.json

ingress:
  - hostname: honeypot.void-nano.uk
    service: ssh://localhost:2222
  - service: http_status:404
```

Key change over time:
- tcp://localhost:22 → tcp://localhost:2222 → ssh://localhost:2222 (final)

Restart after changes:

```
sudo systemctl restart cloudflared
systemctl status cloudflared
```

==========

2. CLOUDFLARE ACCESS APPLICATION
==========

Application: “Honeypot SSH”

- Type: Self-hosted
- URL: honeypot.void-nano.uk

Public hostname:
- Subdomain: honeypot
- Domain: void-nano.uk
- Path: (blank)

Browser rendering:
- Allow automatic Cloudflared auth: ON
- Browser rendering: SSH

Policy:
- Name: Allow SSH Access
- Action: Allow
- Include: Everyone (for class)

==========
3. COWRIE SERVICE + PORT
==========

Service:

```
sudo systemctl status cowrie
```

Expect:

```
Active: active (running)
Main PID: (twistd)
```

Port:

```
sudo ss -tulpn | grep 2222
```

Should show twistd listening on 0.0.0.0:2222

Control:

```
sudo systemctl start cowrie
sudo systemctl stop cowrie
sudo systemctl restart cowrie
```

Test locally:

```
ssh -p 2222 root@localhost
```

==========

4. COWRIE HONEYFS + FS.PICKLE
==========

Key directories:

```
/home/cowrie/cowrie/honeyfs          (fake filesystem tree)
/home/cowrie/cowrie/src/cowrie/data/fs.pickle  (compiled FS index)
```

Config snippets in etc/cowrie.cfg:

```
[honeypot]
data_path = src/cowrie/data
filesystem = ${honeypot:data_path}/fs.pickle
```

Important rule:
- Any time honeyfs changes, you must rebuild fs.pickle.

Rebuild procedure (the correct one we settled on):

```
cd /home/cowrie/cowrie
sudo systemctl stop cowrie
sudo -u cowrie cowrie-env/bin/createfs -l honeyfs -d 6 -o src/cowrie/data/fs.pickle
sudo systemctl start cowrie
```

We originally deleted wrong fs.pickle copies which caused:

- SystemExit: 2 inside cowrie.ssh.fs
- Browser SSH errors: “Unable to open channel…” or “Origin closed…”

Regenerating with the above command fixed it.

==========
5. CTF FILE LAYOUT INSIDE HONEYFS
==========

honeyfs tree (simplified):

```
honeyfs/
  etc/
    cron.d/
      sys-update
    ...
  home/
    admin/
      flag.txt      (decoy flag)
    student/
      notes.txt     (hints)
  proc/ (fake /proc files)
  tmp/
    .cache/
      .data.b64
```

```
      .exfil
    usr/
      local/
        bin/
          update-system
        share/
          .cache/
            .system/
              .exfil_flag
    var/
      log/
        auth.log
        syslog
```

Key file contents:

/etc/cron.d/sys-update

```
* * * * * root /usr/local/bin/update-system
```

/home/admin/flag.txt

```
flag{this-is-not-the-real-flag}
```

/home/student/notes.txt

```
Our admin keeps sensitive info in /tmp/.cache
I saw something about base64 and “bad credentials” last time.
```

/var/log/auth.log

```
Nov 16 04:10:23 sshd[2222]: Failed password for admin from 185.222.81.21 port 55812
Nov 16 04:10:24 sshd[2222]: Failed password for admin from 185.222.81.21 port 55812
Nov 16 04:10:24 sshd[2222]: Failed password for student from 185.222.81.21 port 55812
Nov 16 04:10:25 sshd[2222]: Accepted password for backup from 185.222.81.21 port 55812
Nov 16 04:10:26 sshd[2222]: session opened for user backup
```

/usr/local/bin/update-system

```
#!/bin/bash
# fake malware - runs via cron
# exfiltrates secret flag from a hidden location
cat /usr/share/.cache/.system/.exfil_flag >> /tmp/.cache/.exfil
echo "sent" > /tmp/.cache/status
```

Hidden flag & exfil files in honeyfs:

```
sudo mkdir -p honeyfs/tmp/.cache
sudo mkdir -p honeyfs/usr/share/.cache/.system

sudo tee honeyfs/usr/share/.cache/.system/.exfil_flag >/dev/null << 'EOF'
```

```
flag{intrusion-chain-master}
EOF

sudo tee honeyfs/tmp/.cache/.exfil >/dev/null << 'EOF'
flag{intrusion-chain-master}
EOF

sudo tee honeyfs/tmp/.cache/.data.b64 >/dev/null << 'EOF'
YmFkX2NyZWRlbnRpYWxzPTRmMzJmMzQzZjMyCg==
EOF
```

After editing these, we ran createfs again as above.

==========
6. HOW STUDENTS SOLVE THE CTF
==========

Inside Browser SSH, as root@svr04:

1) Enumerate:

```
ls
ls /home
ls /home/student
cat /home/student/notes.txt
```

2) Check auth log:

```
cat /var/log/auth.log
```

See admin / student failures, backup success.

3) Check syslog (story of malware install, cron, tmp paths):

```
cat /var/log/syslog
```

4) Look in /tmp/.cache:

```
cd /tmp
ls
cd .cache
ls -al
```

5) Decode base64:

```
cat .data.b64
echo "YmFkX2NyZWRlbnRpYWxzPTRmMzJmMzQzZjMyCg==" | base64 -d
```

Gets:

```
bad_credentials=4f32f343f32
```

6) Grab the final flag:

```
cat .exfil
```

Flag:

```
flag{intrusion-chain-master}
```

==========
7. BUGS & HOW WE FIXED THEM
==========

7.1 Cowrie “SystemExit: 2” / PermitTTY error

Symptom:

- journalctl -u cowrie showed SystemExit in fs.HoneyPotFilesystem
- Browser SSH: “Unable to open channel, ensure PermitTTY is enabled”

Cause:

- Missing or invalid fs.pickle after deleting wrong file.

Fix:

```
cd /home/cowrie/cowrie
sudo systemctl stop cowrie
sudo -u cowrie cowrie-env/bin/createfs -l honeyfs -d 6 -o src/cowrie/data/fs.pickle
sudo systemctl start cowrie
```

7.2 “The origin has unexpectedly closed the connection”

Cause:

- Either cloudflared down or cowrie not responding.

Fix checklist:

```
systemctl status cloudflared
systemctl status cowrie
sudo journalctl -u cowrie -n 50
# fix fs.pickle if needed, restart services.
```

7.3 Files exist in honeyfs but students can’t see them

Cause:

- You edited honeyfs but did NOT rebuild fs.pickle.

Fix:

- Run createfs (see above).

7.4 update-system “command not found” inside honeypot

- In Cowrie, scripts are static, not always executable in a real sense.
- That’s okay for this CTF – we pre-created /tmp/.cache/.exfil and .data.b64
  so the story still works without actual cron execution.

==========
8. LIVE LOGS ON YOUR END
==========

Cowrie logs directory:

```
/home/cowrie/cowrie/var/log/cowrie
```

Files:

```
cowrie.log      (text)
cowrie.json     (JSON)
```

Follow live:

```
cd /home/cowrie/cowrie/var/log/cowrie
sudo tail -f cowrie.log
```

Systemd logs:

```
sudo journalctl -u cowrie -f
sudo journalctl -u cloudflared -f
```

==========
9. SAFE POWER-OFF / PERSISTENCE
==========

Shutdown the Pi cleanly:

```
sudo shutdown now
# or
sudo poweroff
```

On next boot:

- cloudflared.service and cowrie.service are enabled.
- They should automatically start.
- Verify:

```
systemctl status cloudflared
systemctl status cowrie
```

==========

10. QUICK COMMAND CHEAT SHEET
==========

Rebuild FS:

```
cd /home/cowrie/cowrie
sudo systemctl stop cowrie
sudo -u cowrie cowrie-env/bin/createfs -l honeyfs -d 6 -o src/cowrie/data/fs.pickle
sudo systemctl start cowrie
```

Check services:

```
systemctl status cloudflared
systemctl status cowrie
```

View live Cowrie log:

```
cd /home/cowrie/cowrie/var/log/cowrie
sudo tail -f cowrie.log
```

Final flag (for your notes):

```
flag{intrusion-chain-master}
```