This challenge was interesting because it was from a twitter post.





The link leads to the download page for the following image:



Once you have downloaded the page, you will want to open it using some version of Linux so that you have access to a variety to hacker tools that will assist you on this journey. The first thing I did was create a **giraffe** directory so that I can keep everything remotely organized. So what do we know from our challenge prompt?

* We are trying to get a link to a Discord server as the flag
* There is a secret service running on some remote server, link or IP unknown
* We are given an image file
* A password file (rockyou.txt) is given, so there will be some level of cracking involved
* Touching port 65000 is out of question (or is it?)

So the first thing you should think about when trying to find hidden data in a file is to test out the **binwalk** command, a tool for searching binary images for embedded files and executable code. I was already aware of this technique from other CTF challenges, but luckily our giraffe friend gives us a hint in case you are unfamiliar:



So first, let’s try and extract that data from the image:

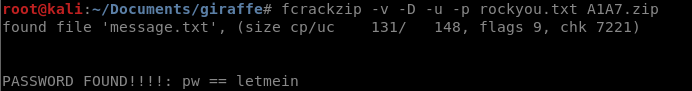


So besides the jpg data of the image, there is a zip archive, which is just a zipped file. Running the command again with the **-x** tagwill actually extract the file. If you are unsure which command to use to unzip the archive, using the **file** command on the resulting image can help.



Since it is just a zip file, we unzip it using the following command:

  
So the file is zipped with a password, meaning we can’t actually unzip it unless we know the password. When you don’t have a password you must crack it, and luckily, we received a hint with a wordlist, so we will try that first. Google searching for rockyou.txt leads to a GitHub page, from which we can download the text file. Be sure that the text file is in the same directory as your zip file, type the following command:



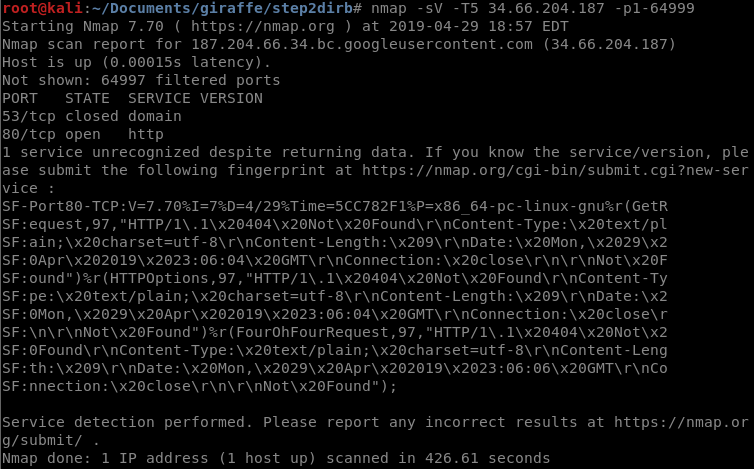
**Frcrackzip** is the name of the tool. Here is an explanation of its components (tags are case sensitive):

* The -v tag gives a more verbose output,
* -D specifies that we will be using a dictionary attack rather than brute force
* -u will make the tool actually test the passwords on our zipped file rather than just parsing the wordlist for valid passwords
* -p is to specify the wordlist, rockyou.txt in our case
* A1A7.zip is our zipped file in question

You can see that it finds the password, set to “letmein”. Re-running the **unzip** command with the password reveals a file called **message.txt**, which we can display with the **cat** command



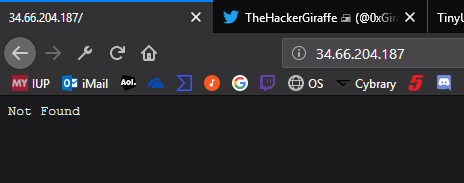
So now we have an IP address, great! For the next step, we are going to run an **nmap** scan to check for open ports, excluding 65000 for now.



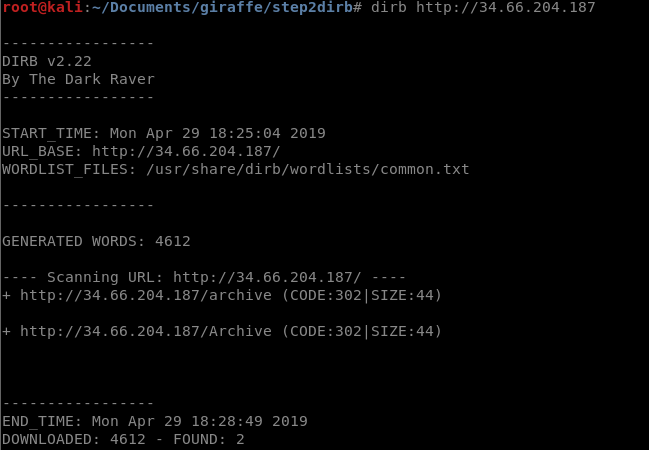
Here is a command breakdown:

* -sV does version enumeration on discovered ports, basically to figure out what service is running
* -T5 defines the rate of packets sent, with -T4 being the default. It is mostly used to slow down the scan on networks you are trying to be stealthy on, but -T5 speeds it up slightly.
* 34.66.204.187 is the target IP address
* -p1-34999 sets the port range to ports 1 to 34999 to exclude 35000. There are a few ports after 35000 but we’ll ignore these for now

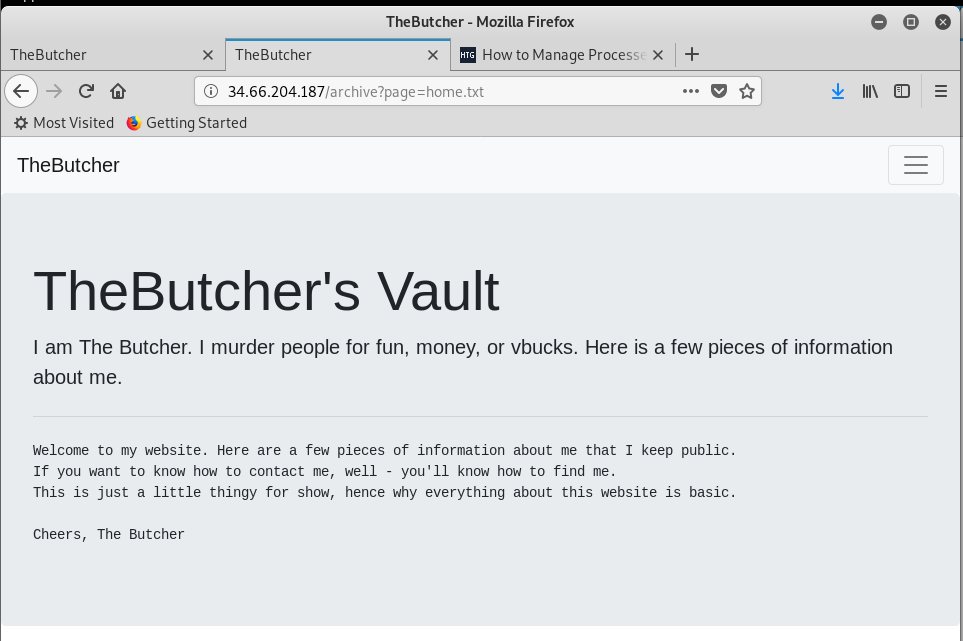
You can see from the results that there is one open port on port 80, which is running an http service. However, if we try to enter the IP address into our browser (which would normally allow us to access the service) we get an error.

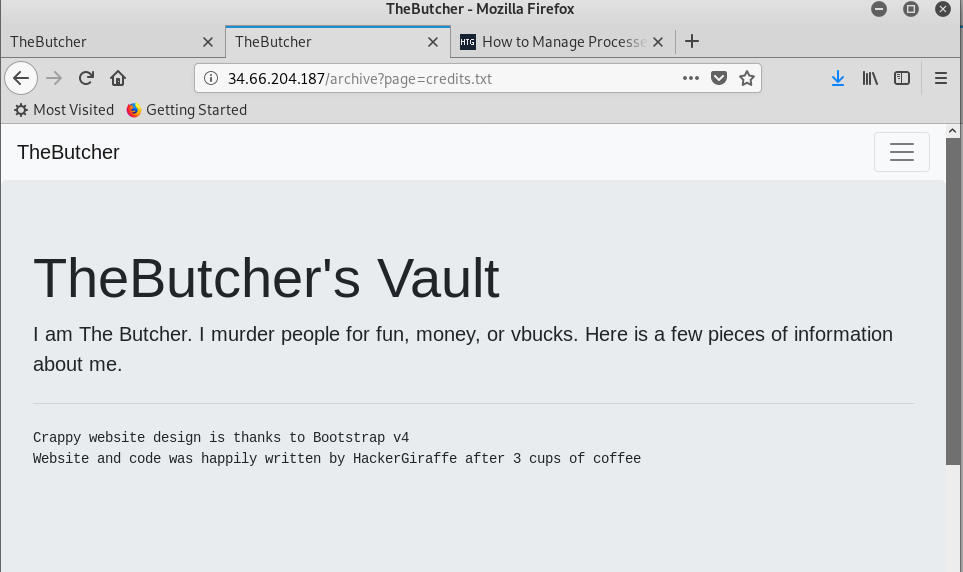


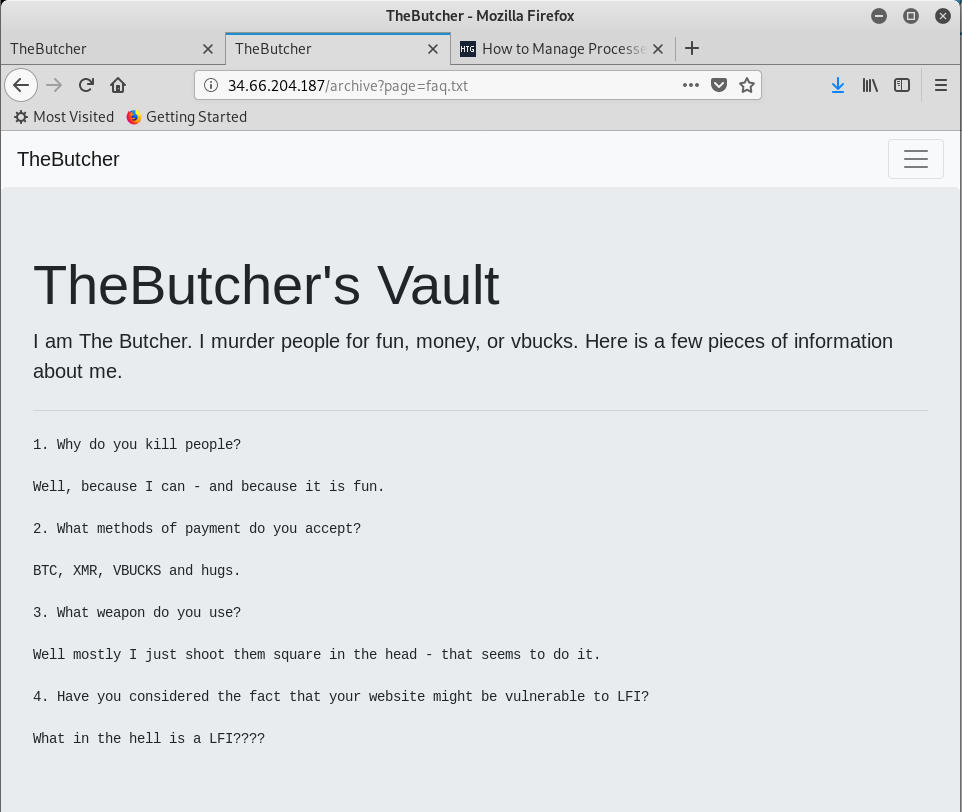
So the next step is to try and find a directory, especially since **message.txt** from earlier said we need to find one. The tool I will use is **dirb**, a tool that tests out common directories to see if they exist. You can also use wordlists with it, but we already used the given list earlier so it is unlikely we will need to use one. There are other tools such as **dirbuster** and **gobuster** that do the same thing (arguable better) but I will just use **dirb** here.



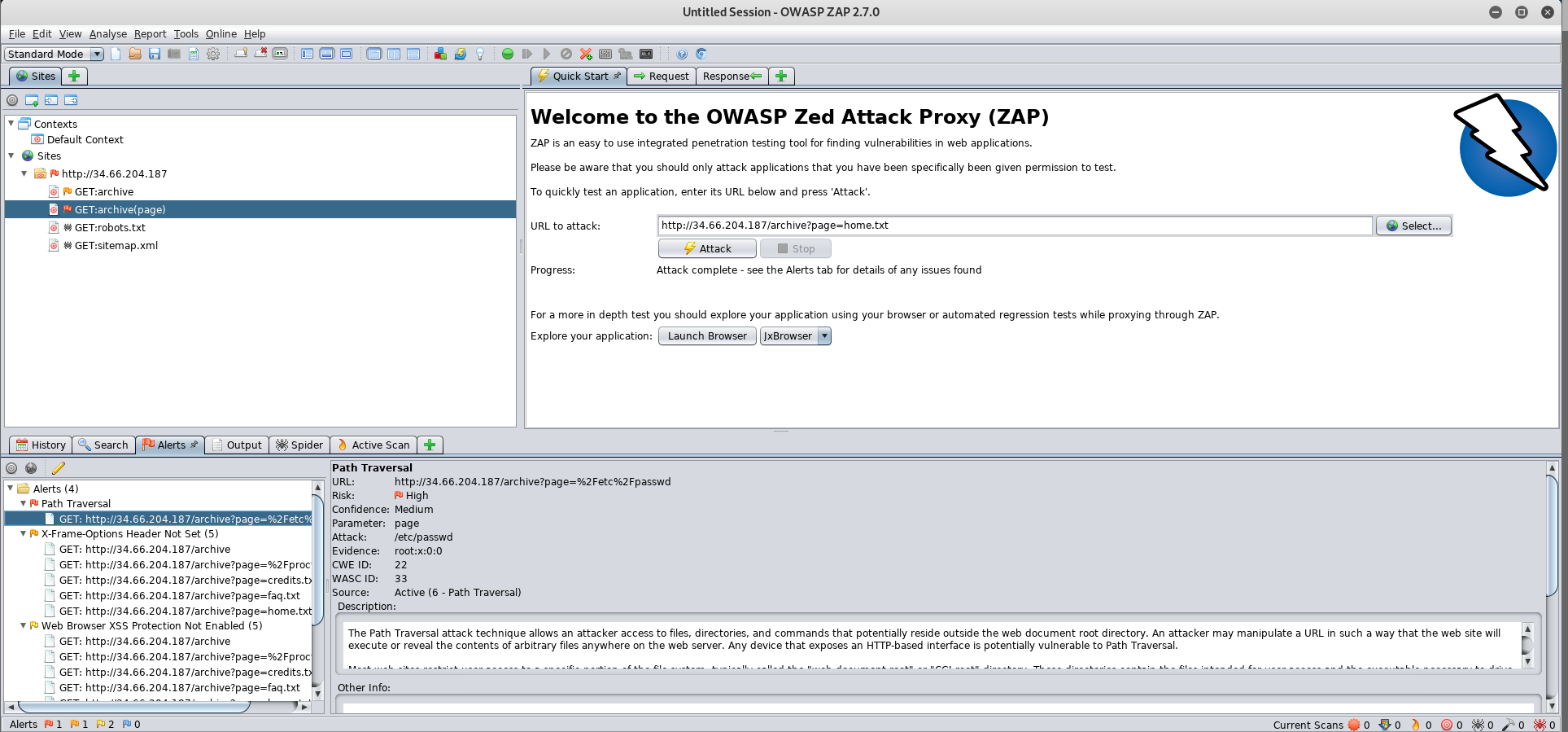
So we ended up finding two directories: **34.66.204.187/archive** and **34.66.204.187/Archive**. Navigating to these pages redirects you to **34.66.204.187/archive?page=home.txt**, which is a home page for the Butcher’s Vault. It has a few pages which I will display now:



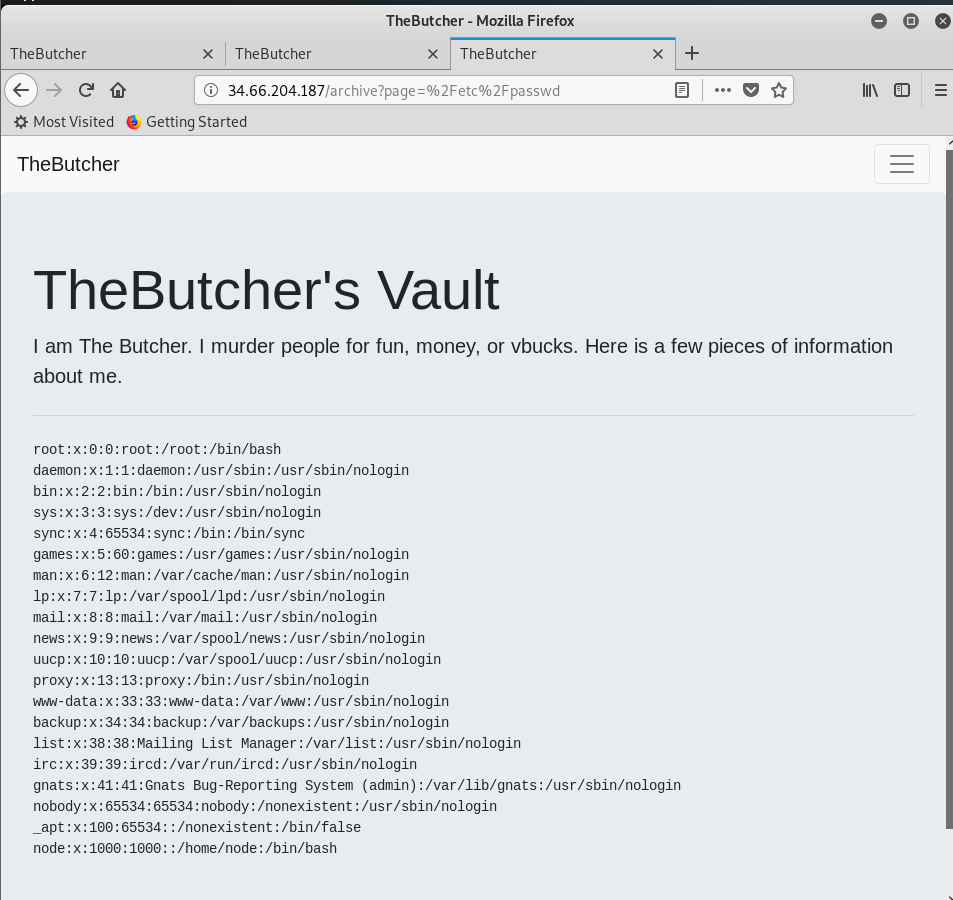




So now that we have a webpage, I will scan it using the OWASP Zap tool, a multipurpose web app testing thingy.

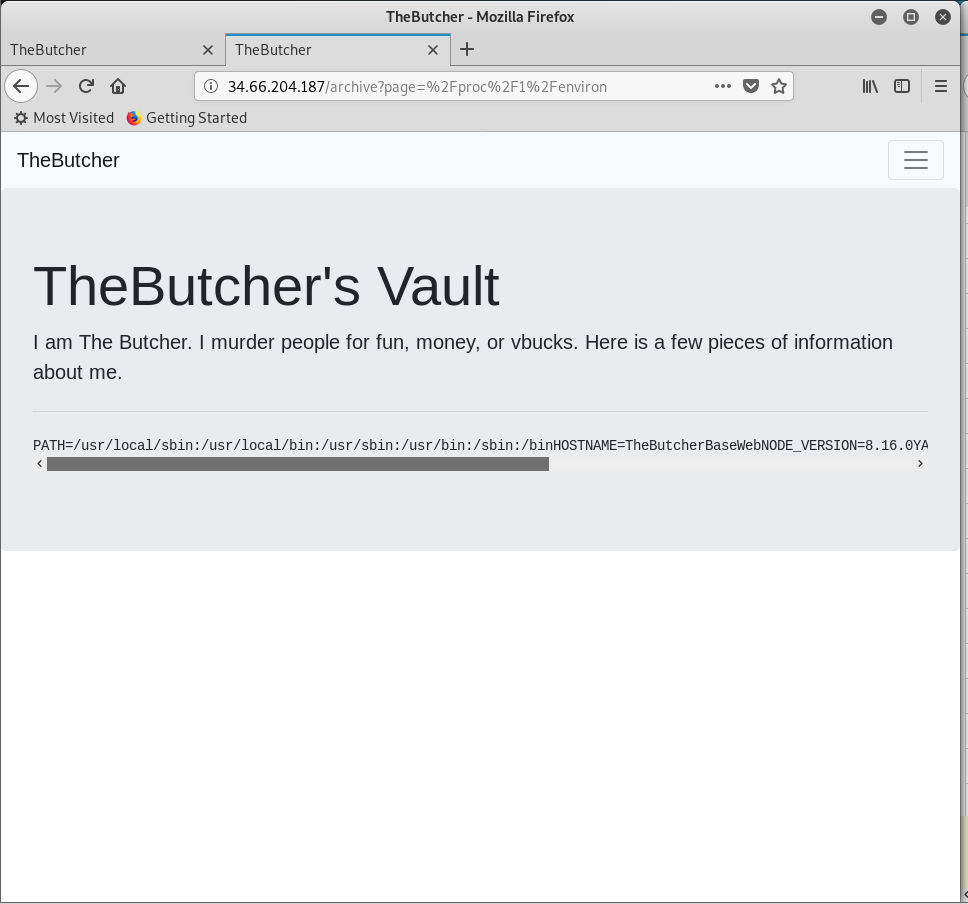


If you look at the bottom, you can see that the page is vulnerable to path traversal, an attack where escape characters allow you to access files on the host machine of a web service. When you enter the given url from OWASP Zap, you get the following page:



As you can see, the page has revealed the /etc/passwd file, which contains information about users and home directories. The reason this works is because the ?page=<something> is used to fetch files from the server, but does not properly sanitize input. %2F is the escape sequence of a ‘/’ character, so entering %2Fetc%2Fpasswd gets us to /etc/passwd. Testing this a bit further, I found that we can get the output of basically any file we want, though we can’t run any actual commands. We see at the bottom of the file that there is a user called node, so we can assume this server is running a NodeJS service somewhere.

After a LOT of digging around, I found a few things of interest. In /proc, the directory of running processes, I found several identical running processes. Within the process’s subfolder, you can check several files to learn more about the process. /cmdline shows the name of the command that is actually running the process, and /environ shows lots of helpful data about the environment in which the process is running. In our case, we get the following:



Expanded out, we get:

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

HOSTNAME=TheButcherBaseWeb

NODE\_VERSION=8.16.0

YARN\_VERSION=1.15.2

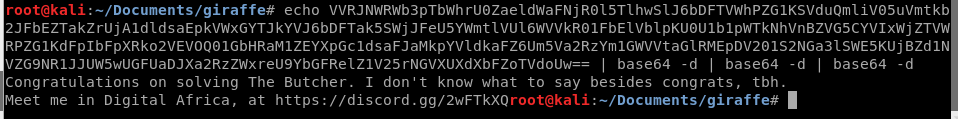
SECRET\_SERVICE\_PASSWORD=ilovebeef

HOME=/home/node

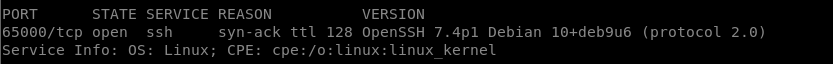
Great, we now have the credentials for a hidden node.js service on the server. All we need to do now is look for ports again on the box and try to connect to the service using a tool such as netcat. Scanning the full range of ports reveals an unknown service on port 1337, so connect there:



We already have the password of ilovebeef, so we enter it. The result is a base64 encoded string (you can tell by the general appearance and the fact that it ends with ==) so we must decode it. It turns out that the string is encoded multiple times so we simply keep piping it into **base64 -d** until we get the result, a string containing the flag (in this case, the Discord link to Digital Africa 2.0)



Just for fun, I used my VPN to switch to an IP that I don’t care if it gets banned and scanned port 35000. Here is what we find there:



It is an **SSH** port, probably used for server management, so we will leave that alone for now. I tried pinging the server again after scanning port 65000 and it seems that I can still connect, so everything is fine.